МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«Харківський політехнічний інститут»

Н. К. Стратієнко, М. Д. Годлевський, І. О. Бородіна

АЛГОРИТМИ І СТРУКТУРИ ДАНИХ: ПРАКТИКУМ

Навчальний посібник

для студентів спеціальностей
«Комп'ютерні науки» та
«Інженерія програмного забезпечення»

Рекомендовано Вченою радою НТУ «ХПІ»

Харків
НТУ «ХПІ»
2017

УДК 510.5
ББК 22.18я7
С 83

Рецензенти:
В. М. Левикін, д-р техн. наук, проф., Харківський національний університет радіоелектроніки;
О. Є. Федорович, д-р техн. наук, проф., Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут»

Рекомендовано Вченою Радою НТУ «ХПІ» як навчальний посібник для студентів спеціальностей «Комп'ютерні науки» та «Інженерія програмного забезпечення», протокол №6 від 07.07.2017р.

Стратієнко Н .К.
С 83 Алгоритми і структури даних: практикум: навч. посіб. / Н. К. Стратієнко, М. Д. Годлевський, І. О. Бородіна. – Харків: НТУ «ХПІ», 2017. – 224 с.

ISBN 978-617-05-0247-6

Навчальний посібник містить приклади розв'язання типових задач, питання для самоконтролю, задачі для аудиторних занять, варіанти індивідуальних домашніх завдань, завдання для лабораторного практикуму і курсового проектування за такими розділами, як базові структури даних, математичні основи теорії алгоритмів, алгоритмічні стратегії, алгоритми сортування, машина Тюрінга, генератори псевдовипадкових чисел, основні алгоритми на графах і геометричні алгоритми.

Призначено для студентів, викладачів, аспірантів, розробників програмного забезпечення, слухачів післядипломної освіти всіх форм навчання.

Іл. 123. Табл. 9. Бібліогр. 23 назви

УДК 510.5
ББК 22.18я7

ISBN 978-617-05-0247-6

ЗМІСТ

ВСТУП

Мета навчального посібника – закріплення, поглиблення розуміння студентами теоретичних знань, набуття стійких практичних навичок, пов'язаних з використанням різноманітних структур даних, а також зі створенням, модифікацією та аналізом алгоритмів.

Складність задач, що виникають при розробці програмного забезпечення систем різноманітного призначення, потребує не лише глибокого знання студентами теорії структур даних і алгоритмів, але й стійких практичних навичок в їх використанні.

Навчальний посібник містить три основних розділи, які поєднують практичні, лабораторні заняття та курсове проектування.

У першому розділі посібника наведені загальні рекомендації щодо проведення практичних занять і виконання індивідуальних домашніх розрахункових завдань, для кожної теми практичного заняття подано короткий теоретичний вступ, докладно описується розв'язання типового прикладу, наводяться запитання для самоконтролю. В даному розділі також наведені тести і варіанти індивідуальних завдань, вимоги до їх виконання і оформлення.

Другий розділ містить рекомендації щодо організації лабораторного практикуму. Наведено перелік тем лабораторних робіт, до кожної теми подано теоретичний вступ, загальні рекомендації щодо виконання роботи та варіанти індивідуальних завдань.

У третьому розділі розглядаються питання курсового проектування. Розглянуто задачі виконання та мета курсової роботи, наведено завдання для курсового проектування, описано основні етапи і календарний план виконання курсового проекту, вимоги до програмного забезпечення, що розробляється,

вимоги до змісту пояснювальної записки до курсового проекту. Окрім того, наведено загальний порядок захисту курсового проекту, вимоги до презентаційних матеріалів, докладу, демонстрації програмного забезпечення і критерії оцінювання проекту.

Посібник призначений для широкого кола читачів: студентів, викладачів, аспірантів, розробників програмного забезпечення, слухачів післядипломної освіти всіх форм навчання.

1. ПРАКТИЧНІ ЗАНЯТТЯ

1.1. Загальні рекомендації щодо проведення практичних занять

План проведення практичних занять подано в табл. 1.1.

Таблиця 1.1 – План проведення практичних занять

№	Теми практичних занять	Кількість годин
1	Математичні основи теорії алгоритмів	2
2	Алгоритми сортування	4
3	Алгоритмічні стратегії	2
4	Машина Тюрінга	2
5	Генератори псевдовипадкових чисел	2
6	Фундаментальні алгоритми на графах і деревах	2
7	Геометричні алгоритми	2

На основі матеріалів, розглянутих на лекції, студентам пропонується виконати ряд завдань, направлених на закріплення теоретичного матеріалу.

На початку кожного практичного заняття студентам пропонуються питання для самоперевірки. Дані питання дозволяють не лише перевірити рівень засвоєння теоретичного матеріалу, але й поєднати вивчений матеріал з практикою.

Для перевірки отриманих знань, пов’язаних з основними поняттями і визначеннями, класифікацією структур даних та алгоритмів, пропонується тест (див. підрозділ 1.9).

На початку навчального семестру кожен студент отримує комплексне індивідуальне домашнє розрахункове завдання, метою якого є закріплення навичок розв’язання практичних задач. Варіанти даних завдань, а також вимоги до їх виконання та оформлення наводяться в підрозділі 1.9.

1.2. Математичні основи теорії алгоритмів

1.2.1. Короткі теоретичні відомості

Для оцінки складності алгоритмів використовують часову та просторову складності.

Часова складність алгоритму (в гіршому випадку) – це функція

від розміру вхідних даних, яка дорівнює максимальній кількості елементарних операцій, що здійснює алгоритм для розв'язання екземпляра задачі вказаного розміру.

Для оцінки просторової складності алгоритмів розглядають обсяг пам'яті, що використовується.

Аналізуючи алгоритм, можна спробувати знайти точну кількість виконуваних ним дій, але в більшості випадків достатньо оцінити асимптоту зростання часу роботи алгоритму, коли розмір входу прямує до нескінченості.

Для запису асимптотичної складності алгоритмів використовують такі асимптотичні позначення: $O(n)$, $\Theta(n)$, $\Omega(n)$, $o(n)$ і $w(n)$.

Запис $f(n) = \Theta(g(n))$ означає, що знайдуться такі константи $c_1, c_2 > 0$ і таке n_0, що $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ для всіх $n \ge n_0$.

Запис $f(n) = \mathrm{O}(g(n))$ означає, що знайдеться така константа $c > 0$ і таке n_0, що $0 \le f(n) \le cg(n)$ для всіх $n \ge n_0$.

Запис $T(n) = \Omega(n)$ означає, що знайдеться така константа $c > 0$ і таке n_0, що $0 \le cg(n) \le f(n)$ для всіх $n \ge n_0$.

Говорять, що $f(n) = o(g(n))$, якщо для будь-якого додатного ε знайдеться таке n_0, що $0 \le f(n) \le \varepsilon g(n)$ при всіх $n \ge n_0$.

Говорять, що $f(n) = w(g(n))$, якщо для будь-якого додатного ε знайдеться таке n_0, що $0 \le \varepsilon g(n) \le f(n)$ при всіх $n \ge n_0$.

Найважливіші функції, які використовуються для оцінки складності алгоритмів:

1) логарифм: $f(n) = \log_a n$ і полілогарифм: $f(n) = \log_a^b n$;

2) багаточлен: $p(n) = \sum_{i=0}^{d} a_i n_i$;

3) експонента: $f(n) = a^n$;

4) факторіал: $f(n) = n!$.

1.2.2. Приклади розв'язання задач

Задача 1

Визначити складність алгоритму, записаного у вигляді псевдокоду:

```
for k=1 to m
  for i=1 to m
    for j=1 to m
      if a[i,k]-a[k,j]≤a[i,j]
        a[i,j]=a [i,k]-a[k,j].
```

Розв'язання

Псевдокод – мова описання алгоритмів, яка використовує ключові слова мов програмування, але не враховує подробиці та специфічний синтаксис. Детальніше про псевдокод можна прочитати в авторів книги [1].

Як було зазначено, часом роботи алгоритму називається число елементарних кроків, які він виконує. Вважаємо, що один рядок псевдокоду потребує не більше фіксованого числа операцій.

Позначимо біля кожного рядка алгоритму його вартість (число операцій) і кількість разів, яку даний рядок буде виконуватись (табл. 1.2).

Таблиця 1.2 – Вартість операцій і кількість разів виконання рядків алгоритму

№ рядка	Алгоритм	Кількість операцій (вартість)	Скільки разів буде виконаний рядок
1	`for k=1 to m`	c_1	$m+1$
2	`  for i=1 to m`	c_2	$m(m+1)$
3	`    for j=1 to m`	c_3	$m^2(m+1)$
4	`      if a[i,k]-a[k,j]≤a[i,j]`	c_4	m^3
5	`        a[i,j] =a [i,k]-a[k,j]`	c_5	$\leq m^3$

Перший рядок виконуватиметься $(m+1)$ разів. Для кожного k від 1 до m підрахуємо, скільки разів буде виконаний другий рядок псевдокоду, для кожного i від 1 до m підрахуємо, скільки разів буде виконаний третій рядок, для кожного j від 1 до m підрахуємо, скільки разів будуть виконуватись 4-й та 5-й рядки.

Поєднавши внески всіх рядків, отримаємо:

$$T(m) = c_1(m+1) + c_2 m(m+1) + c_3 m^2(m+1) + c_4 m^3 + c_5 m^3 =$$
$$= (c_3 + c_4 + c_5)m^3 + (c_2 + c_3)m^2 + (c_1 + c_2)m + c_1.$$

Перетворимо даний вираз, відкинувши члени менших порядків і коефіцієнт при m. Тоді отримаємо $T(m) = O(m^3)$.

Задача 2

Порівняти швидкості зростання заданих функцій, визначивши, чи можна дану пару визначити, як $f(n) = O(g(n))$:

$$f(n) = \sin^k(n),\ g(n) = \log^k(n),\ n,k > 1.$$

Розв'язання

Якщо k – непарне число, то функція $f(n) = \sin^k(n)$ може бути від'ємною, а за визначенням $O(n)$ функції мають бути невід'ємні, а значить, функції $f(n)$ і $g(n)$ не можна порівнювати при непарному значенні k.

Якщо k – парне число, то $0 \le f(n) \le 1$, а значить, $\sin^k(n) = O(\log^k(n))$.

Задача 3

Користуючись O-позначенням, довести:

$$n^2 + 5n = O(n^2).$$

Розв'язання

Позначимо ліву функцію у виразі, який необхідно довести, як $f(n) = n^2 + 5n$. Праву функцію позначимо $g(n) = n^2$.

Тоді у відповідності до визначення O-позначення необхідно вказати такі константи $c > 0$ і n_0, що $\forall n \ge n_0$ буде правильним вираз $0 \le n^2 + 5n \le cn^2$.

Розділимо ліву і праву частини даного виразу на n. Отримаємо

$$0 \le 1 + \frac{5}{n} \le c.$$

Тепер легко вибрати константи, при яких даний вираз залишається правильним. Наприклад, виберемо значення $c = 2$ і $n_0 = 5$. Для $n \ge 5$ (тобто $\forall n \ge n_0$) вираз $(1 + \frac{5}{n})$ зменшуватиметься, а значить, твердження $n^2 + 5n = O(n^2)$ є правильним.

Запитання для самоконтролю

1. Що таке псевдокод?

2. Що називається часом роботи алгоритму?

3. Як визначається складність алгоритму?

4. Перелічіть найважливіші функції, які використовуються для визначення складності алгоритмів. Як за швидкістю зростання пов'язані дані функції?

Задачі для аудиторних занять

1. Визначити складність алгоритмів, записаних у вигляді псевдокоду.

1)
```
for k = 1 to n
    for i = 1 to n
        for j = 1 to n
            a[i][j] = min(a[i][j], a[i][k] + a[k][j])
```

2)
```
is_prime = true
for k = 2 to n – 1
  if n % k == 0
    is_prime = false
  break
```

3)
```
for j = 1 to n
  for i = 1 to j – 1
    if a[i] > a[i + 1]
      //Swap a[i] and a[i + 1]
      a[i], a[i + 1] = a[i + 1], a[i]
```

4)
```
for i = 1 to n - 1
  min_index = i
  for j = i + 1 to n
    if a[min_index] > a[j]
      min_index = j
  if min_index ≠ i
        //Swap a[min_index] and a[i]
        a[min_index], a[i] = a[i], a[min_index]
```

5)
```
merged = []
while !empty(a) && !empty(b)
  if a[1] < b[1]
    merged.append(a[1])
    a.pop_front()
  else:
    merged.append(b[1])
```

```
    b.pop_front()
while !empty(a)
    merged.append(a[1])
    a.pop_front()
while !empty(b)
    merged.append(b[1])
    b.pop_front()
```

6)
```
for i = 1 to m / 2
  //Swap a[i] and a[m − i + 1]
  a[i], a[m - i + 1] = a[m - i + 1], a[i]
```

7)
```
for i = 1 to m
  if a[i] % 3 = 1
    for j = 1 to m
      b[i][j] = a[i]
```

8)
```
result = 1
for i = 1 to k
  result *= k
```

9)
```
a_in_power = a
result = 1
while k > 0
  if k % 2 == 1
    result *= a_in_power
  a_in_power *= a_in_power
  k = k / 2
```

10)
```
i_lower = 1
i_upper = n
while i_lower < i_upper
  i_middle = (i_lower + i_upper) / 2
  if a[i_middle] > goal
    i_upper = i_middle - 1
  elif a[i_med] < goal
    i_lower = i_middle + 1
  else
    break
```

2. Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна кожну пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = \log_k n;$$
$$f_2(n) = \log_3(n^{3n});$$
$$f_3(n) = n^k;$$

$$f_4(n) = k^n;$$
$$f_5(n) = \sqrt{n};$$
$$f_6(n) = \sin n;$$
$$f_7(n) = \sin^2 n;$$
$$f_8(n) = \sin^2 n + \cos^2 n;$$
$$f_9(n) = \text{n!};$$
$$f_{10}(n) = e^n.$$

3. Довести дані твердження, користуючись визначенням O-позначення:

$$7n^3 + n^2 = O(n^3);$$
$$n^{1024} = O(n!);$$
$$n^{14} = O\left(2^n\right);$$
$$n^3 + n^2 + n + 1 = O(n^4);$$
$$n + 0{,}75^n = O(n);$$
$$e^n = O(\pi^n);$$
$$e^n + n^{512} = O(e^n);$$
$$2\log n = O(n);$$
$$n\log n = O(n^2);$$
$$n^2 + n\log^4 n = O(n^2).$$

1.3. Алгоритми сортування

1.3.1. Сортування вставками. Короткі теоретичні відомості та приклади розв'язання задач

Сортування вставками (англ. *Insertion Sort*) зручне для коротких послідовностей, складність алгоритму $O(n^2)$, де n – кількість елементів для сортування. Даний алгоритм не потребує додаткового обсягу пам'яті, окрім однієї змінної.

Суть алгоритму: послідовність даних поділяється на відсортовану та невідсортовану частини (спочатку як відсортована частина приймається перший елемент); елементи з невідсортованої частини по черзі вибираються та вставляються у відсортовану частину таким чином, щоб не порушувати в ній впорядкованість елементів.

Задача 1

Вихідна послідовність подана такими елементами: $6, 3, 8, 1, 5, 7$. Необ-

хідно відсортувати елементи за зростанням, використовуючи алгоритм сортування вставками.

Розв'язання

Починаємо сортування з другого елемента, це число 3 (на рис. 1.1 елемент зображено у колі).

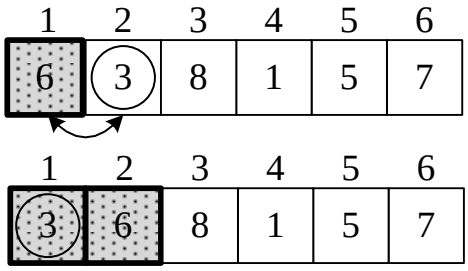


Рисунок 1.1 – Вставка елемента №2 у відсортовану послідовність

Порівнюємо його з першим елементом. Оскільки $3 < 6$, то другий елемент необхідно поміняти місцями з першим (показано стрілкою на рис. 1.1). Будемо показувати вже відсортовану частину послідовності на рисунках затемненою.

Наступним будемо розглядати елемент №3, (це число 8, рис. 1.2). Порівнюємо його з елементом №2, тобто 6. Оскільки елемент, який розглядається, більший, ніж попередній, то він залишається на своєму місці.

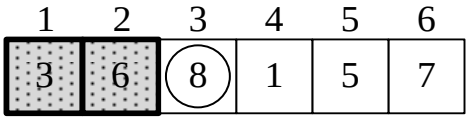


Рисунок 1.2 – Вставка елемента №3 у відсортовану послідовність

Розглянемо елемент №4 в послідовності (це число 1), порівняємо його з елементом №3 (це число 8), оскільки значення попереднього елемента більше, міняємо елементи місцями (показано стрілкою на рис. 1.3).

Далі порівнюємо число 1 з елементом №2 (це число 6), міняємо елементи місцями (див. рис. 1.3). Порівнюємо число 1 з елементом №1 (це число 3), міняємо елементи місцями (див. рис. 1.3).

Вставимо елемент №5 у відсортовану послідовність (це число 5). Порівнюємо його з елементом №4 (це число 8), міняємо елементи місцями (рис. 1.4).

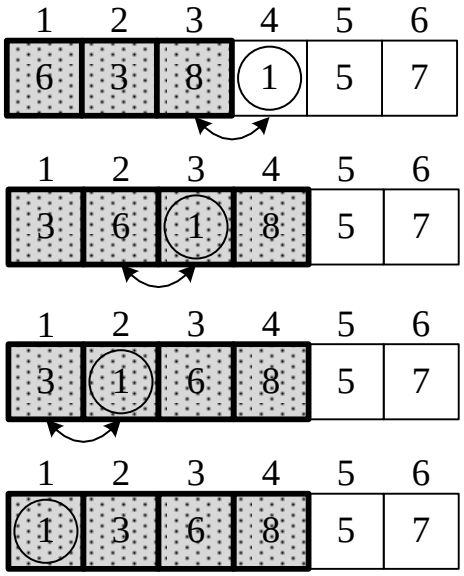


Рисунок 1.3 – Вставка елемента №4 у відсортовану послідовність

Продовжуємо порівняння: порівнюємо число 5 з елементом №3 (це число 6), міняємо елементи місцями, як показано стрілкою на відповідному рисунку та продовжуємо порівняння. Порівняємо 5 з елементом №2 (це число 3). Даний елемент менший за значенням від елемента, що вставляється, значить, закінчуємо порівняння, елемент №5 вставлено у відсортовану послідовність.

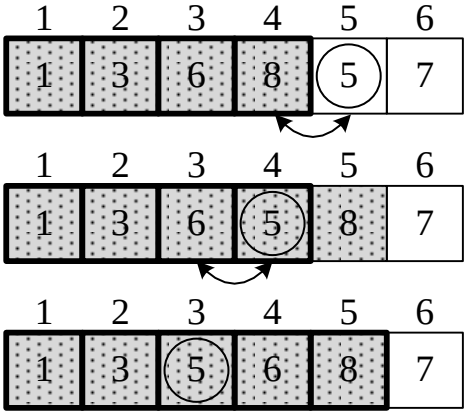


Рисунок 1.4 – Вставка елемента №5 у відсортовану послідовність

Вставка елемента №6 (його значення дорівнює 7) подана на рис. 1.5.

Порівнюємо елемент №6 з елементом №5 (це число 8), міняємо елементи місцями. Порівнюємо 7 з елементом №4 (це число 6).

Даний елемент менший за значенням від елемента, який необхідно вста-

вити, значить, закінчуємо порівняння, елемент №6 додано до відсортованої послідовності.

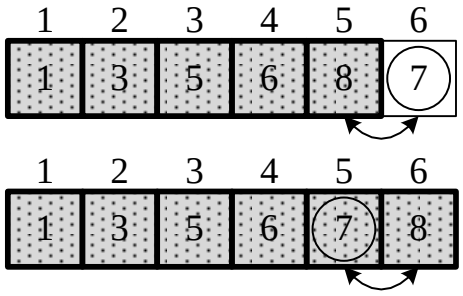


Рисунок 1.5 – Вставка елемента №6 у відсортовану послідовність

Усі елементи було розглянуто, отже, вся вихідна послідовність відсортована за зростанням, результат – $1, 3, 5, 6, 7, 8$.

1.3.2. Сортування бульбашкою. Короткі теоретичні відомості та приклади розв'язання задач

Алгоритм сортування бульбашкою (англ. *Bubble Sort*) є ефективним для невеликих послідовностей, складність алгоритму $O(n^2)$, де n – кількість елементів для сортування.

Суть алгоритму: обмін сусідніх елементів проходить таким чином, що відбувається «виштовхування» менших значень послідовності на її початок. Кількість проходів послідовністю залежить від початкового розташування елементів. Кожен прохід починається з крайнього лівого елемента, відбувається порівняння сусідніх елементів, і якщо порядок сусідніх елементів неправильний, то вони міняються місцями. Якщо здійснено повний прохід і перестановок не було, то послідовність відсортовано.

Задача 2

Вихідна послідовність подана такими елементами: $6, 2, 5, 3, 9$. Необхідно відсортувати елементи за зростанням, використовуючи сортування бульбашкою.

Розв'язання

Перший прохід послідовністю показано на рис. 1.6. Вихідна послідовність подана стовпцем чисел зліва, в якому темним шрифтом виділені елемен-

ти, які порівнюються на кожному кроці. Поряд показана змінена послідовність (якщо була перестановка елементів).

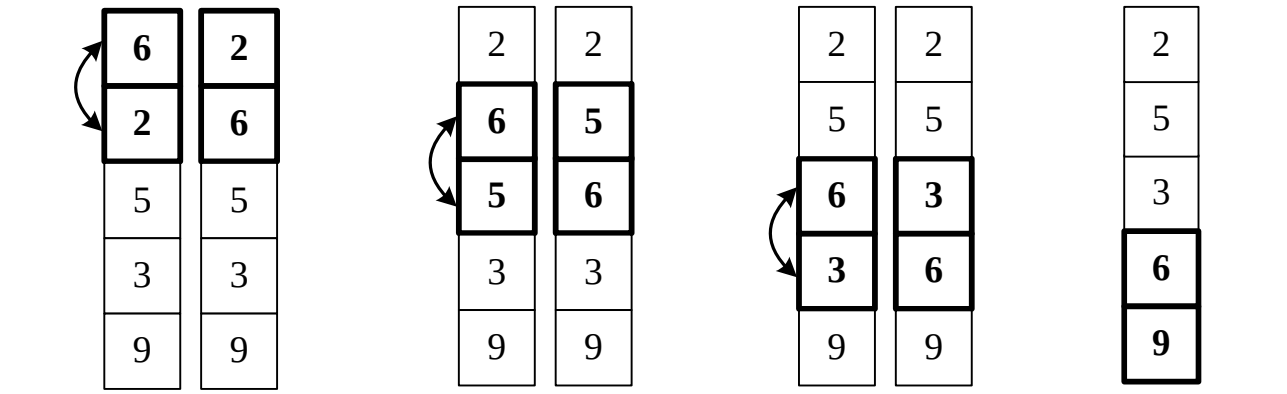


Рисунок 1.6 – Перший прохід послідовністю

Оскільки на першому проході були перестановки (6 з 2, 6 з 5, 6 з 3), то необхідно знову здійснити прохід.

Другий прохід послідовністю показаний на рис. 1.7.

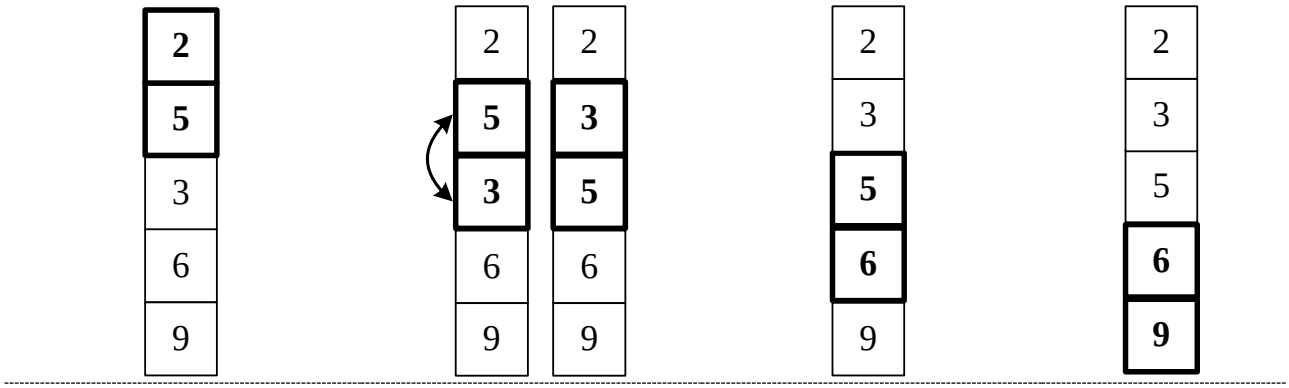


Рисунок 1.7 – Другий прохід послідовністю

Зважаючи на те, що на другому проході були перестановки, необхідний новий прохід. Третій прохід послідовністю подано на рис. 1.8.

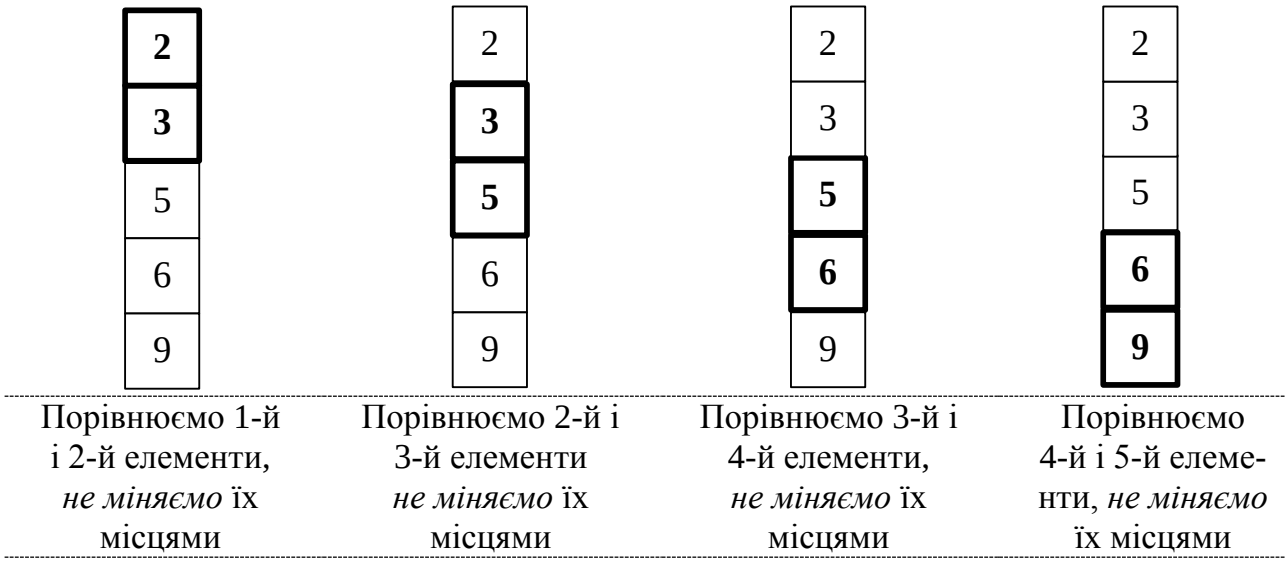


Рисунок 1.8 – Третій прохід послідовністю

На третьому проході послідовністю перестановок не було, значить, послідовність відсортовано за зростанням: $2, 3, 5, 6, 9$.

1.3.3. Сортування вибором. Короткі теоретичні відомості та приклади розв'язання задач

Сортування вибором (англ. *Selection sort*) – алгоритм сортування зі складністю $O(n^2)$, де n – кількість елементів для сортування.

Суть алгоритму: знаходимо мінімальний елемент, міняємо знайдений елемент з першим елементом невідсортованої послідовності (якщо елемент розташований в даній позиції – обмін не відбувається); виключивши із розгляду вже відсортовані елементи, продовжуємо процес, поки не дійдемо до останнього правого елемента.

Задача 3

Використовуючи сортування вибором, впорядкувати масив чисел $6, 3, 8, 1, 5, 7$ за зростанням.

Розв'язання

На *першому кроці* алгоритму переглядаємо весь масив і знаходимо міні-

мальний елемент. Міняємо цей елемент з першим зліва елементом. На рис. 1.9, *а* мінімальний елемент масиву – це число 1, міняємо його з елементом, рівним 6 (показано стрілкою на рис. 1.9, *а*).

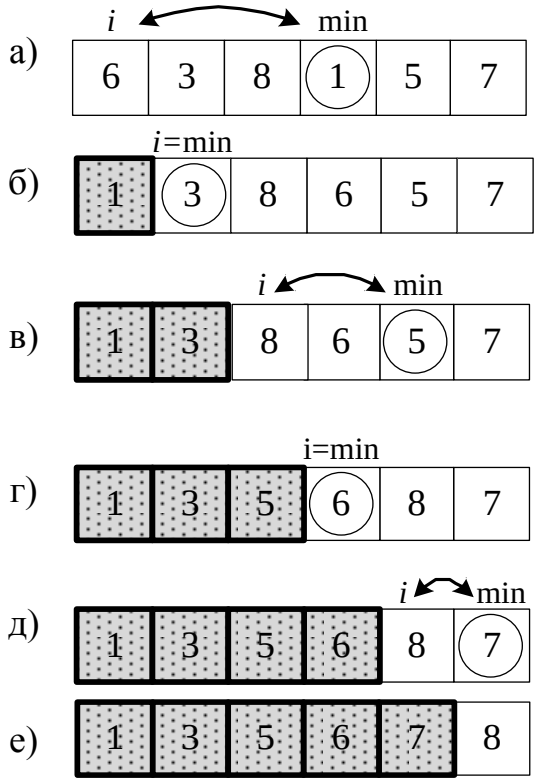


Рисунок 1.9 – Сортування вибором масиву з 8-ми елементів

Затемнені клітинки на рисунку показують вже відсортовану частину масиву. Номер елемента, куди необхідно помістити знайдений на даному кроці елемент, відмічено зверху над клітинкою індексом i. Знайдений мінімальний елемент відмічено зверху над клітинкою min.

На *другому кроці* переглядаємо масив, починаючи з другої клітинки. В даному випадку $i = \min$, тобто мінімальний елемент розташований на своєму місці (рис. 1.9, *б*). Виділяємо темним клітинку №2. Переходимо до наступного кроку.

Крок 3. Переглядаємо масив, починаючи з третьої клітинки. Знаходимо мінімальний елемент, це число 5. Міняємо 5 та 8 місцями (клітинки min та i на рис. 1.9, *в*). Виділяємо темним клітинку №3.

Крок 4. Переглядаємо масив, починаючи з четвертого елемента. В даному випадку мінімальний елемент (число 6) знаходиться у крайній зліва не-

відсортованій частині масиву (рис. 1.9, *г*). Виділяємо темним клітинку №4.

Крок 5. Переглядаємо масив, починаючи з п'ятого елемента. Мінімальний елемент – число 7 (рис. 1.9, *д*), міняємо його місцями з числом 8. Затемнюємо клітинку №5.

Оскільки залишається всього один елемент справа від відсортованої частини масиву, то алгоритм закінчується, масив відсортований за зростанням: 1, 3, 5, 6, 7, 8 (рис. 1.9, *е*).

1.3.4. Сортування злиттям. Короткі теоретичні відомості та приклади розв'язання задач

У сортуванні злиттям використовується принцип «розділяй та володарюй».

Суть алгоритму: задача розбивається на декілька підзадач меншого розміру, які розв'язуються за допомогою рекурсивного виклику чи безпосередньо (якщо мала розмірність), потім їх розв'язки комбінуються. Основні етапи можна подати таким чином:

– ділимо послідовність на елементи (по одному);

– порівнюємо попарно і сортуємо;

– об'єднуємо пари доти, поки вся послідовність не буде поєднана разом.

Задача 4

Вихідна послідовність подана такими елементами: 2, 4, 7, 3, 5, 9, 8, 6. Необхідно відсортувати елементи за зростанням, використовуючи сортування злиттям.

Розв'язання

Перший етап – поділ послідовності на елементи – показано на рис. 1.10.

Отже, на кінець першого етапу ми отримали елементи:

2 4 7 3 5 9 8 6.

Переходимо до наступного етапу. Порівняємо попарно отримані елементи, у нас таких пар 4 (рис. 1.11, I рівень).

Перша пара 2 і 4, оскільки ми сортуємо за зростанням, то просто «склеюємо» ці два елементи в одну послідовність. Друга пара 7 і 3, міняємо елементи місцями, третю пару 5 і 9 залишаємо незмінною, а елементи четвертої пари 8 і 6 міняємо місцями. Отримуємо пари 2 і 4, 3 і 7, 5 і 9, 6 і 8. На рис. 1.11 да-

ному злиттю відповідає перехід від I до II рівня.

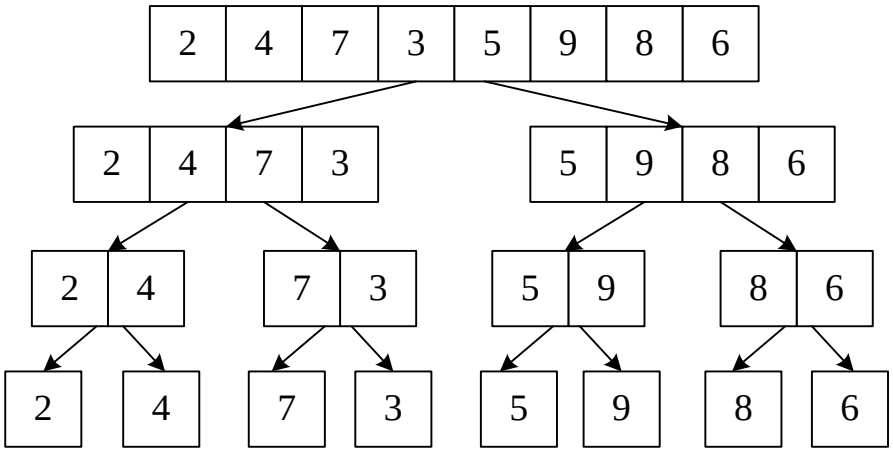


Рисунок 1.10 – Перший етап сортування злиттям

Продовжимо порівняння і об’єднання пар. Перші дві пари 2 і 4 та 3 і 7. Порівнюємо перші елементи цих пар – це 2 і 3, менший елемент, тобто 2, записуємо на перше місце загальної послідовності. Залишається 4 та 3 і 7. Порівнюємо перші елементи, тобто 4 і 3, менший елемент, в даному випадку 3, записуємо в загальну послідовність після 2. Залишається 4 і 7, порівнюємо і записуємо до загальної послідовності спочатку 4, потім 7. Отримали загальну послідовність 2, 3, 4, 7, яка утворилась від злиття послідовностей 2, 4 і 3, 7.

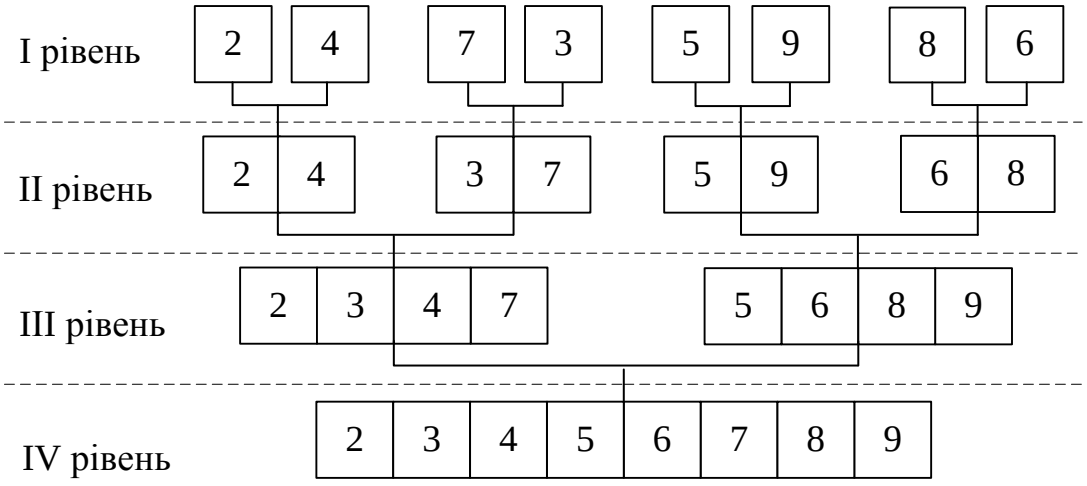


Рисунок 1.11 – Злиття послідовностей

Аналогічно порівнюємо і об’єднуємо пари 5, 9 і 6, 8.

Записуємо до загальної послідовності 5, далі 6, потім 8 та 9.

На рис. 1.11 даному злиттю відповідає перехід від II до III рівня.

Завершальний етап «злиття» послідовностей 2,3,4,7 та 5,6,8,9, поданий переходом від рівня III до рівня IV на рис. 1.11.

На IV рівні на рис. 1.11 подана відсортована за зростанням послідовність: $2, 3, 4, 5, 6, 7, 8, 9$.

1.3.5. Швидке сортування. Короткі теоретичні відомості та приклади розв'язання задач

Алгоритм швидкого сортування (англ. *QuickSort*) є покращеним варіантом сортування бульбашкою. Час роботи алгоритму у гіршому випадку дорівнює $O(n^2)$, на практиці математичне очікування часу його ти $O(n \log n)$, де n – кількість елементів для сортування.

Суть алгоритму: на основі вибраного елемента, що називається опорним, масив ділиться на дві частини: до першої частини потрапляють всі елементи не більші від опорного, а в другу – рівні опорному та більші. Для кожного із отриманих масивів (якщо їх розміри більші за одиницю) виконується рекурсивне розбиття на дві частини за таким самим принципом.

Як опорний елемент може виступати крайній лівий, випадковий елементи чи медіана значень першого, середнього та останнього елементів.

Задача 5

Використовуючи алгоритм швидкого сортування, відсортувати за зростанням елементи масиву $A[1 \ldots 8]$, поданого на рис. 1.12.

	1	2	3	4	5	6	7	8
A	6	4	3	7	5	2	4	8

Рисунок 1.12 – Елементи вихідного масиву A

Розв'язання

На початку виконання сортування визначимося з опорним елементом. Нехай для визначеності це буде крайній зліва. На рис. 1.13 подані всі етапи алгоритму швидкого сортування заданого масиву.

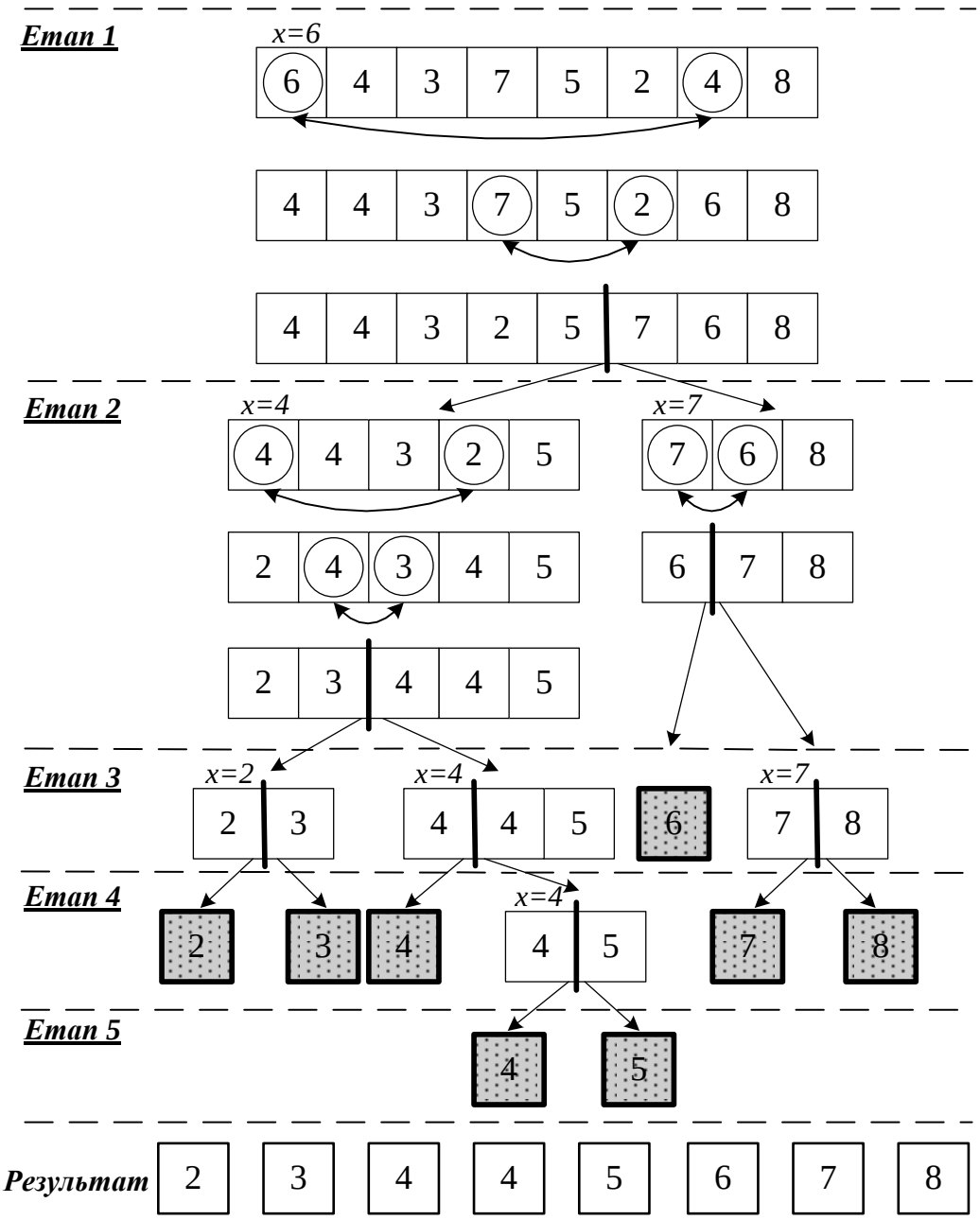


Рисунок 1.13 – Етапи алгоритму швидкого сортування масиву $A[1..8]$

На першому етапі необхідно розділити масив на дві частини так, щоб зліва розташовувались елементи не більші від опорного, а справа – не менші від нього, тобто не менші 6. Починаємо прохід з обох боків назустріч, порівнюючи елементи з опорним. Порівнюємо елементи $A[1] = 6$ і $A[8] = 8$, оскільки обидва елементи не менші від опорного, то з правого кінця масиву переміщаємося на одну клітинку вліво. Порівнюємо елементи $A[1] = 6$ і $A[7] = 4$, оскільки елемент справа менший від опорного, необхідно провести

обмін, міняємо елементи місцями, переміщаємося з лівого краю масиву на один елемент вправо.

Порівнюємо елементи $A[2] = 4$ і $A[6] = 2$, оскільки обидва елементи менші від опорного, переміщаємося з лівого краю масиву на один елемент вправо.

Порівнюємо елементи $A[3] = 3$ і $A[7] = 2$, оскільки обидва елементи менші від опорного, переміщаємося з лівого краю масиву на один елемент вправо.

Порівнюємо елементи A[4]=7 и A[7]=2, оскільки елемент справа менший від опорного, необхідно провести обмін, міняємо елементи місцями, переміщаємося з лівого краю масиву на один елемент вправо.

Всі елементи розглянуто, прохід завершено, масив поділено.

На рис. 1.13 (етап 1) показана кількість перестановок елементів, які довелось зробити. В результаті масив розділився на дві частини так, що в першій міститься 5 елементів, а в другій – 3 елементи.

Далі для кожного з отриманих масивів процедура була повторена. На рис. 1.13 (етап 2) показано опорний елемент для кожного з отриманих масивів. На другому етапі для масиву, розміщеного зліва, – це елемент 4, для масиву, розміщеного справа, – це елемент 7. Кожний з даних масивів був розділений на два. На етапі 3 проведено подальше розбиття масивів.

У результаті отримано відсортований масив: $2, 3, 4, 4, 5, 6, 7, 8$.

1.3.6. Сортування підрахунком. Короткі теоретичні відомості та приклади розв'язання задач

Сортування підрахунком (англ. *Counting sort*) можна використовувати, якщо кожен з елементів послідовності, що сортується, – ціле невід'ємне число у відомому діапазоні, що не перевищує наперед відомого k.

Введемо позначення: A – масив, в якому зберігається вихідна послідовність; B – результуючий масив, в якому зберігається вже відсортована послідовність; C – допоміжний масив з індексами від 0 до k.

Суть алгоритму:

1) підрахуємо кількість елементів з однаковими значеннями в масиві A, запишемо у допоміжний масив C у відповідну клітинку отримане число (наприклад, число 5 зустрілось 3 рази в масиві A, тоді запишемо $C[5] = 3$);

2) перетворимо елементи масиву C, починаючи з останнього, за таким правилом: нове значення елемента дорівнює різниці між новим значенням сусіднього правого елемента і старого значення елемента, що розглядається (для елемента $C[k]$ замість нового значення сусіднього правого елемента береться число n);

3) проходимо послідовно вихідний масив A, починаючи з правого крайнього елемента, запишемо елемент, що розглядається (нехай, наприклад, там записано число m), в клітинку масиву B, номер якої збігається з числом, записаним у клітинці $C[m]$, зменшуємо значення $C[m]$ на одиницю.

Задача 6

Використовуючи сортування підрахунком, відсортувати за зростанням елементи масиву A, поданого на рис. 1.14.

	1	2	3	4	5	6	7	8
A	2	0	3	1	6	5	1	2

Рисунок 1.14 – Елементи вихідного масиву A

Розв'язання

Сформуємо допоміжний масив C (його розмір – від 0 до номера, рівного максимальному елементу в масиві A, тобто до 6). В клітинку $C[0]$ запишемо кількість нулів у масиві A, в $C[1]$ – кількість одиниць і т.д. (рис. 1.15).

	0	1	2	3	4	5	6
C	1	2	2	1	0	1	1

Рисунок 1.15 – Елементи допоміжного масиву C

Перетворимо масив C, записавши в кожній його клітинці максимальний номер (справа), де може розташовуватися число, що відповідає номеру клітинки. Почнемо перетворювати масив з останньої клітинки. Так, наприклад, число 6 може бути записане в останній клітинці (тобто №8) результуючого масиву, число 5 – в клітинці №7 (від номера клітинки розташування попереднього

числа 6 (№8) потрібно відняти кількість цих чисел у масиві – тобто 1, отримаємо клітинку №7).

Нижче на рис. 1.16 поданий перетворений масив C.

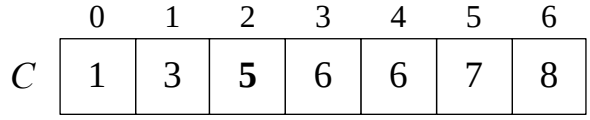


Рисунок 1.16 – Елементи перетвореного масиву C

Починаємо заповнювати результуючий масив B.

Ітерація 1

Знаходимо значення елемента №8 у масиві A – це число 2, знаходимо значення елемента №2 в масиві C – це 5, тоді записуємо в результуючий масив $B[5] = 2$, зменшуючи в масиві C значення елемента №2 на одиницю, тобто від 5 віднімаємо 1, отже, $C[2] = 4$ (рис. 1.17).

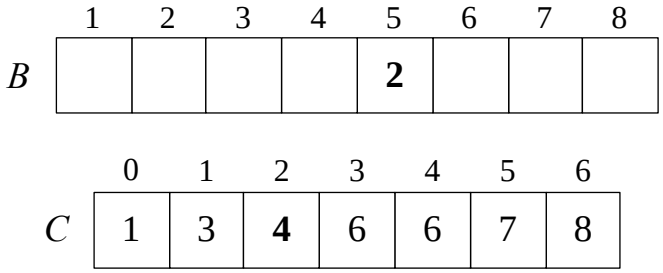


Рисунок 1.17 – Вигляд масивів B та C після першої ітерації

Ітерація 2

Знаходимо значення елемента №7 у масиві A – це значення 1, знаходимо значення елемента №1 у масиві C – це 3, тоді записуємо $B[3] = 1$, зменшуючи в масиві C значення елемента №1 на одиницю, тобто від 3 віднімаємо 1, отже $C[1] = 2$ (рис. 1.18).

Ітерація 3

Знаходимо значення елемента №6 в масиві A – це число 5, знаходимо значення елемента №5 в масиві C – це 7, тоді записуємо $B[7] = 5$, зменшуючи в масиві C значення елемента №5 на одиницю, тобто від 7 віднімаємо 1, тобто $C[5] = 6$ (рис. 1.19).

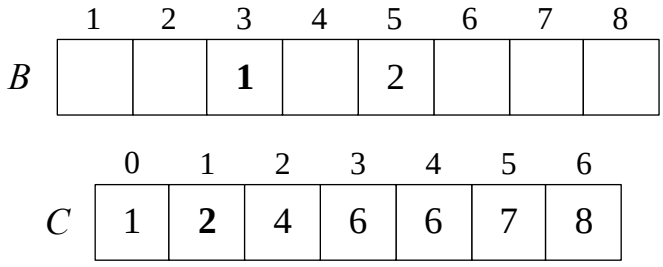


Рисунок 1.18 – Вигляд масивів B та C після другої ітерації

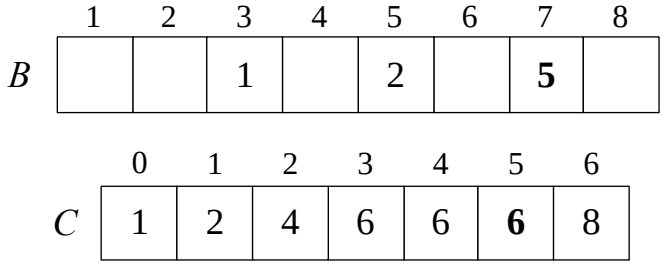


Рисунок 1.19 – Вигляд масивів B та C після третьої ітерації

Ітерація 4

Знаходимо значення елемента №5 в масиві A – це число 6, знаходимо значення елемента №6 в масиві C – це 8, тоді записуємо $B[8] = 6$, зменшуючи в масиві C значення елемента №6 на одиницю, тобто від 8 віднімаємо 1, отже, $C[6] = 7$ (рис. 1.20).

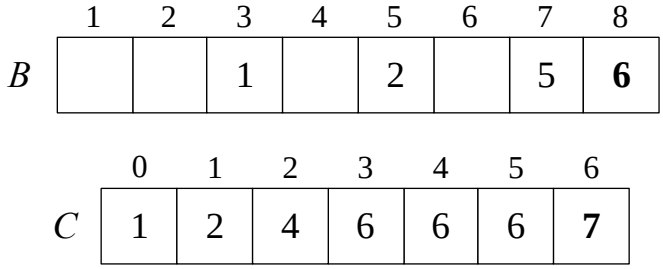


Рисунок 1.20 – Вигляд масивів B та C після четвертої ітерації

Ітерація 5

Знаходимо значення елемента №4 в масиві A – це число 1, знаходимо значення елемента №1 в масиві C – це 2, тоді записуємо $B[2] = 1$, зменшуючи в масиві C значення елемента №1 на одиницю, тобто від 2 віднімаємо 1, отже, $C[1] = 1$ (рис. 1.21).

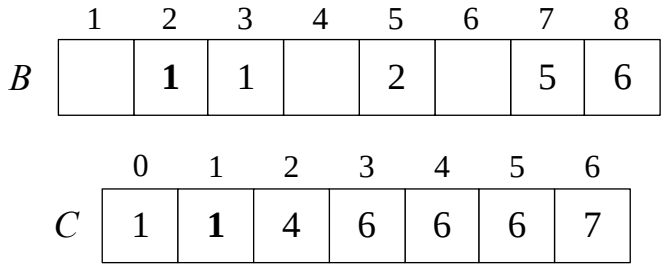


Рисунок 1.21 – Вигляд масивів B та C після п’ятої ітерації

Ітерація 6

Знаходимо значення елемента №3 в масиві A – це число 3, знаходимо значення елемента №3 в масиві C – це 6, тоді записуємо $B[6] = 3$, зменшуючи в масиві C значення елемента №3 на одиницю, тобто від 6 віднімаємо 1, отже, $C[3] = 5$ (рис. 1.22).

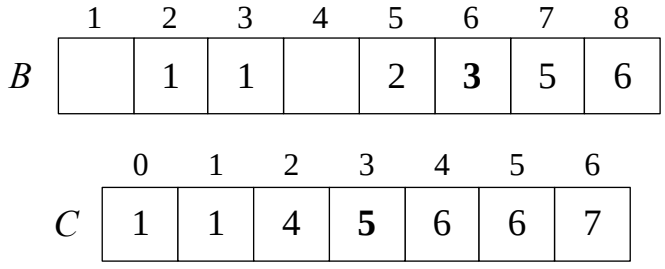


Рисунок 1.22 – Вигляд масивів B та C після шостої ітерації

Ітерація 7

Знаходимо значення елемента №2 в масиві A – це число 0, знаходимо значення елемента №0 в масиві C – це 1, тоді записуємо $B[1] = 0$, зменшуючи в масиві C значення елемента №0 на одиницю, тобто від 1 віднімаємо 1, отже, $C[0] = 0$ (рис. 1.23).

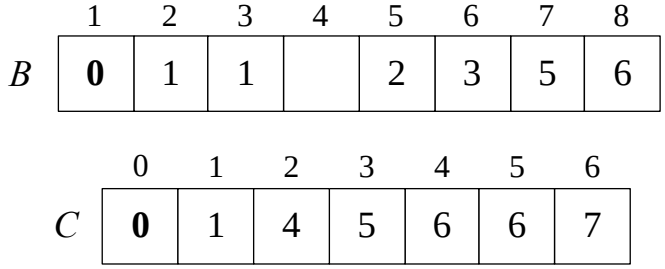


Рисунок 1.23 – Вигляд масивів B та C після сьомої ітерації

Ітерація 8

Знаходимо значення елемента №1 в масиві A – це число 2, знаходимо значення елемента №2 в масиві C – це 4, тоді записуємо $B[4] = 2$, зменшуючи в масиві C значення елемента №2 на одиницю, тобто від 4 віднімаємо 1, отже, $C[2] = 3$ (рис. 1.24).

	1	2	3	4	5	6	7	8
B	0	1	1	**2**	2	3	5	6

	0	1	2	3	4	5	6
C	0	1	**3**	5	6	6	7

Рисунок 1.24 – Вигляд масивів B та C після восьмої ітерації

Результуючий масив B повністю заповнений, елементи масиву A відсортовані за зростанням. Результат сортування: $0, 1, 1, 2, 2, 3, 5, 6$.

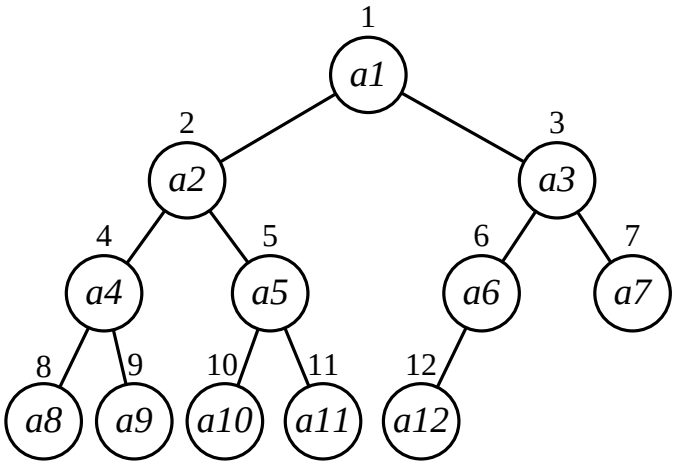
1.3.7. Сортування купою. Короткі теоретичні відомості та приклади розв'язання задач

Сортування купою (англ. *Heapsort*) – алгоритм сортування, в основі якого лежить спеціальний тип бінарного дерева – бінарне сортувальне дерево. Сортувальне дерево – це дерево, у якого значення предка в будь-якому піддереві не менше, ніж значення кожного з його потомків (якщо ж виконується сортування за спаданням елементів – то, відповідно, будь-який предок не більший, ніж кожний з його потомків).

Це дерево називають також двійковою купою чи пірамідою, а сам алгоритм називають також пірамідальним сортуванням.

На рис. 1.25 масив з 12 чисел подано у вигляді двійкового дерева.

Сортування починається з побудови двійкового дерева. Спочатку елементи масиву записуються послідовно у відповідні вершини (тобто номер вершини відповідає номеру елемента в масиві).



a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12
1	2	3	4	5	6	7	8	9	10	11	12

Рисунок 1.25 – Масив і його подання у вигляді двійкового дерева

Суть алгоритму:

1) для невідсортованої частини масиву (спочатку це весь масив) будуємо сортувальне дерево, в результаті чого максимум «піднімається» на вершину купи (в початок масиву), для цього рухаємося від потомків уверх до предків і перевіряємо властивість купи, якщо потомок більший від предка, то міняємо їх місцями (якщо такий обмін відбувся, то предка, який опустився на один рівень, необхідно порівняти з потомками, що знаходяться нижче);

2) міняємо місцями максимум та останній елемент невідсортованої частини масиву, виключаємо останній елемент з невідсортованої частини;

3) оскільки в результаті обміну елементів могла порушитися властивість купи, для невідсортованої частини масиву, починаючи з першої вершини, здійснюється перевірка даної властивості і процедура перебудови в сортувальному дереві повторюється, поки всі елементи не будуть розглянуті (поки в сортувальному дереві не залишиться один елемент).

Задача 7

Відсортувати послідовність з 10-ти елементів $1, 3, 5, 2, 4, 6, 7, 2, 3, 8$ в порядку зростання, використовуючи сортування купою.

Розв'язання

Запишемо вихідну послідовність у масив A, який складається з 10-ти

елементів, як наведено на рис. 1.26.

№ елемента	1	2	3	4	5	6	7	8	9	10
A	1	3	5	2	4	6	7	2	3	8

Рисунок 1.26 – Вихідна послідовність чисел у масиві A

Сформуємо дерево, як показано на рис. 1.27. Номери вершин підписані зверху, вони відповідають номеру елемента в масиві A.

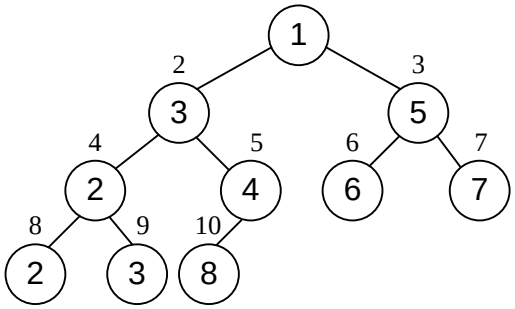


Рисунок 1.27 – Вихідне дерево

Ітерація 1

Починаємо «обгортання» з вершини, яка має максимальний номер з усіх вершин, що мають потомків. Це номер $\frac{n}{2}$, де n – кількість елементів вихідної послідовності, в даному випадку це вершина № 5.

Перевіримо властивість купи для даної вершини, тобто значення елемента, розташованого у вершині-предкові, має бути не меншим, ніж значення будь-якого з елементів, розташованих в її вершинах-потомках.

Вершина № 5 має тільки одного потомка – вершину №10. Оскільки значення елемента в 5-й вершині менше за значення елемента у вершині-потомкові, то міняємо елементи місцями (рис. 1.28).

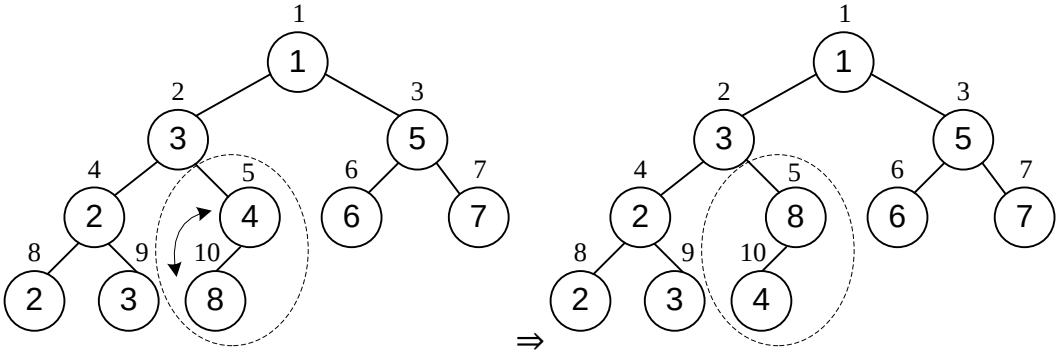


Рисунок 1.28 – Перевірка властивості купи для вершин №5 та №10

Перевіримо властивість купи для вершини №4 та її потомків – вершин №8 та №9, окільки властивість купи не виконується, то елементи у вершинах №4 та №9 міняємо місцями (рис. 1.29).

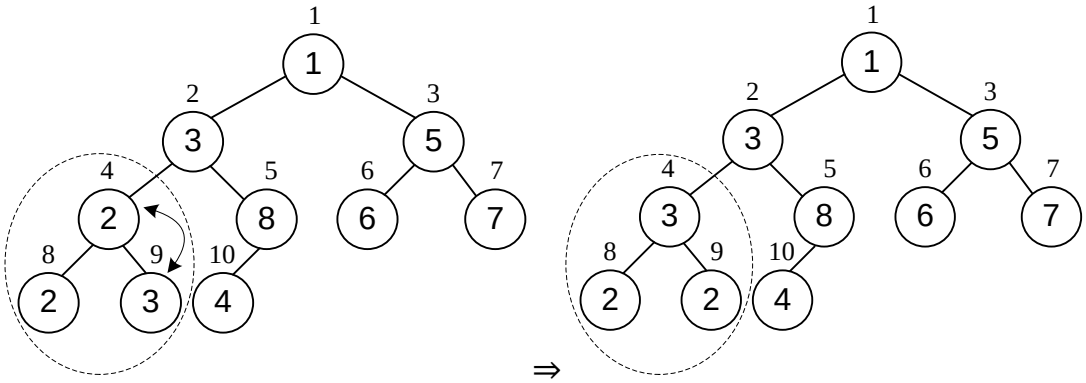


Рисунок 1.29 – Перевірка властивості купи для вершин №4, №8 та №9

Розглянемо вершини №3, №6 та №7, перевіряємо для них властивість купи. Оскільки властивість не виконується, то міняємо місцями елементи у вершинах №3 та №7 (рис. 1.30).

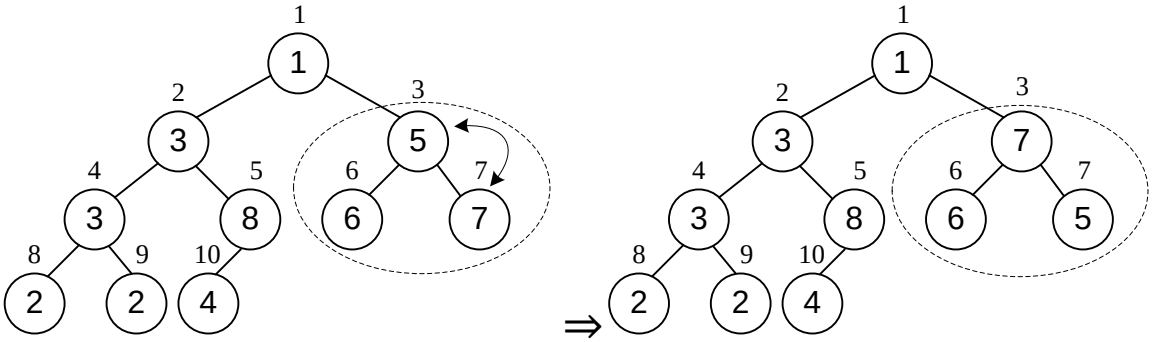


Рисунок 1.30 – Перевірка властивості купи для вершин №3, №6 та №7

Перевіряємо властивість купи для вершин №2, №4 і №5 (рис. 1.31). Властивість не виконується, тому міняємо місцями елементи у вершинах №2 та №5.

Оскільки елемент у вершині №5 був змінений, то перевіримо властивість купи для вершин №5 та №10. Оскільки властивість не виконується, то міняємо елементи в даних вершинах місцями (рис. 1.32).

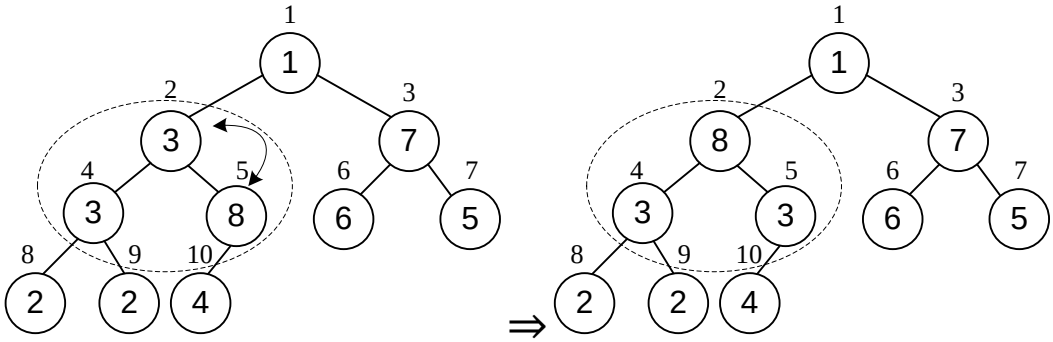


Рисунок 1.31 – Перевірка властивості купи для вершин №2, №4 та №5

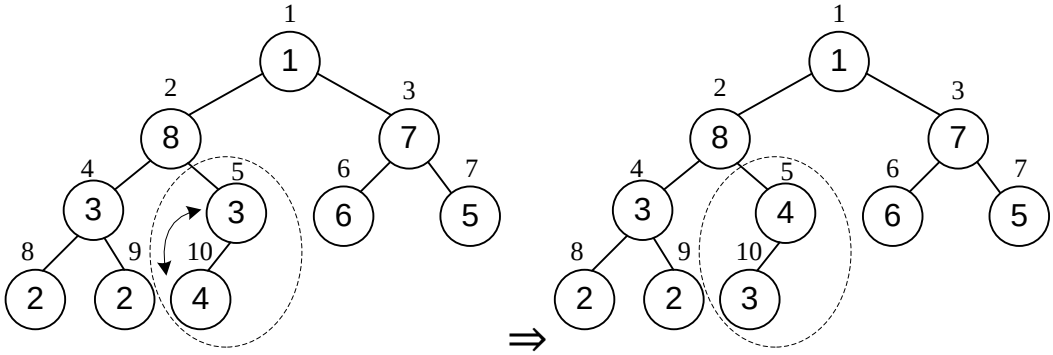


Рисунок 1.32 – Перевірка властивості купи для вершин №5 та №10

Порівняємо значення елементів у вершинах №1, №2 і №3 (рис. 1.33). Міняємо елементи зі значеннями 1 та 8 місцями (вершини №1 і №2).

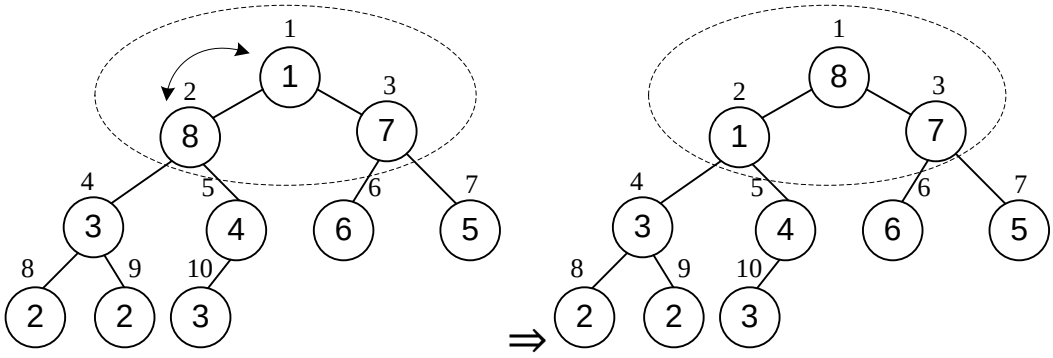


Рисунок 1.33 – Перевірка властивості купи для вершин №1, №2 та №3

Оскільки значення елемента у вершині №2 змінилося, то перевіримо властивість купи для вершин №2, №4 та №5 (рис. 1.34). З огляду на те, що

властивість купи не виконується, то міняємо місцями елементи у вершинах №2 та №5.

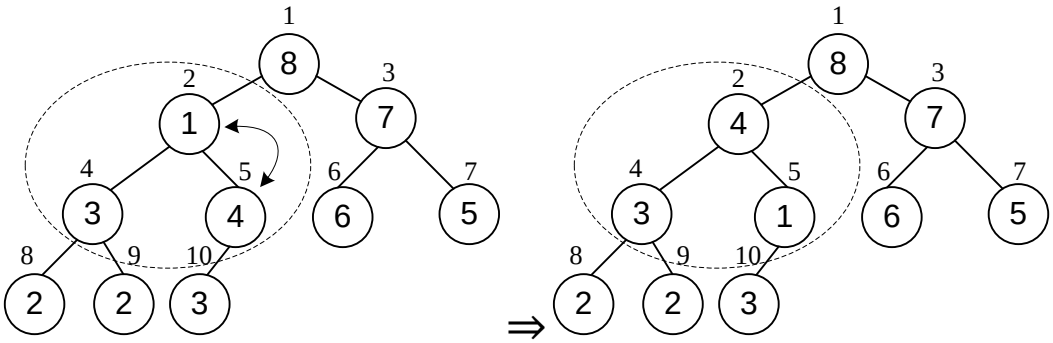


Рисунок 1.34 – Перевірка властивості купи для вершин №2, №4 та №5

Розглянемо вершину №5 та її потомків, властивість купи не виконується, отже, міняємо елементи вершин №5 і №10 місцями (рис. 1.35).

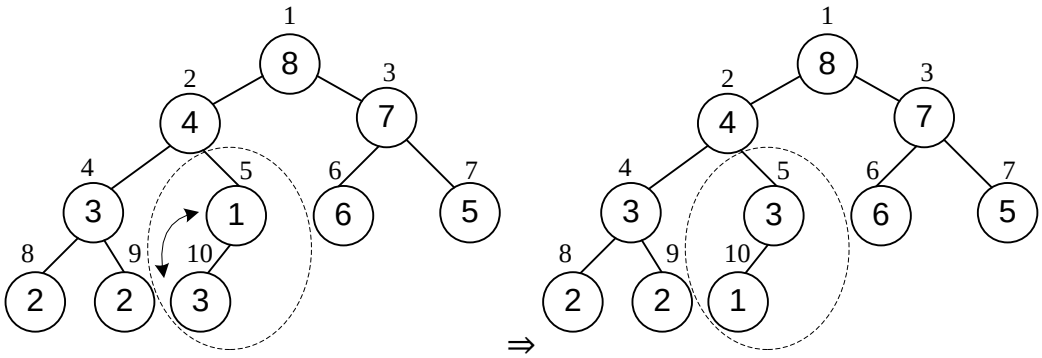


Рисунок 1.35 – Перевірка властивості купи для вершин №5 та №10

Для вершини №4 властивість купи не порушується. Таким чином, властивість купи виконується для всіх вершин дерева. Максимальний елемент масиву розташований в кореневій вершині дерева.

Оскільки ми сортуємо послідовність чисел за зростанням, то можна розмістити цей елемент останнім у масиві *A*. Поміняємо елемент, розташований у вершині №1, з елементом у вершині №10. Вигляд дерева після 1-ї ітерації та масив *A* наведені відповідно на рис. 1.36 та 1.37.

На наступних ітераціях цей, знайдений останнім, елемент не розглядається, зобразимо його затемненим і на дереві, і в масиві.

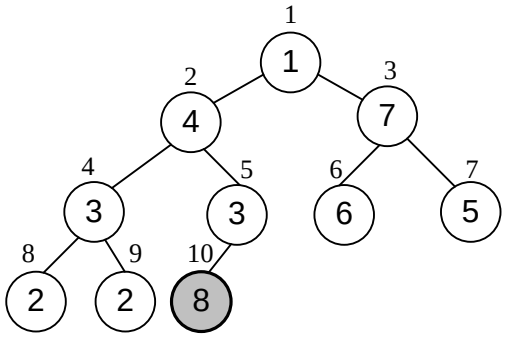


Рисунок 1.36 – Вигляд дерева після 1-ї ітерації

№ елемента	1	2	3	4	5	6	7	8	9	10
A	1	4	7	3	3	6	5	2	2	8

Рисунок 1.37 – Вигляд масиву *A* після 1-ї ітерації

Ітерація 2

Починаємо перевіряти властивість купи з вершини №1, оскільки, міняючи елементи у вершинах №1 і №10 на останньому кроці попередньої ітерації, ми могли порушити властивість купи для вершини №1 і її потомків. Оскільки властивість купи не виконується, то міняємо елемент, що дорівнює 1, з максимальним, тобто 7, що розташований у вершині №3. Даний процес показаний на рис. 1.38.

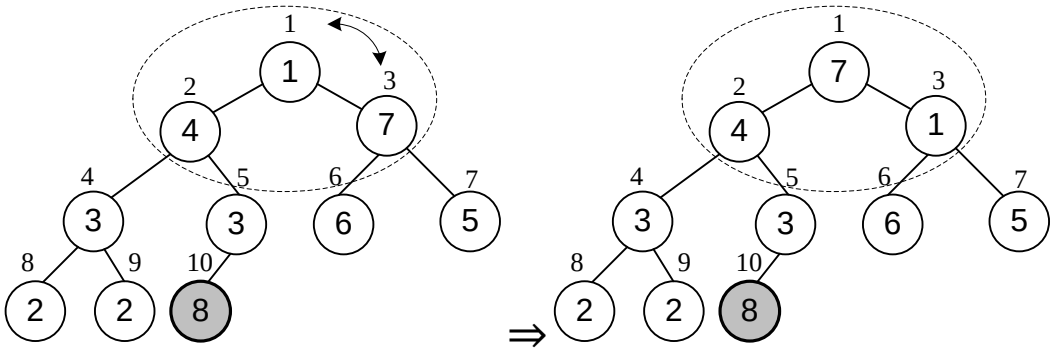


Рисунок 1.38 – Перевірка властивості купи для вершин №5 та №10

Оскільки ми замінили елемент у вершині №3, то перевіримо, чи не порушується для неї та її потомків властивість купи (рис. 1.39). Порівняємо елементи $1, 6, 5$ у відповідних вершинах. Оскільки властивість не виконується, то

міняємо місцями елементи 1 і 6 (вершини №3 і №6). Оскільки вершина №6 не має потомків, то властивість купи виконується для всіх вершин.

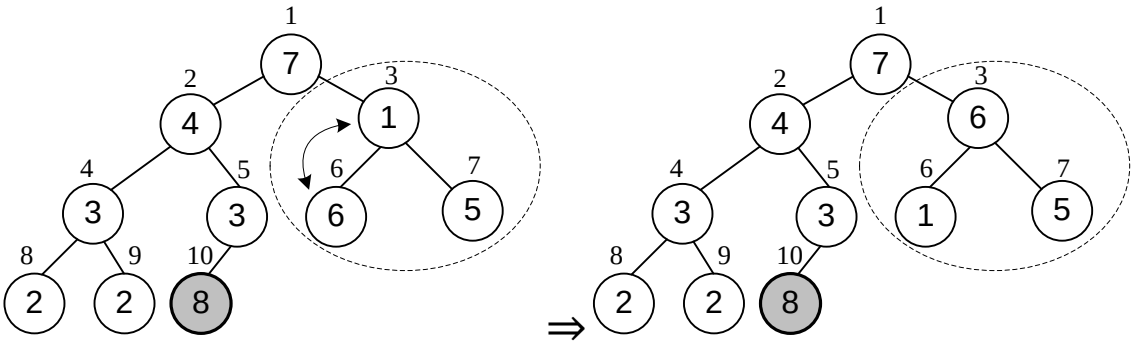


Рисунок 1.39 – Перевірка властивості купи для вершин №3, №6 та №7

Міняємо елемент, що розташований у вершині №1, з елементом у вершині №9, оскільки вершину №10 ми виключили з розгляду на першій ітерації, розташувавши в ній максимальний елемент із значенням 8. Результат наведений на рис. 1.40.

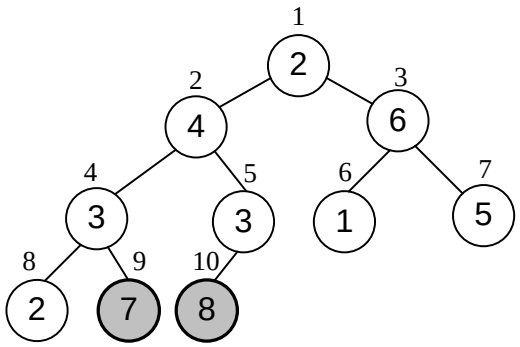


Рисунок 1.40 – Вигляд дерева після 2-ї ітерації

У масиві *A* знайдені два останніх елементи зображені затемненими, оскільки вони виключаються з розгляду на наступних ітераціях (рис. 1.41). Виключаємо з подальшого розгляду вершини №9 і №10 (на рис. 1.41 вони зображені затемненими).

№ елемента	1	2	3	4	5	6	7	8	9	10
A	2	4	6	3	3	1	5	2	7	8

Рисунок 1.41 – Масив *A* після 2-ї ітерації

Ітерація 3

Починаємо перевіряти властивість купи з вершини №1. Порівнюємо значення в даній вершині зі значеннями її потомків – 2 з 4 і 6. Оскільки властивість купи не виконується, то міняємо елемент у вершині №1 з максимальним елементом, тобто 6 (рис. 1.42).

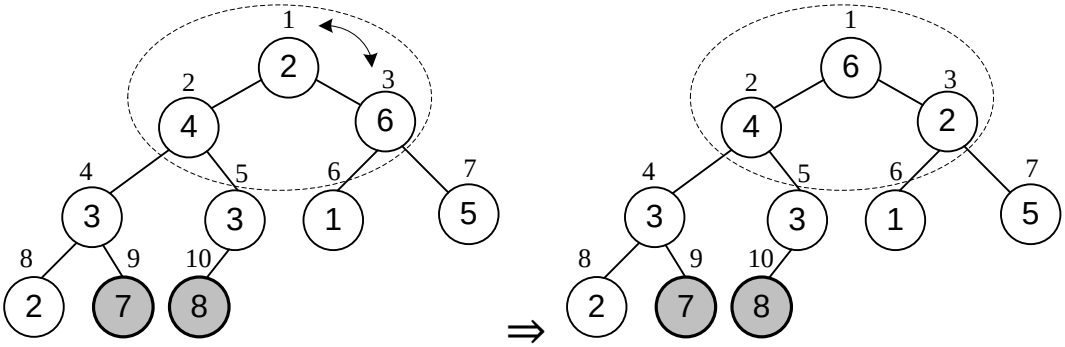


Рисунок 1.42 – Перевірка властивості купи для вершин №1, №2 та №3

Оскільки змінився елемент у вершині №3, то перевіримо для неї властивість купи. Порівнюємо 2 з 1 і 5. Оскільки елемент, розташований у вершині-предкові, менший від 5, то міняємо місцями елементи 2 і 5, як показано на рис. 1.43.

Елемент у вершині №7 змінився, але дана вершина не має потомків, отже, властивість купи виконується для всіх вершин дерева.

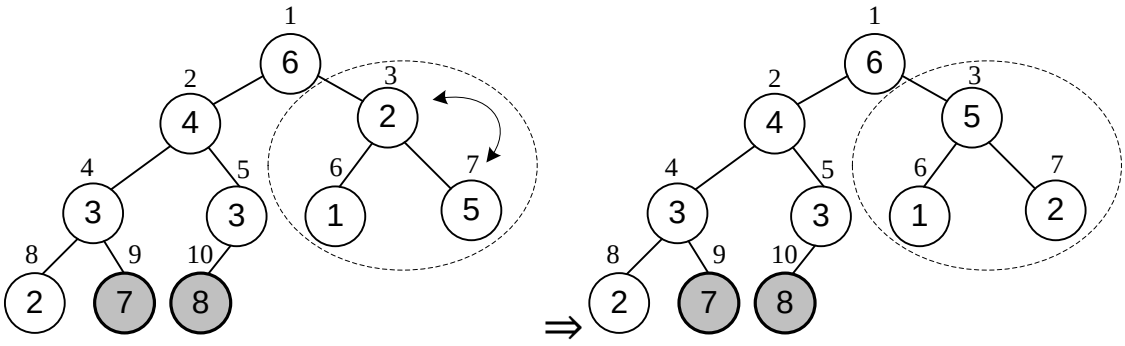


Рисунок 1.43 – Перевірка властивості купи для вершин №3, №6 та №7

У вершині №1 знаходиться максимальний з усіх елементів, що розглядаються на даній ітерації. Міняємо місцями елементи у вершинах №8 і №1. Виключаємо з подальшого розгляду вершини №8, №9 і №10 (рис. 1.44).

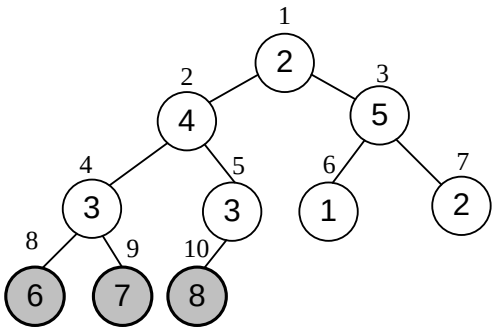


Рисунок 1.44 – Вигляд дерева після 3-ї ітерації

На рис. 1.45 поданий масив *A* після 3-ї ітерації.

№ елемента	1	2	3	4	5	6	7	8	9	10
A	2	4	5	3	3	1	2	6	7	8

Рисунок 1.45 – Масив *A* після 3-ї ітерації

Ітерація 4

Починаємо перевіряти властивість купи з вершини №1. Оскільки властивість купи не виконується, то міняємо місцями елементи, що розташовані у вершинах №1 і №3 (рис. 1.46).

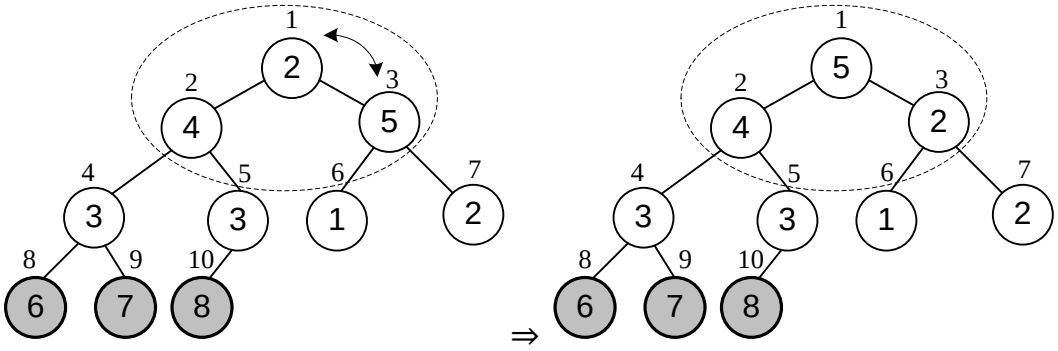


Рисунок 1.46 – Перевірка властивості купи для вершин №1, №2 та №3

Оскільки змінився елемент у вершині №3, то перевіримо для неї властивість купи (рис. 1.47). Властивість купи для даних вершин не порушується, отже, властивість виконується для всіх вершин дерева.

У вершині №1 знаходиться максимальний елемент з усіх, що розглядаються на даній ітерації. Розташовуємо його у вершину з максимальним номером із усіх, що розглядаються (тобто у вершину №7), а елемент з вершини №7

розмістимо у вершину №1 (рис. 1.48). Виключаємо з подальшого розгляду вершину №7.

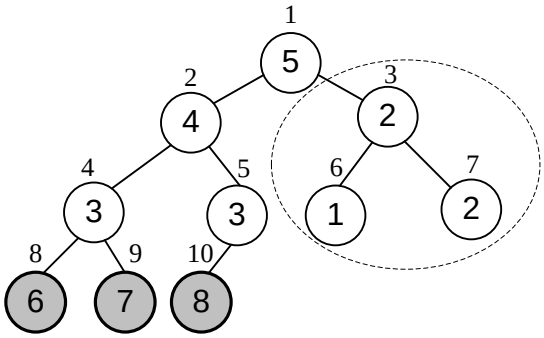


Рисунок 1.47 – Перевірка властивості купи для вершин №3, №6 та №7

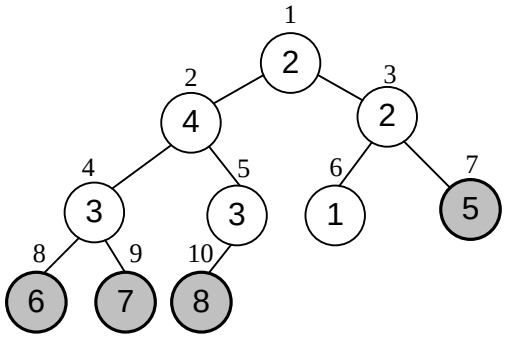


Рисунок 1.48 – Вигляд дерева після 4-ї ітерації

На рисунку 1.49 поданий масив *A* після 4-ї ітерації.

№ елемента	1	2	3	4	5	6	7	8	9	10
A	2	4	2	3	3	1	5	6	7	8

Рисунок 1.49 – Вигляд масиву *A* після 4-ї ітерації

Ітерація 5

Починаємо перевіряти властивість купи з вершини №1. Оскільки властивість купи не виконується, то міняємо місцями елементи, розташовані у вершинах №1 і №2 (рис. 1.50).

Оскільки змінився елемент у вершині №2, то перевіримо для неї властивість купи (рис. 1.51). Властивість купи не виконується, тому міняємо місцями елементи 3 і 2 (вершини №2 і №5).

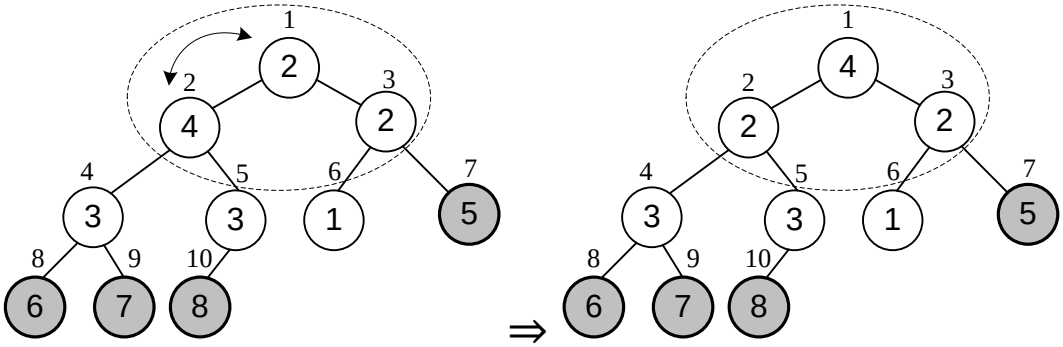


Рисунок 1.50 – Перевірка властивості купи для вершин №1, №2 та №3

Оскільки елемент у вершині №5 змінився, то розглянемо дану вершину. Вона має одного потомка, проте він був виключений із переліку вершин, що розглядаються на даній ітерації, тому закінчуємо перевірку виконання властивості купи.

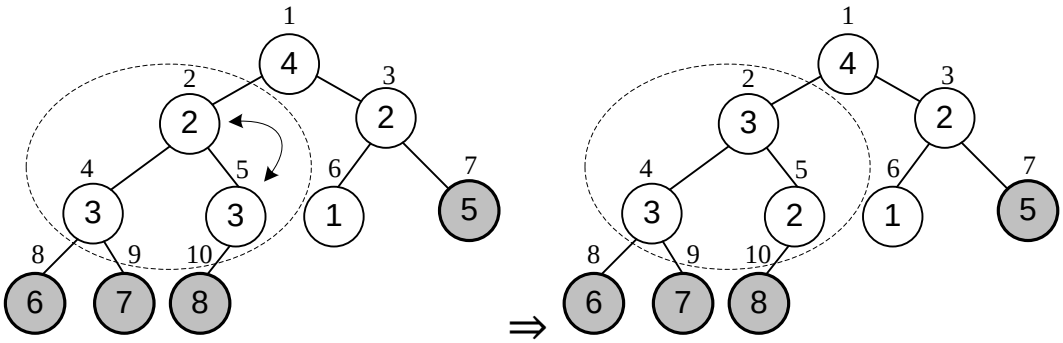


Рисунок 1.51 – Перевірка властивості купи для вершин №2, №4 та №5

У вершині №1 розташований максимальний для даної ітерації елемент. Міняємо його з елементом, розташованим у вершині №6 (рис. 1.52). Виключаємо з подальшого розгляду вершину №6.

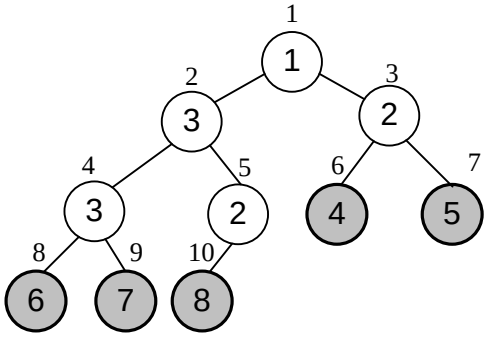


Рисунок 1.52 – Вигляд дерева після 5-ї ітерації

У масиві *A* вже відсортованими є 5 елементів (рис. 1.53).

№ елемента	1	2	3	4	5	6	7	8	9	10
A	1	3	2	3	2	4	5	6	7	8

Рисунок 1.53 – Масив *A* після 5-ї ітерації

Ітерація 6

На даній ітерації розглядаємо перші п’ять вершин. Знайдений максимальний елемент буде розміщений у вершині №5 і, відповідно, в п’яту позицію масиву *A*.

Перевіряємо властивість купи для вершини №1 та її потомків. Властивість купи не виконується, міняємо місцями елементи, розташовані у вершинах №1 і №2 (рис. 1.54).

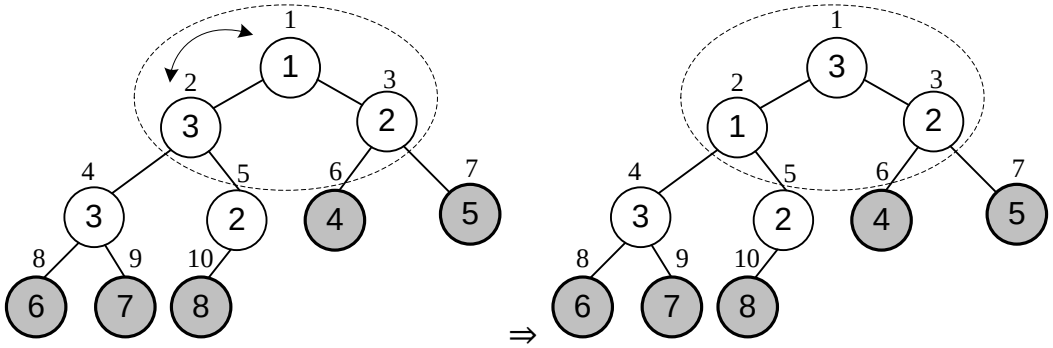


Рисунок 1.54 – Перевірка властивості купи для вершин №1, №2 та №3 (6-та ітерація)

Оскільки змінилось значення елемента у вершині №2, то перевіримо для неї властивість купи. Властивість купи порушується, тому міняємо місцями елементи 1 і 3 у вершинах №2 і №4 (рис. 1.55).

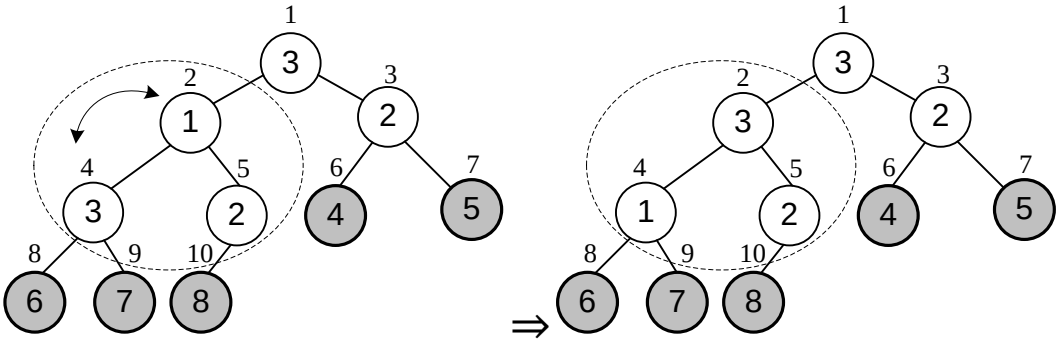


Рисунок 1.55 – Перевірка властивості купи для вершин №2, №4 та №5

Вершина №4, значення елемента якої змінилось, має потомків, які були виключені з розгляду, значить, властивість купи на даній ітерації виконується для всіх вершин дерева.

У вершині №1 розташований максимальний елемент, міняємо його з елементом у вершині №5 (рис. 1.56).

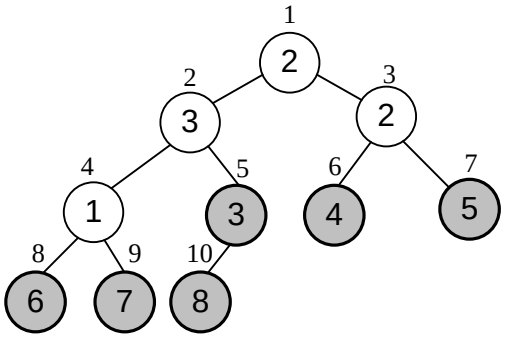


Рисунок 1.56 – Вигляд дерева після 6-ї ітерації

Масив *A* поданий на рис. 1.57. Вершину №5 виключаємо з подальшого розгляду.

№ елемента	1	2	3	4	5	6	7	8	9	10
A	2	3	2	1	3	4	5	6	7	8

Рисунок 1.57 – Вигляд масиву *A* після 6-ї ітерації

Ітерація 7

На даній ітерації розглядаємо перші чотири вершини. Перевіряємо властивість купи для вершини №1 та її потомків. Властивість купи порушена, міняємо місцями елементи 2 і 3 у вершинах №1 і №2 (рис. 1.58).

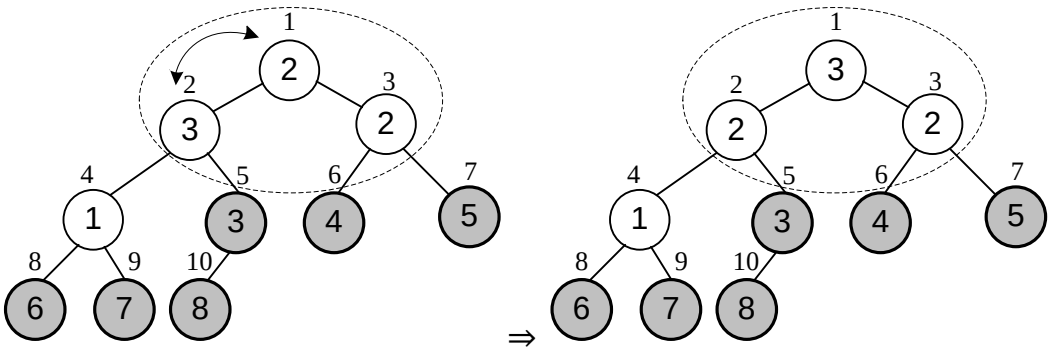


Рисунок 1.58 – Перевірка властивості купи для вершин №1, №2 та №3 (7-ма ітерація)

Оскільки елемент у вершині №2 змінився, то перевіримо властивість купи для неї та її потомків (рис. 1.59). Властивість купи не порушується, закінчуємо перевірку на дереві.

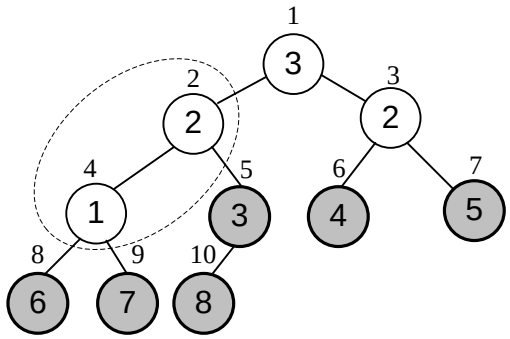


Рисунок 1.59 – Перевірка властивості купи для вершин №2 та №4

У вершині №1 розташований максимальний елемент, міняємо його з елементом у вершині №4 (рис. 1.60).

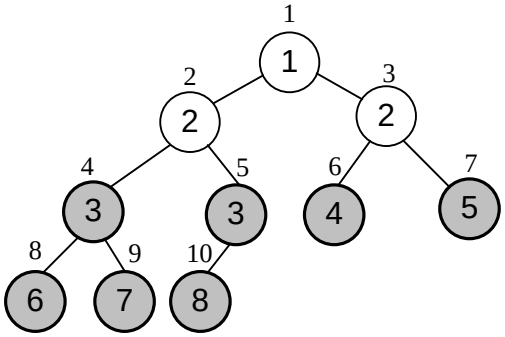


Рисунок 1.60 – Вигляд дерева після 7-ї ітерації

Масив *A* після 7-ї ітерації поданий на рис. 1.61.

№ елемента	1	2	3	4	5	6	7	8	9	10
A	1	2	2	3	3	4	5	6	7	8

Рисунок 1.61 – Масив *A* після 7-ї ітерації

Вершини №4– №10 виключаємо з подальшого розгляду.

Ітерація 8

На даній ітерації розглядаємо перші три вершини. Знайдений максимальний елемент розмістимо у вершину №3. Перевіряємо властивість купи для

вершини №1 та її потомків. Властивість не виконується, міняємо місцями елементи 1 і 3 (рис. 1.62).

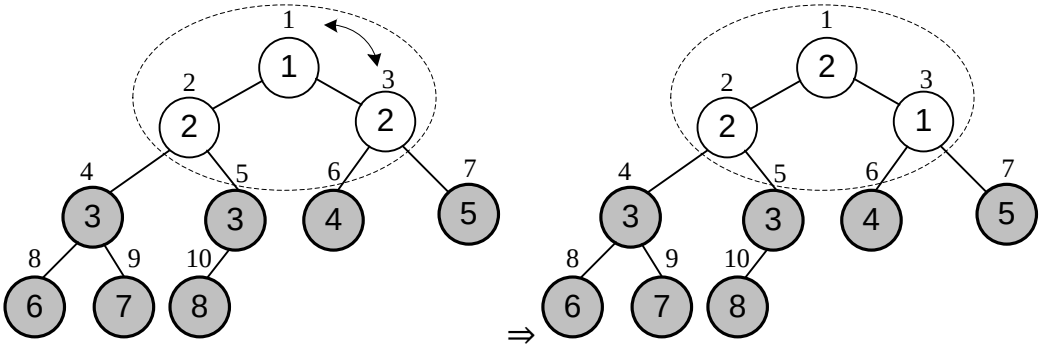


Рисунок 1.62 – Перевірка властивості купи для вершин №1, №2 та №3 (8-ма ітерація)

У вершини №3 немає потомків, які можна розглядати на даній ітерації, значить, закінчуємо перевіряти виконання властивості купи для дерева. У вершині №1 розташований максимальний елемент, міняємо його місцями з елементом у вершині №3 (рис. 1.63).

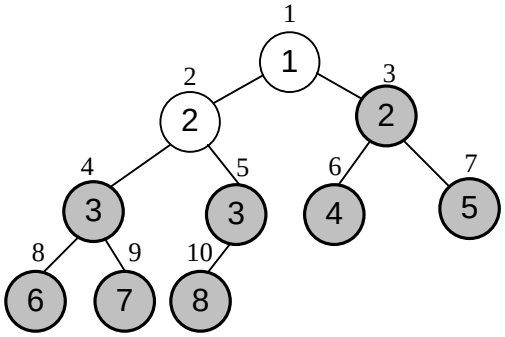


Рисунок 1.63 – Вигляд дерева після 8-ї ітерації

Масив *A* після 8-ї ітерації поданий на рис. 1.64.

Вершини №3– №10 виключаємо з подальшого розгляду.

№ елемента	1	2	3	4	5	6	7	8	9	10
A	1	2	2	3	3	4	5	6	7	8

Рисунок 1.64 – Масив *A* після 8-ї ітерації

Ітерація 9

Розглядаємо дві вершини, перевіряємо для них властивість купи (рис. 1.65). Властивість купи не виконується, тому міняємо місцями елементи в даних вершинах.

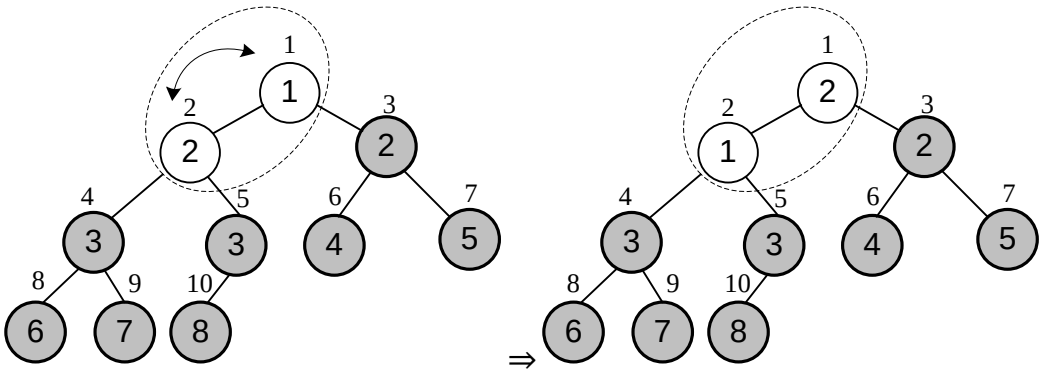


Рисунок 1.65 – Перевірка властивості купи для вершин №1 та №2 (9-та ітерація)

Оскільки у вершини №2 немає потомків, які можна розглянути, то закінчуємо перевіряти властивість купи, в першій вершині знаходиться максимальний елемент.

Елементи у вершинах №1 і №2 міняємо місцями (рис. 1.66).

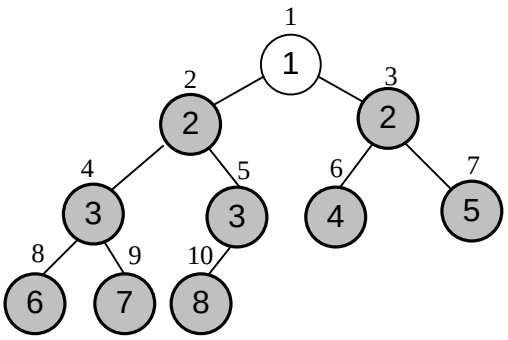


Рисунок 1.66 – Вигляд дерева після 9-ї ітерації

Масив *A* після 9-ї ітерації поданий на рис. 1.67.

№ елемента	1	2	3	4	5	6	7	8	9	10
A	1	2	2	3	3	4	5	6	7	8

Рисунок 1.67 – Масив *A* після 9-ї ітерації

Ітерація 10

Для розгляду залишився один елемент у вершині №1, з цього випливає, що всі елементи відсортовані в порядку зростання їх значень. На рис. 1.68 зображене дерево, яке було отримано в результаті сортування, на рис. 1.69 – відсортований масив A.

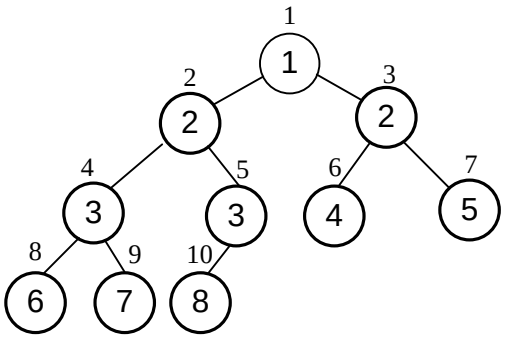


Рисунок 1.68 – Відсортоване дерево

№ елемента	1	2	3	4	5	6	7	8	9	10
A	1	2	2	3	3	4	5	6	7	8

Рисунок 1.69 – Відсортований масив A

Запитання для самоконтролю

1. Назвіть основні алгоритми сортування.
2. У чому полягає суть алгоритму вставками?
3. У чому полягає суть алгоритму швидкого сортування?
4. Для яких масивів використовується алгоритм сортування бульбашкою?

Задачі для аудиторних занять

1. Використовуючи сортування вставками, впорядкувати дані масиви:

а) за зростанням;

б) за спаданням.

Підрахувати кількість перестановок і порівнянь для обох випадків.

1) $10, 8, 7, 11, 5, 6, 4$; 2) $4, 5, 3, 8, 6, 5, 8$;

3) $3, 2, 6, 9, 5, 6, 5$; 4) $7, 6, 3, 4, 8, 5, 6$.

2. Відсортувати масив за зростанням, використовуючи сортування вставками і вибором:

1) $3, 8, 9, 2, 1, 5, 4, 7$; 2) $4, 5, 9, 8, 6, 5, 8, 9$;

3) $1, 8, 6, 9, 2, 4, 7, 3$; 4) $5, 6, 7, 9, 8, 6, 3, 8$.

Порівняти кількість перестановок і порівнянь для даних методів, зробити висновки.

3. Використовуючи сортування бульбашкою, розташувати елементи масиву з парними номерами за спаданням, а з непарними – за зростанням:

1)

1	2	3	4	5	6	7	8	9	10
11	12	10	14	8	9	6	10	7	9

2)

1		3		5	6	7	8	9	10
8	6	5	10	6	9		12	3	6

4. Упорядкувати масиви за спаданням, використовуючи для цього сортування злиттям:

1)

1	2	3	4	5	6	7	8	9	10	11	12
6	10	9	15	4	1	3	5	11	8	9	8

2)

1	2	3	4	5	6	7	8	9	10	11	12
7	2	4	8	7	3	12	7	1	5	6	2

3)

1		3	4	5	6	7	8	9	10	10	11
16	12	10	17	8	3	11	10		4	14	19

4)

1	2	3	4	5	6	7	8	9	10	11	12
5	1	15	11	4	10	2	9	12	8	14	3

5. Упорядкувати послідовність за зростанням на основі швидкого сортування. Як опорний елемент взяти:

а) крайній лівий елемент;

б) крайній правий елемент;

в) медіану значень першого, середнього і останнього елементів.

Порівняти ефективність алгоритму при різних виборах опорного елемента на даних масивах:

1) 5, 4, 5, 6, 7, 8, 8, 8, 9; 2) 3, 9, 7, 2, 5, 3, 9, 7, 8;
3) 4, 7, 3, 5, 1, 9, 2, 3, 5; 4) 6, 4, 8, 2, 3, 5, 8, 2, 1.

6. Використовуючи сортування підрахунком, упорядкувати за зростанням:

1) 4, 5, 0, 1, 0, 4, 3; 2) 5, 3, 3, 2, 2, 0, 5;
3) 2, 4, 1, 2, 3, 0, 3; 4) 1, 3, 4, 3, 1, 5, 4.

7. Упорядкувати масив за зростанням, використовуючи сортування купою:

	1	2	3	4	5	6	7	8	9	10
1)	5	2	1	9	8	3	6	5	7	2

	1	2	3	4	5	6	7	8	9	10
2)	12	15	14	9	7	10	5	11	4	5

	1	2	3	4	5	6	7	8	9	10
3)	3	8	2	8	10	6	11	1	5	7

	1	2	3	4	5	6	7	8	9	10
4)	2	3	1	8	7	5	2	11	5	10

8. Дано одновимірний числовий масив. Розташувати:

а) спочатку від’ємні елементи за спаданням, а потім додатні – за зростанням;

б) перші 5 елементів – за зростанням, інші – за спаданням:

	1	2	3	4	5	6	7	8	9	10
1)	–4	7	3	–2	6	0	3	–8	9	2

	1	2	3	4	5	6	7	8	9	0
2)	0	–6	5	–7	6	1	–5	10	–3	8

Обґрунтувати вибір методу сортування.

1.4. Алгоритмічні стратегії

1.4.1. Короткі теоретичні відомості

Розглянемо два важливих методи побудови та аналізу ефективних алгоритмів – динамічне програмування та жадібний алгоритм – на прикладі таких задач:

1) про маршрут на прямокутному полі;

2) про найбільшу спільну підпослідовність;

3) про вибір заявок.

Динамічне програмування – метод розв'язання задачі шляхом її розбиття на декілька однакових підзадач, рекурентно пов'язаних між собою. Суть динамічного програмування полягає в тому, щоб розв'язати кожну підзадачу лише один раз, скоротивши тим самим кількість обчислень, що є особливо корисним тоді, коли є велике число підзадач, що повторюються.

Алгоритм, оснований на динамічному програмуванні, будується таким чином:

1) описується будова оптимальних розв'язань;

2) виписується рекурентне співвідношення, яке пов'язує оптимальне значення параметра для підзадач;

3) рухаючись знизу вверх, обчислюється оптимальне значення параметра для підзадач;

4) користуючись отриманою інформацією, будується оптимальне розв'язання [1].

Задача про маршрут на прямокутному полі є однією з класичних задач динамічного програмування.

Задача 1

Дано прямокутне поле розміром $m \times n$, в кожній клітинці якого записане деяке число. Можна здійснювати кроки довжиною в одну клітинку вправо чи вниз. Необхідно потрапити з верхньої лівої клітинки до правої нижньої, набравши максимальну суму чисел відвіданих клітинок.

Уведемо позначення:

– $k[i, j]$ – число, записане в клітинці з координатами $[i, j]$ прямокутного поля ($i = \overline{1,m}$, $j = \overline{1,n}$);

– $s[i, j]$ – максимальна сума, яку можна набрати, якщо маршрут закін-

чується в клітинці $[i, j]$, де $i = \overline{1,m}$, $j = \overline{1,n}$.

Максимальна сума, яку можна набрати, якщо маршрут закінчується в клітинці $[1, 1]$, – це число, записане в даній клітинці, тобто $s[1,1] = k[1,1]$.

До будь-якої клітинки першого рядка, починаючи з другої, можна потрапити лише з клітинки зліва, значить, щоб знайти для неї максимальну суму, необхідно до значення максимальної суми, набраної в попередній зліва клітинці, додати число, записане в поточній клітинці:

$$s[1, j] = s[1, j-1] + k[1, j], \ j = \overline{2,n}. \tag{1.1}$$

До будь-якої клітинки першого стовпця, починаючи з другої, можна потрапити лише з клітинки зверху, значить, щоб знайти для неї максимальну суму, необхідно до значення максимальної суми, набраної в попередній зверху клітинці, додати число, записане в поточній клітинці:

$$s[i,1] = s[i-1,1] + k[i,1], \ i = \overline{2,m}. \tag{1.2}$$

До будь-якої клітинки з координатами $[i, j]$ (окрім клітинок першого стовпця і першого рядка) можна прийти тільки зверху чи зліва, тобто з клітинок з координатами $[i-1, j]$ та $[i, j-1]$. Щоб знайти максимальну суму для клітинки $[i, j]$, необхідно вибрати максимальну суму з клітинок $[i-1, j]$, $[i, j-1]$ і додати до неї число, записане в поточній клітинці:

$$s[i, j] = \max\{s[i-1, j], s[i, j-1]\} + k[i, j], \ i = \overline{2,m}, \ j = \overline{2,n}. \tag{1.3}$$

Таким чином, максимальна сума набирається послідовно, рядок за рядком зверху вниз (а в кожному рядку – зліва направо). Розв'язком задачі буде значення $s[m,n]$.

Задача про знаходження найбільшої спільної підпослідовності (англ. *longest common subsequence, LCS*) – одна з класичних задач інформатики. Вона формулюється таким чином.

Задача 2

Задані дві послідовності X та Y, необхідно для них знайти спільну підпослідовність найбільшої довжини.

Зауваження:

– підпослідовність можна отримати з деякої кінцевої послідовності, якщо видалити з останньої деяку множину її елементів (можливо і пусте);

– послідовність Z є спільною підпослідовністю послідовностей X та Y, якщо Z є підпослідовністю як X, так і Y;

– найбільших спільних підпослідовностей може бути декілька.

Запишемо вихідні послідовності як $X =< x_1, x_2, ... x_m >$ та $Y =< y_1, y_2, ... y_n >$. Позначимо $c[i, j]$ довжину найбільшої спільної підпослідовності для послідовностей X_i та Y_j (де $i = \overline{0,m}$, $j = \overline{0,n}$).

Для розв'язання даної задачі методом динамічного програмування використаємо таке рекурентне співвідношення [1]:

$$c[i, j] = \begin{cases} 0, \ \textit{якщо } i = 0 \ \textit{чи } \ j = 0, \\ c[i-1, j-1] + 1, \ \textit{якщо } i, j > 0 \ \textit{та } x_i = y_j, \\ \max\{c[i, j-1], c[i-1, j]\}, \ \textit{якщо } i, j > 0 \ \textit{та } x_i \neq y_j, \end{cases} \tag{1.4}$$

де $i = \overline{0,m}$, $j = \overline{0,n}$.

Якщо $i = 0$, то послідовність X_0 пуста, отже, $c[0, j] = 0$ для будь-яких підпослідовностей Y_j ($j = \overline{0,n}$). І навпаки, якщо $j = 0$, то послідовність Y_0 пуста, отже, $c[i,0] = 0$ для будь-яких підпослідовностей X_i ($i = \overline{0,m}$).

Числа $c[i, j]$, які обчислюються, записуються в масив рядками, а кожний рядок заповнюється зліва направо. Довжина найбільшої спільної підпослідовності для X та Y буде дорівнювати значенню $c[m, n]$.

Щоб побудувати найбільшу спільну підпослідовність, необхідно разом з обчисленням $c[i, j]$ запам'ятовувати для кожної задачі ту підзадачу, через яку вона розв'язується. Для цього використовується додатковий масив $a[1..m,1..n]$, до комірки $a[i, j]$ якого заноситься стрілка, що вказує на клітинку:

– $a[i-1, j-1]$, якщо $x_i = y_j$;

– $a[i-1, j]$, якщо $x_i \neq y_j$ і $c[i-1, j] \geq c[i, j-1]$;

– $a[i, j-1]$, якщо $x_i \neq y_j$ і $c[i-1, j] < c[i, j-1]$.

Далі необхідно, починаючи з останнього елемента масиву $a[m,n]$, піднятися вверх по комірках, на які вказують стрілки, і виписати ті x_i, яким у комірці $a[i, j]$ відповідають стрілки, що вказують на лівий верхній кут, тобто на клітинку $a[i-1, j-1]$. Це і буде розв'язком задачі.

Суть жадібного алгоритму (англ. *Greedy algorithm*) – прийняття локально оптимальних рішень на кожному етапі, допускаючи, що кінцеве рішення також виявиться оптимальним. Як зазначають автори роботи [1], це не завжди так, але для більшого числа алгоритмічних задач жадібні алгоритми дійсно дають оптимальне рішення. Однією з таких задач і є задача про вибір заявок.

Задача 3

Дано n заявок на проведення занять в деякій аудиторії. В кожній i-й заявці вказані початок a_i і кінець заняття b_i. Заявки називаються сумісними, якщо інтервали їх проведення не пересікаються (але початок одного заняття може збігатися із закінченням другого). Необхідно набрати максимальну кількість сумісних одна з одною заявок.

Алгоритм розв'язання задачі.

На вхід даному алгоритму подається множина заявок на проведення занять з часом їх початку та кінця, відсортована за зростанням часу закінчення. Результатом роботи алгоритму буде множина A, що містить номери вибраних заявок. Перша заявка поміщується до множини A, а змінній j присвоюється її номер. Жадібний алгоритм шукає заявку, що починається не раніше від закінчення j-ї, потім знайдену заявку включає до A, а j присвоює її номер. Таким чином, кожного разу вибирається заявка на проведення заняття, до кінця якого залишилось найменше часу. Алгоритм закінчується, коли всі заявки будуть розглянуті.

Для поглибленого вивчення даної теми пропонується використовувати роботи [1, 2].

1.4.2. Приклади розв'язання задач

Задача 1

Розв'язати задачу про маршрут на прямокутному полі, яке подане на рис. 1.70.

2	3	4	11	8
10	7	8	1	2
6	3	3	4	6
4	5	9	6	7

Рисунок 1.70 – Зовнішній вигляд вихідного прямокутного поля

Розв'язання

Випишемо значення елементів початкового масиву K:

$k[1,1] = 2;\ k[1,2] = 3;\ k[1,3] = 4;\ k[1,4] = 11;\ k[1,5] = 8;$

$k[2,1] = 10;\ k[2,2] = 7;\ k[2,3] = 8;\ k[2,4] = 1;\ k[2,5] = 2;$

$k[3,1] = 6;\ k[3,2] = 3;\ k[3,3] = 3;\ k[3,4] = 4;\ k[3,5] = 6;$

$k[4,1] = 4;\ k[4,2] = 5;\ k[4,3] = 9;\ k[4,4] = 6;\ k[4,5] = 7.$

Заповнення результуючого масиву M здійснюється зліва направо за рекурентними формулами (1.1-1.3). Обчислимо елементи першого рядка:

$m[1,1] = k[1,1] = 2;$

$m[1,2] = m[1,1] + k[1,2] = 2 + 3 = 5;$

$m[1,3] = m[1,2] + k[1,3] = 5 + 4 = 9;$

$m[1,4] = m[1,3] + k[1,4] = 9 + 11 = 20;$

$m[1,5] = m[1,4] + k[1,5] = 20 + 8 = 28.$

Обчислимо перший елемент другого рядка

$$m[2,1] = m[1,1] + k[2,1] = 2 + 10 = 12.$$

Інші елементи другого рядка обчислюються як сума двох доданків. Перший з доданків – це максимальний з елемента результуючого масиву, розташованого рядком вище, та елемента, розташованого стовпцем лівіше від елемента, що розглядається; другий з доданків – елемент початкового масиву:

$$m[2,j] = \max\{m[2,j-1], m[1,j]\} + k[2,j], j = \overline{2,5}.$$

$m[2,2] = \max\{m[2,1], m[1,2]\} + k[2,2] = \max\{12,5\} + 7 = 12 + 7 = 19;$

$m[2,3] = \max\{m[2,2], m[1,3]\} + k[2,3] = \max\{19,9\} + 8 = 19 + 8 = 27;$

$m[2,4] = \max\{m[2,3], m[1,4]\} + k[2,4] = \max\{27,20\} + 1 = 27 + 1 = 28;$
$m[2,5] = \max\{m[2,5], m[1,5]\} + k[2,5] = \max\{28,28\} + 2 = 28 + 2 = 30.$

Аналогічно обчислюємо елементи 3-го рядка:

$m[3,1] = m[2,1] + k[3,1] = 12 + 6 = 18;$
$m[3,2] = \max\{m[3,1], m[2,2]\} + k[3,2] = \max\{18,19\} + 3 = 19 + 3 = 22;$
$m[3,3] = \max\{m[3,2], m[2,3]\} + k[3,3] = \max\{22,27\} + 3 = 27 + 3 = 30;$
$m[3,4] = \max\{m[3,3], m[2,4]\} + k[3,4] = \max\{30,28\} + 4 = 30 + 4 = 34;$
$m[3,5] = \max\{m[3,4], m[2,5]\} + k[3,5] = \max\{34,30\} + 6 = 34 + 6 = 40.$

Аналогічно обчислюємо елементи 4-го рядка:

$m[4,1] = m[3,1] + k[4,1] = 18 + 4 = 22;$
$m[4,2] = \max\{m[4,1], m[3,2]\} + k[4,2] = \max\{22,22\} + 5 = 22 + 5 = 27;$
$m[4,3] = \max\{m[4,2], m[3,3]\} + k[4,3] = \max\{27,30\} + 9 = 30 + 9 = 39;$
$m[4,4] = \max\{m[4,3], m[3,4]\} + k[4,4] = \max\{39,34\} + 6 = 39 + 6 = 45;$
$m[4,5] = \max\{m[4,4], m[3,5]\} + k[4,5] = \max\{45,40\} + 7 = 45 + 7 = 52.$

Значення останнього елемента масиву $m[4,5] = 52$ є розв'язком задачі. На рис. 1.71 поданий результат розв'язання задачі.

2	5	9	20	28
12	19	27	28	30
18	22	30	34	40
22	27	39	45	52

Рисунок 1.71 – Результат розв'язання задачі

Задача 2

Дані дві підпослідовності $X = \langle BACAABDA \rangle$ та $Y = \langle ABACDAB \rangle$. Знайти найбільшу спільну підпослідовність.

Розв'язання

Спочатку побудуємо таблицю, яка містить $(n + 1)$ стовпців (з номерами від 0 до n), $(m + 1)$ строк (з номерами від 0 до m), де n – число членів послідовності X, m – число членів послідовності Y.

У відповідності до рекурентного співвідношення (1.4) запишемо нулі в нульовий стовпчик та нульову строку таблиці. Тоді таблиця буде мати вигляд, поданий на рис. 1.72.

	j	0	1	2	3	4	5	6	7
i		y_j	A	B	A	C	D	A	B
0	x_i	0	0	0	0	0	0	0	0
1	B	0							
2	A	0							
3	C	0							
4	A	0							
5	A	0							
6	B	0							
7	D	0							
8	A	0							

Рисунок 1.72 – Вигляд таблиці із заповненими першим рядком та стовпцем

Обчислимо інші елементи таблиці, починаючи з першого елемента першого рядка (див. рекурентне співвідношення 1.4).

Порівняємо x_1 та y_1, оскільки $x_1 = B$, $y_1 = A$, то $x_1 \neq y_1$, а значить, значення $h[1,1]$ дорівнює значенню максимального із елементів $h[0,1]$ і $h[1,0]$, тобто $h[1,1] = \max\{h[0,1], h[1,0]\} = \max\{0, 0\} = 0$.

Оскільки $h[0,1] = h[1,0]$, то можна вибрати будь-який з напрямків (вліво чи вгору). Для визначеності оберемо напрямок вгору. Обраний напрямок покажемо стрілкою у відповідній клітинці таблиці.

Далі порівнюємо x_1 та y_2, оскільки $x_1 = B$, $y_2 = B$, то $x_1 = y_2$, а значить, значення $h[1,2]$ на одиницю більше від значення елемента у верхній діагональній клітинці: $h[1,2] = h[0,1] + 1 = 0 + 1 = 1$. Фіксуємо відповідний напрямок.

Аналогічно обчислюємо значення інших елементів рядка:

$x_1 = B, y_3 = A \Rightarrow x_1 \neq y_3, \Rightarrow h[1,3] = \max\{h[0,3], h[1,2]\} = \max\{0,1\} = 1;$

$x_1 = B, y_4 = C \Rightarrow x_1 \neq y_4, \Rightarrow h[1,4] = \max\{h[0,4], h[1,3]\} = \max\{0,1\} = 1;$

$x_1 = B, y_5 = D \Rightarrow x_1 \neq y_5, \Rightarrow h[1,5] = \max\{h[0,5], h[1,4]\} = \max\{0,1\} = 1;$

$x_1 = B, y_6 = A \Rightarrow x_1 \neq y_6, \Rightarrow h[1,6] = \max\{h[0,6], h[1,5]\} = \max\{0,1\} = 1;$

$x_1 = B, y_7 = B \Rightarrow x_1 = y_7, \Rightarrow h[1,7] = h[0,6] + 1 = 0 + 1 = 1.$

Відповідні результати обчислень елементів першої строки наведені на рис. 1.73.

	j	0	1	2	3	4	5	6	7
i		y_j	A	B	A	C	D	A	B
0	x_i	0	0	0	0	0	0	0	0
1	B	0	↑0	↖ 1	←1	←1	←1	←1	↖ 1
2	A	0							
3	C	0							
4	A	0							
5	A	0							
6	B	0							
7	D	0							
8	A	0							

Рисунок 1.73 – Результат обчислень елементів 1-го рядка

Обчислимо елементи 2-го рядка:

$x_2 = A, y_1 = A \Rightarrow x_2 = y_1, \Rightarrow h[2,1] = h[1,0] + 1 = 0 + 1 = 1;$

$x_2 = A, y_2 = B \Rightarrow x_2 \neq y_2, \Rightarrow h[2,2] = \max\{h[1,2], h[2,1]\} = \max\{1,1\} = 1;$

$x_2 = A, y_3 = A \Rightarrow x_2 = y_3, \Rightarrow h[2,3] = h[1,2] + 1 = 1 + 1 = 2;$

$x_2 = A, y_4 = C \Rightarrow x_2 \neq y_4, \Rightarrow h[2,4] = \max\{h[1,4], h[2,3]\} = \max\{1,2\} = 2;$

$x_2 = A, y_5 = D \Rightarrow x_2 \neq y_5, \Rightarrow h[2,5] = \max\{h[1,5], h[2,4]\} = \max\{1,2\} = 2;$

$x_2 = A, y_6 = A \Rightarrow x_2 = y_6, \Rightarrow h[2,6] = h[1,5] + 1 = 1 + 1 = 2;$

$x_2 = A, y_7 = B \Rightarrow x_2 \neq y_7, \Rightarrow h[2,7] = \max\{h[1,7], h[2,6]\} = \max\{1,2\} = 2.$

Елементи 3-го рядка:

$x_3 = C, y_1 = A \Rightarrow x_3 \neq y_1, \Rightarrow h[3,1] = \max\{h[2,1], h[3,0]\} = \max\{1,0\} = 1;$

$x_3 = C, y_2 = B \Rightarrow x_3 \neq y_2, \Rightarrow h[3,2] = \max\{h[2,2], h[3,1]\} = \max\{1,1\} = 1;$

$x_3 = C, y_3 = A \Rightarrow x_3 \neq y_3, \Rightarrow h[3,3] = \max\{h[2,3], h[3,2]\} = \max\{2,1\} = 2;$

$x_3 = C, y_4 = C \Rightarrow x_3 = y_4, \Rightarrow h[3,4] = h[2,3] + 1 = 2 + 1 = 3;$

$x_3 = C, y_5 = D \Rightarrow x_3 \neq y_5, \Rightarrow h[3,5] = \max\{h[2,5], h[3,4]\} = \max\{2,3\} = 3;$

$x_3 = C, y_6 = A \Rightarrow x_3 \neq y_6, \Rightarrow h[3,6] = \max\{h[2,6], h[3,5]\} = \max\{2,3\} = 3;$

$x_3 = C, y_7 = B \Rightarrow x_3 \neq y_7, \Rightarrow h[3,7] = \max\{h[2,7], h[3,6]\} = \max\{2,3\} = 3.$

Елементи 4-го рядка:

$x_4 = A, y_1 = A \Rightarrow x_4 = y_1, \Rightarrow h[4,1] = h[3,0] + 1 = 0 + 1 = 1;$

$x_4 = A, y_2 = B \Rightarrow x_4 \neq y_2, \Rightarrow h[4,2] = \max\{h[3,2], h[4,1]\} = \max\{1,1\} = 1;$

$x_4 = A, y_3 = A \Rightarrow x_4 = y_3, \Rightarrow h[4,3] = h[3,2] + 1 = 1 + 1 = 2;$

$x_4 = A, y_4 = C \Rightarrow x_4 \neq y_4, \Rightarrow h[4,4] = \max\{h[3,4], h[4,3]\} = \max\{3,2\} = 3;$

$x_4 = A, y_5 = D \Rightarrow x_4 \neq y_5, \Rightarrow h[4,5] = \max\{h[3,5], h[4,4]\} = \max\{3,3\} = 3;$
$x_4 = A, y_6 = A \Rightarrow x_4 = y_6, \Rightarrow h[4,6] = h[3,5] + 1 = 3 + 1 = 4;$
$x_4 = A, y_7 = B \Rightarrow x_4 \neq y_7, \Rightarrow h[4,7] = \max\{h[3,7], h[4,6]\} = \max\{3,4\} = 4.$

Елементи 5-го рядка:

$x_5 = A, y_1 = A \Rightarrow x_5 = y_1, \Rightarrow h[5,1] = h[4,0] + 1 = 0 + 1 = 1;$
$x_5 = A, y_2 = B \Rightarrow x_5 \neq y_2, \Rightarrow h[5,2] = \max\{h[4,2], h[5,1]\} = \max\{1,1\} = 1;$
$x_5 = A, y_3 = A \Rightarrow x_5 = y_3, \Rightarrow h[5,3] = h[4,2] + 1 = 1 + 1 = 2;$
$x_5 = A, y_4 = C \Rightarrow x_5 \neq y_4, \Rightarrow h[5,4] = \max\{h[4,4], h[5,3]\} = \max\{3,2\} = 3;$
$x_5 = A, y_5 = D \Rightarrow x_5 \neq y_5, \Rightarrow h[5,5] = \max\{h[4,5], h[5,4]\} = \max\{3,3\} = 3;$
$x_5 = A, y_6 = A \Rightarrow x_5 = y_6, \Rightarrow h[5,6] = h[4,5] + 1 = 3 + 1 = 4;$
$x_5 = A, y_7 = B \Rightarrow x_5 \neq y_7, \Rightarrow h[5,7] = \max\{h[4,7], h[5,6]\} = \max\{4,4\} = 4.$

Елементи 6-го рядка:

$x_6 = B, y_1 = A \Rightarrow x_6 \neq y_1, \Rightarrow h[6,1] = \max\{h[5,1], h[6,0]\} = \max\{1,0\} = 1;$
$x_6 = B, y_2 = B \Rightarrow x_6 = y_2, \Rightarrow h[6,2] = h[5,1] + 1 = 1 + 1 = 2;$
$x_6 = B, y_3 = A \Rightarrow x_6 \neq y_3, \Rightarrow h[6,3] = \max\{h[5,3], h[6,2]\} = \max\{2,2\} = 2;$
$x_6 = B, y_4 = C \Rightarrow x_6 \neq y_4, \Rightarrow h[6,4] = \max\{h[5,4], h[6,3]\} = \max\{3,2\} = 3;$
$x_6 = B, y_5 = D \Rightarrow x_6 \neq y_5, \Rightarrow h[6,5] = \max\{h[5,5], h[6,4]\} = \max\{3,3\} = 3;$
$x_6 = B, y_6 = A \Rightarrow x_6 \neq y_6, \Rightarrow h[6,6] = \max\{h[5,6], h[6,6]\} = \max\{4,3\} = 4;$
$x_6 = B, y_7 = B \Rightarrow x_6 = y_7, \Rightarrow h[6,7] = h[5,6] + 1 = 4 + 1 = 5.$

Елементи 7-го рядка:

$x_7 = D, y_1 = A \Rightarrow x_7 \neq y_1, \Rightarrow h[7,1] = \max\{h[6,1], h[7,0]\} = \max\{1,0\} = 1;$
$x_7 = D, y_2 = B \Rightarrow x_7 \neq y_2, \Rightarrow h[7,2] = \max\{h[6,2], h[7,1]\} = \max\{2,1\} = 2;$
$x_7 = D, y_3 = A \Rightarrow x_7 \neq y_3, \Rightarrow h[7,3] = \max\{h[6,3], h[7,2]\} = \max\{2,2\} = 2;$
$x_7 = D, y_4 = C \Rightarrow x_7 \neq y_4, \Rightarrow h[7,4] = \max\{h[6,4], h[7,3]\} = \max\{3,2\} = 3;$
$x_7 = D, y_5 = D \Rightarrow x_7 = y_7, \Rightarrow h[7,5] = h[6,4] + 1 = 3 + 1 = 4;$
$x_7 = D, y_6 = A \Rightarrow x_7 \neq y_6, \Rightarrow h[7,6] = \max\{h[6,6], h[7,5]\} = \max\{4,4\} = 4;$
$x_7 = D, y_7 = B \Rightarrow x_7 \neq y_7, \Rightarrow h[7,7] = \max\{h[6,7], h[7,6]\} = \max\{5,4\} = 5.$

Елементи 8-го рядка:

$x_8 = A, y_1 = A \Rightarrow x_8 = y_1, \Rightarrow h[8,1] = h[7,0] + 1 = 0 + 1 = 1;$
$x_8 = A, y_2 = B \Rightarrow x_8 \neq y_2, \Rightarrow h[8,2] = \max\{h[7,2], h[8,1]\} = \max\{2,1\} = 2;$
$x_8 = A, y_3 = A \Rightarrow x_8 = y_3, \Rightarrow h[8,3] = h[7,2] + 1 = 2 + 1 = 3;$
$x_8 = A, y_4 = C \Rightarrow x_8 \neq y_4, \Rightarrow h[8,4] = \max\{h[7,4], h[8,3]\} = \max\{3,3\} = 3;$
$x_8 = A, y_5 = D \Rightarrow x_8 \neq y_5, \Rightarrow h[8,5] = \max\{h[7,5], h[8,4]\} = \max\{4,3\} = 4;$
$x_8 = A, y_6 = A \Rightarrow x_8 = y_6, \Rightarrow h[8,6] = h[7,5] + 1 = 4 + 1 = 5;$

$x_8 = A, y_7 = B \Rightarrow x_8 \neq y_7, \Rightarrow h[8,7] = \max\{h[7,7], h[8,6]\} = \max\{5,5\} = 5.$

Підсумкові результати обчислень елементів подані на рис. 1.74.

Цифра 5 у нижній правій клітинці показує кількість елементів в найбільшій спільній підпослідовності.

	j	0	1	2	3	4	5	6	7
i		y_j	A	B	A	C	D	A	B
0	x_i	0	0	0	0	0	0	0	0
1	B	0	↑ 0	↖ 1	← 1	← 1	← 1	← 1	↖ 1
2	A	0	↖ 1	↑ 1	↖ 2	← 2	← 2	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↑ 2	↖ 3	← 3	← 3	← 3
4	A	0	↖ 1	↑ 1	↖ 2	↑ 3	↑ 3	↖ 4	← 4
5	A	0	↖ 1	↑ 1	↖ 2	↑ 3	↑ 3	↖ 4	↑ 4
6	B	0	↑ 1	↖ 2	↑ 2	↑ 3	↑ 3	↑ 4	↖ 5
7	D	0	↑ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4	↑ 5
8	A	0	↖ 1	↑ 2	↖ 3	↑ 3	↑ 4	↖ 5	↑ 5

Рисунок 1.74 – Підсумкові результати обчислень елементів результуючої таблиці

Підніємося від клітинки $h[8,7]$ таблиці за стрілками вгору, в таблиці на рис. 1.74 даний маршрут виділений сірим кольором. Тепер випишемо результуючу спільну підпослідовність, використовуючи ті сірі клітинки, де стрілка вказує на верхній лівий кут: $\langle B, A, C, A, B \rangle$.

Задача 3

Дано 10 заявок на проведення занять в одній і тій самій аудиторії:

$[0,4); [1,3); [1,5); [2,8); [4,7); [4,8); [6,12); [8,10); [8,11); [10,12).$

Вибрати максимальну кількість сумісних одна з одною заявок.

Розв'язання

Спочатку впорядкуємо заявки за зростанням часу їх закінчення. Заявки з однаковим часом закінчення розташуємо в довільному порядку:

$$a_1 = [1,3), a_2 = [0,4), a_3 = [1,5), a_4 = [4,7), a_5 = [2,8), a_6 = [4,8),$$
$$a_7 = [8,10), a_8 = [8,11), a_9 = [6,12), a_{10} = [10,12).$$

На першому етапі вибираємо першу заявку. Будемо показувати вибрані заявки затемненими (рис. 1.75 (етап I та далі)).

На другому етапі шукають заявку, що починається не раніше, ніж закін-

чується перша заявка. Розглянемо заявки в порядку зростання їх номерів. Як показано на рис. 1.75 (II етап), заявки a_2 та a_3 починаються раніше, ніж закінчується перша заявка, тому вони виключаються з подальшого розгляду. Заявка a_4 задовольняє вимогам, тому на II етапі вибирається саме вона.

Переходимо до наступного етапу. Перевіряємо послідовно заявки a_5,a_6. Оскільки час їх початку менший, ніж час закінчення заявки a_4, то дані заявки виключаються з подальшого розгляду. Час початку заявки a_7 пізніший, ніж закінчення a_4, тому a_7 додається до загального розв'язку (див. рис. 1.75 (III етап)).

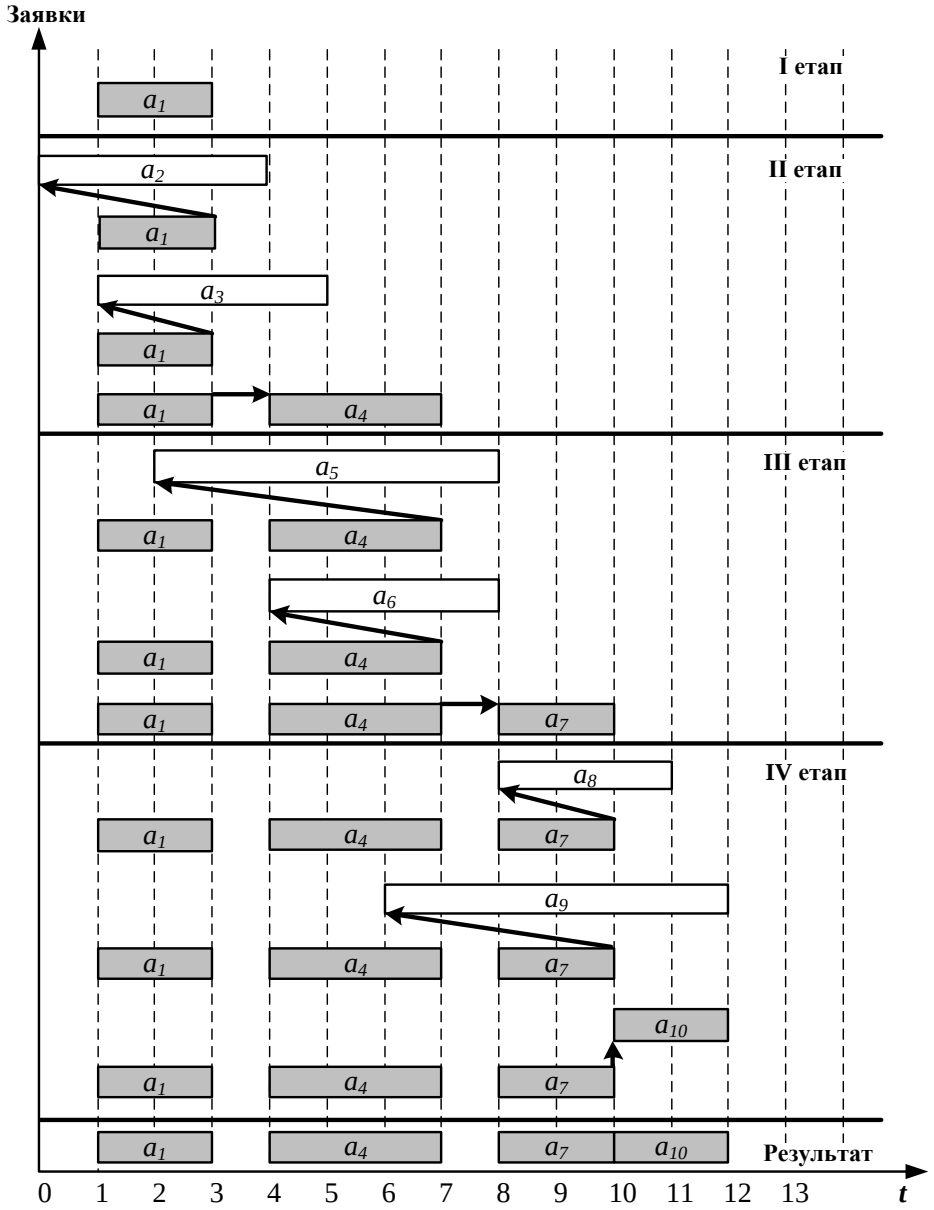


Рисунок 1.75 – Розв'язання задачі про заявки жадібним алгоритмом

На IV етапі виключаємо заявки a_8 та a_9, оскільки час їх початку менший від часу закінчення заявки a_7. Заявка a_{10} додається до загального розв'язку, оскільки час її початку збігається з часом закінчення заявки a_7.

Таким чином, оптимальний розв'язок містить заявки a_1, a_4, a_7 та a_{10}.

Запитання для самоконтролю

1. Дайте визначення підпослідовності.

2. Яка підпослідовність називається «найбільшою спільною підпослідовністю»?

3. Сформулюйте задачу про найбільшу спільну підпослідовність.

4. Чи притаманна для задачі про найбільшу спільну підпослідовність властивість оптимальності для підзадач?

5. Запишіть рекурентне співвідношення для вартості оптимального рішення в задачі про найбільшу спільну підпослідовність.

6. У чому полягає суть жадібного алгоритму?

7. Сформулюйте задачу про вибір заявок.

8. Опишіть особливості застосування жадібного алгоритму.

9. Сформулюйте принцип жадібного вибору.

Задачі для аудиторних занять

1. Розв'язати задачу про маршрут на прямокутному полі, поданому нижче.

а)

2	3	4	5
7	4	6	7
3	4	3	4

б)

5	1	2	6
3	6	4	2
4	3	5	8

в)

8	7	6	2
2	9	1	3
1	8	3	4

г)

3	2	1	4	3
2	5	3	6	2
1	7	1	8	4
3	6	8	5	2

д)

4	1	5	3	6
2	1	6	8	7
5	3	1	2	4
4	2	7	3	8

е)

7	3	6	5
4	2	9	6
6	7	2	2
5	1	4	7

2. Знайти найбільшу спільну підпослідовність таких послідовностей:

а) $X = \langle FMNNOPPEK \rangle$, $Y = \langle NOFFPTEKK \rangle$;

б) $X = \langle LLEONMBBC \rangle$, $Y = \langle ALTONKBCC \rangle$;

в) $X = \langle ABCDEFABC \rangle$, $Y = \langle BBCEEFAAB \rangle$;

г) $X = \langle DEFFABCDK \rangle$, $Y = \langle FADDBDCD \rangle$;

д) $X = \langle AFBCCDEF \rangle$, $Y = \langle BFCBDFECA \rangle$;

е) $X = \langle BCCLMON \rangle$, $Y = \langle CLKOONBL \rangle$;

є) $X = \langle KLMALETK \rangle$, $Y = \langle LKATLDEKT \rangle$;

ж) $X = \langle DAECNFD \rangle$, $Y = \langle ECDEKND \rangle$;

з) $X = \langle LMCTDAABCL \rangle$, $Y = \langle MATCALBAL \rangle$.

3. Перевірити, чи є підпослідовність $MLCDF$ найбільшою спільною підпослідовністю двох поданих послідовностей:

а) $X = \langle FMLKCDAF \rangle$, $Y = \langle MFLTCADF \rangle$;

б) $X = \langle AMLACDAF \rangle$, $Y = \langle MACALMDF \rangle$.

4. Указати всі найбільші спільні підпослідовності двох послідовностей:

а) $X = \langle ACDBL \rangle$, $Y = \langle CADBBL \rangle$;

		A	C	D	B	L
	0	0	0	0	0	0
C	0	← 0	↖ 1	← 1	← 1	← 1
A	0	↖ 1	← 1	← 1	← 1	← 1
D	0	↑ 1	← 1	↖ 2	← 2	← 2
B	0	↑ 1	← 1	↖ 2	↖ 3	← 3
B	0	↑ 1	← 1	← 1	↖ 3	← 3
L	0	↑ 1	← 1	← 1	← 1	↖ 4

б) $X = \langle BCAFKM \rangle$, $Y = \langle CBAKFM \rangle$;

		B	C	A	F	K	M
	0	0	0	0	0	0	0
C	0	← 0	↖ 1	← 1	← 1	← 1	← 1
B	0	↖ 1	← 1	← 1	← 1	← 1	← 1
A	0	↑ 1	← 1	↖ 2	← 2	← 2	← 2
K	0	↑ 1	← 1	↑ 2	← 2	↖ 3	← 3
F	0	↑ 1	← 1	↑ 2	↖ 3	← 3	← 3
M	0	↑ 1	← 1	↑ 2	↑ 3	← 3	↖ 4

5. Розв’язати задачу про заявки жадібним алгоритмом. Вихідна множина заявок наведена нижче:

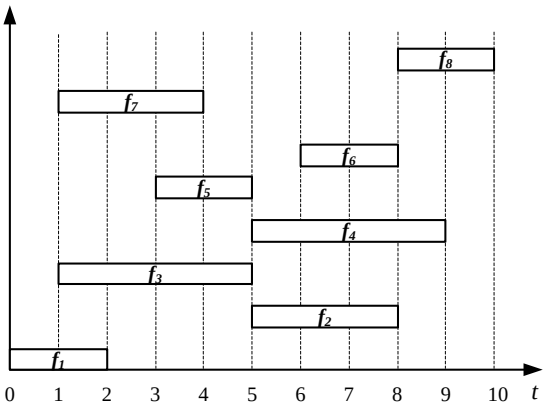
1) $f_1 = [1,3);\ f_2 = [2,4);\ f_3 = [1,4);\ f_4 = [3,5);\ f_5 = [5,6);\ f_6 = [4,8);$

2) $f_1 = [1,4);\quad f_2 = [2,5);\quad f_3 = [6,7);\quad f_4 = [5,6);\quad f_5 = [4,5);\quad f_6 = [6,8);$ $f_7 = [8,10);\ f_8 = [7,9);\ f_9 = [10,11);$

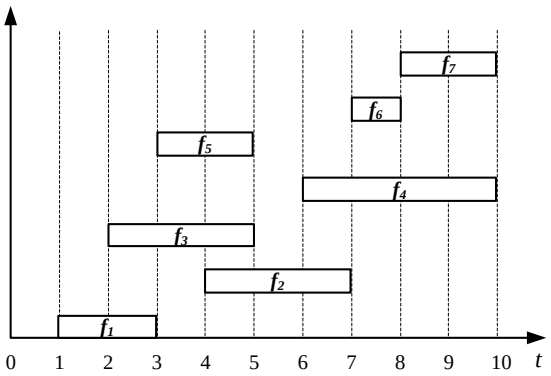
3) $f_1 = [2,6);\ \ f_2 = [1,2);\ \ f_3 = [7,12);\ \ f_4 = [7,9);\ \ f_5 = [6,7);\ \ f_6 = [11,12);$ $f_7 = [10,12);\ f_{81} = [3,8);\ f_9 = [3,6);$

4) $f_1 = [2,6);\ f_2 = [6,8);\ f_1 = [1,5);\ f_4 = [4,6);\ f_5 = [5,7);\ \ f_6 = [3,5);$ $f_7 = [2,3);\ f_8 = [3,6);\ f_9 = [1,2);\ f_{10} = [5,8);$

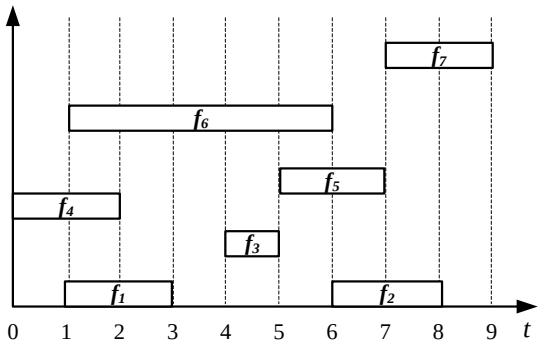
5)



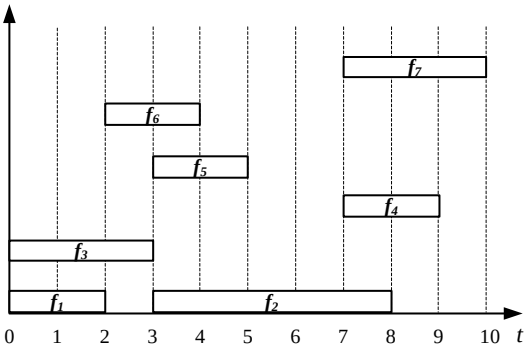
6)



7)



8)



1.5. Машина Тюрінга

1.5.1. Короткі теоретичні відомості

Під машиною Тюрінга розуміють деяку гіпотетичну (умовну) машину, що складається з таких частин:

1) інформаційної стрічки, що являє собою необмежену пам’ять машини, розділену на окремі комірки, в кожній комірці можна розмістити лише один символ, в тому числі 0;

2) головки, що зчитує, – спеціального чуттєвого елемента, який

здатний оглядати вміст поточної комірки; вздовж головки інформаційна стрічка може переміщатися так, щоб у кожний момент часу, що розглядається, головка знаходилась над визначеною коміркою;

3) керуючого пристрою, який в кожний момент часу, що розглядається, знаходиться в деякому стані.

Допускається скінченність таких станів.

Стан керуючого пристрою часто називають внутрішнім станом машини. Один з таких станів називається завершальним і відповідає завершенню роботи машини.

У кожній комірці машини Тюрінга може знаходитися один із символів кінцевого алфавіту, а пристрій керування може бути в одному із кінцевих станів.

Машина Тюрінга може зсувати стрічку на одну комірку вправо чи вліво, залишаючи її вміст незмінним, а також може змінювати вміст комірки, що сприймається, залишаючи стрічку нерухомою. Вказаний список операцій може бути розширений.

Машина Тюрінга називається стандартною, якщо вона при зсуві стрічки може попередньо змінювати стан комірки, яка зчитується.

Нехай $A = \{S_0, \dots, S_n\}$ – алфавіт машини Тюрінга, де S_0 – пуста комірка, $Q = \{q_0, \dots, q_m\}$ – множина станів керуючого пристрою, де q_0 – завершальний, чи кінцевий, стан.

Кінцева сукупність символів алфавіту A називається зовнішнім алфавітом, а кінцева сукупність станів керуючого пристрою Q – внутрішнім алфавітом.

Сукупність, утворена послідовністю станів усіх комірок стрічки і станом керуючого пристрою, називається конфігурацією машини, яка задається у вигляді слова, що описує певний стан машини.

Конфігурація машини Тюрінга для випадку, показаного на рис. 1.76, може бути записана таким чином:

$$S_{j1}\, S_{j2}\, S_{j3} \dots q_i\, S_{jk} \dots\, S_{jr},$$

де r – число заповнених комірок на стрічці;

q_i – стан пристрою керування.

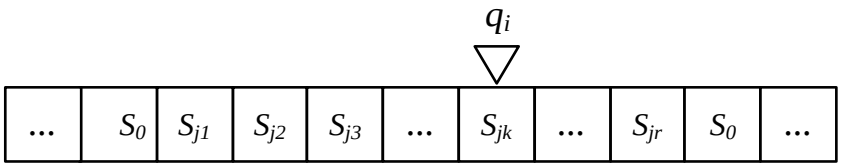


Рисунок 1.76 – Стрічка машини Тюрінга

Кожна конфігурація містить лише одне входження символу q_i із внутрішнього алфавіту, який може бути самим лівим, але не може бути самим правим, так як справа від нього повинен поміщатися символ стану комірки, що розглядається. S_0 – символ, що позначає пусту комірку.

Якщо стандартна машина Тюрінга, знаходячись в стані q_i і сприймаючи записаний на стрічці символ S_k, переходить до нового стану q_j, здійснюючи при цьому заміну символу в комірці, що розглядається, на символ S_m і зсув стрічки вліво на одну комірку, то говорять, що машина виконує команду:

$$q_i S_k \rightarrow q_j S_m \text{ Л}.$$

Стрілка в запису команди може бути відсутня:

$$q_i S_k \; q_j S_m \text{ Л}.$$

При маніпуляції зі стрічкою приймаються позначення:

Л – рух стрічки вліво;

П – рух стрічки вправо;

С – стрічка не рухається.

Стандартна машина Тюрінга називається нестираючою, якщо вона здатна виконувати лише команди виду:

$$q_\alpha 0 q_\beta u T$$
$$q_\alpha 1 q_\beta 1 T$$
$$u = \{0,1\}$$
$$T = \{\text{Л}, \text{П}, \text{С}\}.$$

Сукупність усіх команд, які може виконувати машина Тюрінга, називається її програмою.

Програму машини Тюрінга можна подати у вигляді таблиці відповідності (табл. 1.3), де зліва в строках перечислюються всі стани, в яких може знаходитися машина, а зверху (в стовпцях) – всі символи алфавіту зовнішнього алфавіту A.

Таблиця 1.3 – Таблиця відповідності машини Тюрінга

Q \ A	s_1	s_2	s_3	…	s_n
q_0	$q_1 s_2 Л$	…	…	…	…
q_1	…	…	…	…	…
q_2	…	…	…	…	…
…	…	…	…	…	…
q_m	…	…	…	…	…

Тоді на перетині відповідного рядка і стовпця ми можемо знайти ті дії, які повинна виконати машина Тюрінга, якщо вона знаходиться у відповідному стані і сприймає відповідний символ. Так, наприклад, якщо машина знаходиться в стані q_0 і в комірці, що сприймається, написаний символ s_1, то на перетині відповідних рядка і стовпця в таблиці 1.3 ми читаємо $q_1 s_2 Л$, тобто символ у комірці зміниться на s_2, стрічка зсунеться вліво, а машина перейде до стану q_1.

Машина Тюрінга вважається заданою, якщо задані її внутрішній та зовнішній алфавіти, програма, початкова конфігурація і вказано, які символи позначають пусту комірку і завершальний стан.

Для поглибленого вивчення даної теми пропонується використовувати додаткову літературу [8-10].

1.5.2. Приклади розв'язання задач

Задача 1

Маємо машину Тюрінга з алфавітами $A = \{0,1\}$, $Q = \{q_0, q_1\}$ і командами:

$$q_1 1 q_1 1 Л$$
$$q_1 0 q_0 1 С.$$

Нехай на стрічці записано слово 10100. Початкова конфігурація $10 q_1 100$.

Яке слово отримаємо в результаті виконання двох послідовних команд?

Розв'язання

Зважаючи на те, що машина Тюрінга знаходиться в стані q_1 і в комірці, що розглядається, знаходиться символ 1, то застосуємо команду $q_1 1 q_1 1$ Л. В

результаті вміст комірки, що розглядається, не зміниться, машина, як і раніше, буде знаходитися в стані q_1, але головка буде розглядати комірку справа (оскільки стрічка переміститься на одну комірку вліво відносно пристрою керування).

У комірці, що сприймається, знаходиться символ 0, значить, застосуємо команду $q_1 0 q_0 1$С. Міняємо вміст комірки на 1, машина переходить до стану q_0, стрічка не рухається.

Розв'язання даної задачі крок за кроком проілюстровано на рис. 1.77.

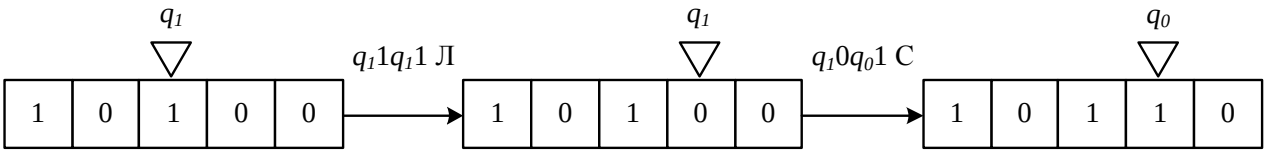


Рисунок 1.77 – Ілюстрація розв'язання задачі

У результаті роботи отримано слово 10110.

Задача 2

Нехай машина Тюрінга задана таким чином.

Зовнішній алфавіт: $A = \{S_0, a, b, c, d\}$, S_0 – пуста комірка.

Внутрішній алфавіт: $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$, q_5 – завершальний стан.

Сукупність команд:

$$q_0 a q_1 a \text{Л}$$
$$q_0 b q_0 b \text{Л}$$
$$q_2 a q_5 d \text{П}$$
$$q_0 c q_0 c \text{Л}$$
$$q_1 d q_2 c \text{П}$$
$$q_3 a q_4 d \text{П}$$
$$q_4 b q_2 c \text{П}.$$

Машина знаходиться в стані q_0. Необхідно застосувати дану машину для переробки вихідного слова $bcadc$.

Розв'язання

Запишемо програму даної машини Тюрінга у вигляді таблиці відповідності (табл. 1.4).

Таблиця 1.4 – Таблиця відповідності для заданої машини Тюрінга

Q \ A	S_0	a	b	c	d
q_0	–	$q_1 a$Л	$q_0 b$Л	$q_0 c$Л	–
q_1	–	–	–	–	$q_2 c$П
q_2	–	$q_5 d$П	–	–	–
q_3	–	$q_4 d$П	–	–	–
q_4	–	–	$q_2 c$П	–	–
q_5	Зупинка				

Розглянемо роботу заданої машини Тюрінга на слові $bcadc$. Вихідна конфігурація виглядає таким чином: $q_0 bcadc$. Визначаємо за таблицею відповідності наступну команду (на перетині рядка q_0 і стовпця b). Із табл. 1.4 випливає, що виконується команда $q_0 b q_0 b$Л. У результаті виконання даної команди символ в комірці, що розглядається, не змінюється, внутрішній стан машини також не змінюється, а стрічка зсувається вліво на одну комірку. Отримуємо наступну конфігурацію $bq_0 cadc$.

Визначаємо за таблицею відповідності наступну команду $q_0 c q_0 c$Л (на перетині рядка q_0 і стовпця c). У результаті виконання даної команди символ у комірці, що розглядається, не змінюється, внутрішній стан машини q_0 також не змінюється, а стрічка зсувається вліво на одну комірку. Отримуємо наступну конфігурацію $bcq_0 adc$.

Визначаємо за таблицею відповідності наступну команду $q_0 a q_1 a$Л (на перетині рядка q_0 і стовпця a). У результаті виконання даної команди символ у комірці, що розглядається, не змінюється, внутрішній стан машини змінюється на q_0, стрічка зсувається вліво на одну комірку. Отримуємо наступну конфігурацію $bcaq_1 dc$.

Визначаємо за таблицею відповідності наступну команду $q_1 d q_2 c$П (на перетині рядка q_1 і стовпця d). У результаті виконання даної команди символ d в комірці, що розглядається, змінюється на c, внутрішній стан машини q_1 змінюється на q_2, стрічка зсувається вправо на одну комірку. Отримуємо наступну конфігурацію $bcq_2 acc$.

Визначаємо за таблицею відповідності наступну команду $q_2 a q_5 d$П (на

перетині рядка q_{12} і стовпця a). У результаті виконання даної команди символ a в комірці, що розглядається, змінюється на d, стрічка зсувається вправо на одну комірку, а внутрішній стан машини змінюється на q_5, тобто машина переходить до завершального стану. Отримуємо наступну конфігурацію bq_5cdcc і вихідне слово $bcdcc$.

В даній задачі переробка вхідного слова $bcadc$ в слово $bcdcc$ була здійснена в результаті виконання послідовності команд: $q_0bq_0bЛ$, $q_0cq_0cЛ$, $q_0aq_1aЛ$, $q_1dq_2cП$, $q_2aq_5dП$.

Рисунок 1.78 ілюструє покрокову роботу заданої машини Тюрінга на слові $bcadc$.

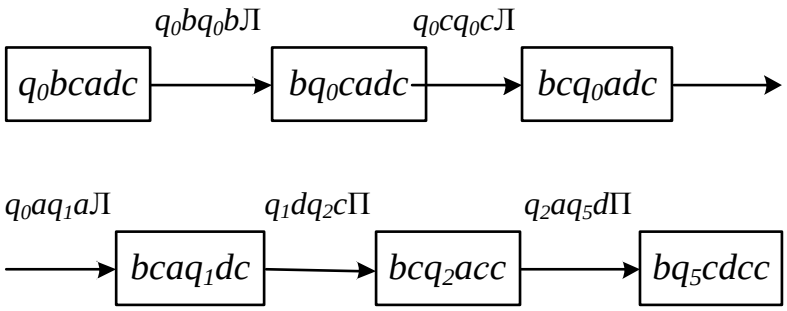


Рисунок 1.78 – Ілюстрація роботи машини Тюрінга на слові $bcadc$

Запитання для самоконтролю

1. Чим є машина Тюрінга?
2. Що таке програма машини Тюрінга? Як вона записується?
3. Які складові необхідно вказувати для задання машини Тюрінга?
4. Як задається таблиця переходів?

Задачі для аудиторних занять

1. На стрічці машини Тюрінга записані символи 0. Написати програму для машини Тюрінга, що замінює кожен другий символ на 1.

2. Скласти таблицю відповідності для машини Тюрінга, яка відшуковує першу після групи нулів групу одиниць справа від початкової комірки і зупиняється біля останньої з цих одиниць.

3. Скласти таблицю відповідності для машини Тюрінга, яка починаючи роботу із заповненої комірки, рухається вліво і зупиняється на дві комірки зліва від групи одиниць.

4. Машина Тюрінга задана наступною таблицею відповідності:

Q \ A	S_0	a	b	c	d
q_0	–	$q_0 a Л$	$q_1 d Л$	$q_0 c Л$	$q_2 d П$
q_1	–	–	–	$q_2 c Л$	$q_2 c П$
q_2	–	$q_4 d П$	–	$q_4 c П$	–
q_3	–	–	$q_2 c П$	–	–
q_4	Зупинка				

Початковий стан машини q_0. Необхідно застосувати дану машину для переробки вихідного слова $acabcd$.

1.6. Генератори псевдовипадкових чисел

1.6.1. Короткі теоретичні відомості

Криптографічні додатки використовують для генерації випадкових чисел спеціальні алгоритми. Якщо вибрати гарний алгоритм, то отримана числова послідовність пройде більшість тестів на випадковість. Такі числа називають псевдовипадковими числами.

Генератор псевдовипадкових чисел (ГПВЧ) – алгоритм, що породжує послідовність чисел, елементи якої майже незалежні один від одного та підпорядковуються заданому розподілу (зазвичай рівномірному).

Лінійний конгруентний метод є одним з найпростіших ГПВЧ, суть його полягає в обчисленні членів лінійної рекурентної послідовності за формулою

$$X_{k+1} = (aX_k + c)\ mod\ m,$$

де a і c – деякі цілочислові коефіцієнти,

m – деяке натуральне число,

X_k – попереднє псевдовипадкове число.

Отримана послідовність залежить від вибору стартового числа, і при різних його значеннях виходять різні послідовності випадкових чисел. Разом з тим багато властивостей цієї послідовності визначаються вибором коефіцієнтів у формулі і не залежать від вибору стартового числа. Для даних констант виписані умови, що гарантують задовільну якість випадкових чисел, що отримуються.

При реалізації алгоритму рекомендується вибирати m^e, де e – число бі-

тів в машинному слові, тому що це дозволяє позбутися відносно повільної операції приведення за модулем.

Один з прикладів вдалих параметрів методу:

$$m = 2^{32},\ a = 69069, c = 5.$$

Особливості розподілу випадкових чисел, що генеруються лінійним конгруентним алгоритмом, роблять неможливим їх використання в статистичних алгоритмах, які потребують високого дозволу.

Один з найбільш поширених датчиків Фібоначчі оснований на такій формулі:

$$X_k = \begin{cases} X_{k-a} - X_{k-b}, \ \textit{якщо}\ X_{k-a} \geq X_{k-b}, \\ X_{k-a} - X_{k-b} + 1, \ \textit{якщо}\ X_{k-a} < X_{k-b}, \end{cases}$$

де X_k – дійсні числа з діапазону $[0, 1)$;

a, b – цілі позитивні числа, які називаються лагами.

При реалізації через цілі числа досить формули $X_k = X_{k-a} - X_{k-b}$.

Для роботи датчика Фібоначчі потрібно знати попередні згенеровані випадкові числа. Для старту датчика Фібоначчі необхідно $\max(a,b)$ випадкових чисел, які можуть бути згенеровані простим конгруентним датчиком.

Найбільшу популярність датчики Фібоначчі отримали у зв'язку з тим, що швидкість виконання арифметичних операцій з дійсними числами зрівнялась зі швидкістю цілочислової арифметики, а фібоначчієві датчики природно реалізуються в арифметиці дійсних чисел.

Отримані випадкові числа мають гарні статистичні властивості, причому всі біти випадкового числа рівнозначні за статистичними властивостями.

Для вибору лагів a і b рекомендуються такі значення:

$$(a, b) = (55{,}24),\ (17{,}5) \text{ чи } (97{,}33).$$

Якість отримуваних випадкових чисел залежить також від значення констант.

Статистичні тести на випадковість дозволяють отримати числову характеристику послідовності і дати відповідь на питання, чи є випадковою послідовність.

Розглянемо деякі з тестів пакета NIST.

Тест 1. Частотний побітовий тест.

Суть даного тесту полягає у визначенні співвідношення між нулями і одиницями у всій двійковій послідовності. Мета – з'ясувати, чи дійсно число нулів і одиниць у послідовності приблизно однакове, як це можна було б припустити у випадку істинно випадкової бінарної послідовності. Всі наступні тести проводяться за умови, що даний тест пройдено.

Вхідні дані для тесту – послідовність бітів $\epsilon = \epsilon_1, \epsilon_2, \ldots, \epsilon_n$.

Тест обчислює статистику $S_{obs} = \frac{\sum_{i=1}^{n} X_i}{\sqrt{n}}$, $X_i = 2\epsilon_1 - 1$, тобто X_i може набувати значень $+1$ та -1. Розподіл даної статистики є напівнормальним для великих n (якщо величина $|z|$ розподілена як нормальна, то розподіл називається напівнормальним). Якщо послідовність випадкова, то плюс і мінус одиниці будуть прямувати до взаємного винищення, а підсумкова статистика прагне до нуля. Якщо ж все-таки є занадто багато одиниць чи занадто багато нулів у початковій послідовності, то статистика буде більшою від нуля.

Покрокове описання тесту.

Крок 1. Перетворити нулі та одиниці вихідної послідовності в -1 і $+1$ та скласти: $S_n = X_1 + X_2 + \cdots + X_n, X_i = 2\,\epsilon_i - 1$.

Крок 2. Обчислити статистику $S_{obs} = \frac{|S_n|}{\sqrt{n}}$.

Крок 3. Обчислити значення $p = erfc\left(\frac{S_{obs}}{\sqrt{2}}\right)$, де $erfc$ – це додаткова функція помилок. Вона визначається як $erfcx = 1 - erfx = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-t^2} dt$, де $erfx = \frac{2}{\sqrt{\pi}} \int_0^{x} e^{-t^2} dt$ – функція помилок. Наведені інтеграли неможливо взяти, тому для обчислення даних функцій використовуються розкладення до ряду. Значення додаткової функції помилок доступні у вигляді довідкових таблиць. Є готові реалізації даної функції в бібліотеках C++ (файл *cmath*, функції *erfcf( )* та *erfcl ( )*), Java (метод *Erf.erfc( )* в пакеті org.apache.commons.math3.special), Python (*math.erfc()*).

Крок 4. Інтерпретація результатів. Якщо $p < 0{,}01$, то послідовність вважається невипадковою.

Тест 2. Частотний блочний тест.

Суть тесту – визначення частки одиниць всередині блока довжиною m біт. Мета – з'ясувати, чи дійсно частота повторів одиниць у блоці довжиною M біт приблизно дорівнює $M/2$, як можна було б припустити у випадку абсолютно випадкової послідовності. Якщо взяти $M=1$, даний тест переходить у тест №1 (частотний побітовий тест).

Вхідні дані для тесту – послідовність бітів $\epsilon = \epsilon_1, \epsilon_2, \ldots, \epsilon_n$ і число M – довжина блока.

Тест обчислює статистику $\chi^2(obs)$, яка відображає, наскільки добре пропорція одиниць у даному блоці довжини M відповідає очікуваній пропорції $1/2$. Розподіл даної статистики – χ^2.

Покрокове описання тесту.

Крок 1. Розділити послідовність на $N = \left\lfloor \frac{n}{M} \right\rfloor$ блоків, які не перекриваються. Біти, які не використовуються, відкидаються.

Крок 2. Визначити пропорцію π_i одиниць у кожному блоці довжиною M біт, використовуючи формулу $\pi_i = \frac{\sum_{j=1}^{M} \in_{(i-1)M+j}}{M}$, $1 \le i \le N$.

Крок 3. Обчислити статистику $\chi^2(obs) = 4M \sum_{i=1}^{N} (\pi_i - 0{,}5)^2$.

Крок 4. Обчислити значення $p = Q\left(\frac{N}{2}, \frac{\chi^2(obs)}{2}\right)$, де Q (також позначається як $igamc$) – неповна верхня гамма-функція, яка визначається як $Q(a,x) = \frac{1}{\Gamma(a)} \int_x^\infty e^{-t} t^{a-1} dt$, де $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$ – стандартна гамма-функція. Значення даної функції доступні в таблицях, а також у бібліотеках C++ (файл boost/math/special_functions/gamma.hpp, функція *gamma_q*), Java (метод Gamma.regularizedGammaQ з пакета org.apache.commons.math3.special), Python (scipy.special.gammainc і mpmath.gammainc).

Крок 5. Інтерпретація результатів. Якщо $p < 0{,}01$, то послідовність вважається невипадковою.

3. Тест на послідовність однакових бітів.

Суть полягає в підрахунку повного числа рядів у вихідній послідовності, де під словом «ряд» розуміють безперервну підпослідовність однакових бітів. Ряд довжиною k біт складається з k абсолютно ідентичних бітів, починається і закінчується з біта, що містить протилежне значення. Мета даного тесту – зробити висновок про те, чи дійсно кількість рядів, що складаються з одиниць і нулів різної довжини, відповідає їх кількості у випадковій послідовності. Зокрема, визначається, швидко чи повільно чергуються одиниці і нулі у вихідній послідовності. Якщо обчислене в ході тесту значення ймовірності $p < 0{,}01$, то дана двійкова послідовність не є істинно випадковою. В протилежному випадку, вона має випадковий характер.

Вхідні дані для тесту – послідовність бітів $\epsilon = \epsilon_1, \epsilon_2, \ldots, \epsilon_n$.

Тест обчислює статистику $V_n(obs)$ – загальну кількість безперервних відрізків, що складаються лише з нулів чи лише одиниць. Розподіл даної статистики – χ^2.

Покрокове описання тесту.

Крок 1. Обчислити пропорцію одиниць у вихідній послідовності:

$$\pi = \frac{\sum_j \epsilon_j}{n}.$$

Крок 2. Визначити, чи виконана передумова $\left|\pi - \frac{1}{2}\right| < \tau;\ \tau = \frac{2}{\sqrt{n}}$. Якщо вона не виконана, то тест вже не пройдено, а підсумкове значення p кладеться рівним нулю.

Крок 3. Обчислити статистику $V_n(obs) = \sum_{k=1}^{n-1} r(k) + 1$, де $r(k) = 0$, якщо $\epsilon_k = \epsilon_{k+1}$, інакше $r(k) = 1$.

Крок 4. Обчислити $p = erfc\left(\frac{\left|V_n(obs) - 2n\pi(1-\pi)\right|}{2\sqrt{2n}\pi(1-\pi)}\right)$.

Крок 5. Інтерпретація результатів. Якщо $p < 0{,}01$, то послідовність вважається невипадковою.

4. Тест на найдовшу послідовність одиниць у блоці.

У даному тесті визначається найдовший ряд одиниць в середині блока довжиною M біт. Мета – з'ясувати, чи дійсно довжина такого ряду відповідає очікуванням довжини найдовшого ряду одиниць у випадку абсолютно випад-

кової послідовності. Якщо обчислене в ході тесту значення ймовірності $p < 0{,}01$, вважається, що вихідна послідовність не є випадковою. В протилежному випадку роблять висновок про її випадковість. Необхідно зазначити, що з припущення про приблизно однакову частоту появи одиниці і нулів (тест №1) випливає, що такі самі результати даного тесту будуть отримані при розгляді найдовшого ряду нулів. Тому вимірювання можна проводити лише з одиницями.

Вхідні дані для тесту:

– послідовність бітів $\epsilon = \epsilon_1, \epsilon_2, \dots, \epsilon_n$;

– довжина блока M;

– кількість блоків N.

Рекомендовані значення M залежно від довжини послідовності n наведені в табл. 1.5.

Таблиця 1.5 – Рекомендовані значення довжини блока M залежно від довжини послідовності n

Мінімальна довжина послідовності n	Довжина блока M
128	8
6272	128
750 000	10 000

Тест обчислює статистику $\chi^2(obs)$, яка відображає, наскільки спостережувана довжина найдовшої послідовності з одиниць у блоці довжиною M відповідає очікуваній. Розподіл даної статистики – χ^2.

Покрокове описання тесту.

Крок 1. Розділити послідовність на блоки довжиною M.

Крок 2. Розподілити за таблицею частоти v_i найдовших послідовностей з одиниць в кожному блоці, де кожна клітинка містить кількість послідовностей одиниць даної довжини (табл. 1.6).

Крок 3. Обчислити $\chi^2(obs) = \dfrac{\sum_{i=0}^{K}(v_i - N\pi_i)^2}{N\pi_i}$, де K та N знаходять з табл. 1.7.

Теоретичні ймовірності π_i задаються константами. Для $K = 3$, $M = 8$ необхідно взяти $\pi_0 = 0{,}2148$, $\pi_1 = 0{,}3672$, $\pi_2 = 0{,}2305$, $\pi_3 = 0{,}1875$.

Таблиця 1.6 – Визначення частоти v_i

v_i	$M = 8$	$M = 128$	$M = 10\,000$
v_0	≤ 1	≤ 4	≤ 10
v_1	2	5	11
v_2	3	6	12
v_3	≥ 4	7	13
v_4		8	14
v_5		≥ 9	15
v_6			≥ 16

Крок 4. Обчислити $p = Q\left(\frac{K}{2}, \frac{\chi^2(obs)}{2}\right)$.

Таблиця 1.7 – Визначення величин K та N залежно від M

M	K	N
8	3	16
128	5	49
10 000	6	75

Крок 5. Інтерпретація результатів. Якщо $p < 0{,}01$, то послідовність вважається невипадковою.

1.6.2. Приклади розв'язання задач

Задача 1

Використовуючи лінійний конгруентний метод, згенерувати послідовність з 10 чисел.

Розв'язання

Задамо коефіцієнти для обчислення елементів послідовності $a = 69069$, $c = 5$. Покладемо $m = 2^{32}$. Виберемо довільно стартове число послідовності $x_1 = 56473829$.

Використовуємо формулу лінійного конгруентного методу $X_{k+1} = (aX_k + c) \bmod m$:

$x_2 = (ax_1 + c)\ mod\ m = 760590438;$

$x_3 = (ax_2 + c)\ mod\ m = 1475964851;$

$x_4 = (ax_3 + c)\ mod\ m = 2367523164;$

$x_5 = (ax_4 + c)\ mod\ m = 167553713;$
$x_6 = (ax_5 + c)\ mod\ m = 2125507778;$
$x_7 = (ax_6 + c)\ mod\ m = 419574111;$
$x_8 = (ax_7 + c)\ mod\ m = 1419926552;$
$x_9 = (ax_8 + c)\ mod\ m = 1623783229;$
$x_{10} = (ax_9 + c)\ mod\ m = 2897810654.$

Задача 2

Використовуючи метод Фібоначчі із запізненнями, згенерувати послідовність з 10 чисел.

Розв'язання

Задамо лаги $(a, b) = (5,2)$і початкові елементи $(x_0, x_1, x_2, x_3, x_4) = (0{,}2;\ 0{,}5;\ 0{,}1;\ 0{,}8;\ 0{,}7)$. Використовуємо формулу методу Фібоначчі із запізненнями $X_k = \begin{cases} X_{k-a} - X_{k-b},\ якщо\ X_{k-a} \geq X_{k-b}, \\ X_{k-a} - X_{k-b} + 1,\ якщо\ X_{k-a} < X_{k-b}, \end{cases}$:

$x_5 = x_0 - x_3 + 1 = 0{,}4;$
$x_6 = x_1 - x_4 + 1 = 0{,}8;$
$x_7 = x_2 - x_5 + 1 = 0{,}7;$
$x_8 = x_3 - x_6 = 0;$
$x_9 = x_4 - x_7 + 1 = 1{,}0;$
$x_{10} = x_5 - x_8 = 0{,}4;$
$x_{11} = x_6 - x_9 + 1 = 0{,}8;$
$x_{12} = x_7 - x_{10} = 0{,}3;$
$x_{13} = x_8 - x_{11} + 1 = 0{,}2;$
$x_{14} = x_9 - x_{12} = 0{,}7;$
$x_{15} = x_{10} - x_{13} = 0{,}2.$

Задача 3

Провести перевірку випадковості послідовності $\epsilon = 0100110010$, використовуючи тест на однакові біти, що йдуть підряд.

Розв'язання

Крок 1. Перетворимо нулі та одиниці вихідної послідовності в -1 і $+1$ та складемо:

$S_n = X_1 + X_2 + \cdots + X_n,\ X_i = 2\epsilon_i - 1,$

$S_n = -1 + 1 - 1 - 1 + 1 + 1 - 1 - 1 + 1 - 1 = -2.$

Крок 2. Обчислимо статистику $S_{obs} = \frac{|S_n|}{\sqrt{n}}$:

$$S_{obs} = \frac{|-2|}{\sqrt{10}} = 0{,}632455.$$

Крок 3. Обчислимо значення $p = erfc\left(\frac{S_{obs}}{\sqrt{2}}\right)$:

$$p = erfc\left(\frac{0{,}632455}{\sqrt{2}}\right) = erfc\ 0{,}447213 = 0{,}5270896.$$

Крок 4. Оскільки $p \geq 0{,}01$, то послідовність визнається випадковою.

Задача 4

Провести перевірку випадковості послідовності $\epsilon = 0100110010$, використовуючи частотний блочний тест. Довжина блока $M = 3$.

Розв'язання

Крок 1. Розділимо послідовність на $N = \left\lfloor \frac{n}{M} \right\rfloor = \left\lfloor \frac{10}{3} \right\rfloor = 3$ блоків, що не перекриваються: 010, 011, 001. Останній біт 0, що не використовується, відкидається.

Крок 2. Визначимо пропорцію π_i одиниць у кожному блоці довжиною M біт, використовуючи формулу $\pi_i = \frac{\sum_{j=1}^{M} \in_{(i-1)M+j}}{M}$, $1 \leq i \leq N$:

$\pi_1 = \frac{1}{3}, \pi_2 = \frac{2}{3}, \pi_3 = \frac{1}{3}.$

Крок 3. Обчислимо статистику $\chi^2(obs) = 4M\sum_{i=1}^{N}(\pi_i - 0{,}5)^2$:

$\chi^2(obs) = 4 \cdot 3\left(\left(\frac{1}{3} - 0{,}5\right)^2 + \left(\frac{2}{3} - 0{,}5\right)^2 + \left(\frac{1}{3} - 0{,}5\right)^2\right) = 12(0{,}0278 + +0{,}0278 + 0{,}0278) = 1.$

Крок 4. Обчислимо значення:

$p = Q\left(\frac{N}{2}, \frac{\chi^2(obs)}{2}\right) = Q\left(\frac{3}{2}, \frac{1}{2}\right) = 0{,}801252.$

Крок 5. Оскільки $p \geq 0{,}01$, то послідовність вважається випадковою.

Задача 5

Провести перевірку випадковості послідовності $\epsilon = 0100110010$, використовуючи тест на послідовність однакових бітів.

Розв'язання

Крок 1. Обчислимо пропорцію одиниць у вихідній послідовності:

$$\pi_i = \frac{\sum_j \in_j}{M}, = \frac{4}{10} = 0,4.$$

Крок 2. Перевіримо передумову $\left|\pi - \frac{1}{2}\right| < \tau;\ \tau = \frac{2}{\sqrt{n}}$:

$\left|\pi - \frac{1}{2}\right| = |0,4 - 0,5| = 0,1;$

$\tau = \frac{2}{\sqrt{n}} = \frac{2}{\sqrt{10}} = 0,632455;$

$0,1 < 0,632455.$

Умову виконано, тому тест продовжується.

Крок 3. Обчислимо статистику $V_n(obs) = \sum_{k=1}^{n-1} r(k) + 1,\ \ r(k) = 0,$ якщо $\epsilon_k = \epsilon_{k+1}$, інакше $r(k) = 1.$

$\epsilon = 0100110010$

$V_n(obs) = (1 + 1 + 0 + 1 + 0 + 1 + 0 + 1 + 1) + 1 = 6 + 1 = 7.$

Крок 4. Обчислимо:

$p = erfc\left(\frac{|V_n(obs) - 2n\pi(1-\pi)|}{2\sqrt{2n}\pi(1-\pi)}\right) = erfc\left(\frac{|7 - 2\cdot 10\cdot 0,4(1-0,4)|}{2\sqrt{2\cdot 10}\cdot 0,4(1-0,4)}\right) = erfc\left(\frac{|7-4,8|}{2,1466}\right) =$

$= erfc(1,0249) = 0,1472.$

Крок 5. Оскільки $p \geq 0,01$, то послідовність вважається випадковою.

Задача 6

Провести перевірку випадковості послідовності, використовуючи тест на найдовшу послідовність одиниць у блоці:

11001100 00010101 01101100 01001100 11100000 00000010
01001101 01010001 00010011 11010110 10000000 11010111
11001100 11100110 11011000 10110010.

Розв'язання

Крок 1. Розділяємо послідовність на блоки довжиною $M = 8$, тому що довжина послідовності дорівнює 128.

Крок 2. Розподіляємо за таблицею частоти v_i найдовших послідовностей з одиниць у кожному блоці, де кожна клітинка містить кількість послідовнос-

тей одиниць даної довжини.

Блок	Довжина одиниць	Блок	Довжина одиниць
11001100	2	00010011	2
00010101	1	11010110	2
01101100	2	10000000	1
01001100	2	11010111	3
11100000	3	11001100	2
00000010	1	11100110	3
01001101	2	11011000	2
01010001	1	10110010	2

$v_0 = \{$ кількість блоків з максимальною довжиною $\leq 1\} = 4;$
$v_1 = \{$ кількість блоків з максимальною довжиною $= 2\} = 9;$
$v_2 = \{$ кількість блоків з максимальною довжиною $= 3\} = 3;$
$v_3 = \{$ кількість блоків з максимальною довжиною $\geq 4\} = 0.$

Крок 3. Обчислимо $\chi^2(obs) = \dfrac{\sum_{i=0}^{K}(v_i - N\pi_i)^2}{N\pi_i}$:

$$\chi^2(obs) = \frac{(4-16\cdot 0{,}2148)^2}{16\cdot 0{,}2148} + \frac{(9-16\cdot 0{,}3672)^2}{16\cdot 0{,}3672} + \frac{(3-16\cdot 0{,}2305)^2}{16\cdot 0{,}2305} + \frac{(0-16\cdot 0{,}1875)^2}{16\cdot 0{,}1875} = 4{,}8826.$$

Крок 4. Обчислимо $p = Q\left(\frac{K}{2}, \frac{\chi^2(obs)}{2}\right) = Q\left(\frac{3}{2}, \frac{4{,}8826}{2}\right) = 0{,}1860.$

Крок 5. Оскільки $p \geq 0{,}01$, то послідовність вважається випадковою.

Задачі для аудиторних занять

Дані завдання передбачається виконувати, не використовуючи комп'ютер, тому їх розмірність є невеликою і не потрібно обчислювати значення функцій $erfc$ і Q.

1. Визначте послідовність з перших десяти чисел, використовуючи лінійний конгруентний метод для різних параметрів a, c та x_0 (при $m = 32$):

1) $a = 2, c = 6, x_0 = 16;$
2) $a = 3, c = 1, x_0 = 25;$
3) $a = 4, c = 3, x_0 = 31;$
4) $a = 5, c = 6, x_0 = 13;$
5) $a = 7, c = 9, x_0 = 27;$

6) $a = 9, c = 11, x_0 = 19$;

7) $a = 10, c = 3, x_0 = 23$;

8) $a = 11, c = 8, x_0 = 7$.

2. Обчисліть послідовність з десяти чисел, що генерується методом Фібоначчі із запізненням, використовуючи такі вихідні дані: пару (a, b) і початкові значення $x_0, \dots, x_k$:

1) (5; 2); (0,29; 0,57; 0,96; 0,90; 0,91);

2) (5;2), (0,07; 0,42; 0,04; 0,88; 0,66);

3) (5;3); (0,13; 0,30; 0,86; 0,63; 0,84);

4) (5;3); (0,13; 0,46; 0,17; 0,62; 0,07);

5) (7;2); (0,21; 0,68; 0,43; 0,75; 0,38; 0,78; 0,61);

6) (7;3); (0,84; 0,41; 0,74; 0,22; 0,05; 0,42; 0,61);

7) (7;4); (0,22; 0,33; 0,88; 0,29; 0,93; 0,50; 0,58);

8) (7;5); (0,46; 0,69; 0,75; 0,59; 0,04; 0,11; 0,36).

3. Обчисліть статистику для перевірки на випадковість заданих послідовностей чисел:

а) частотний побітовий тест – статистика S_{obs};

б) частотний блочний тест (довжина блока $M = 3$) – статистика $\chi^2(obs)$;

в) тест на послідовність однакових бітів – статистика $V_n(obs)$.

Послідовності:

1) 0 0 0 1 1 1 0 0 0 1;

2) 0 1 0 1 0 1 1 0 0 0;

3) 0 1 1 1 1 1 0 0 0 1;

4) 1 0 0 0 1 1 0 0 0 0;

5) 0 0 1 0 1 0 0 1 0 0;

6) 1 1 1 0 1 1 1 0 0 1;

7) 1 0 1 1 0 1 1 1 1 0;

8) 0 1 1 0 1 0 0 1 0 0.

4. Обчисліть статистику для тесту на найдовшу послідовність одиниць в блоці. Довжина кожної вихідної послідовності 128 біт.

1) 00110011 00100000 11001010 01000000 11100101 10000101 10110110 00100110 10010101 10011000 11010010 00101000 10011000 00011101 01001010

00010011;

2) 10100011 11011000 10010110 10000000 10001001 10110010 00010111 10000101 10000111 11000100 11111011 10110100 00101100 10110001 00001000 10000111;

3) 00111111 01001111 10110000 10101001 10111110 11101000 10110001 00011011 00100101 01101001 01101001 00001100 11111100 00100111 00101111 11111000;

4) 10100001 00100000 01000011 01010010 01000101 00000101 11100111 11100011 00010111 10011100 10100110 00010111 01000011 00100100 10100101 10100000;

5) 11110001 11010101 01011101 11111110 01010111 00111001 00000010 10100101 10010111 00001101 11100011 11100000 00000000 01010100 10011100 11111000;

6) 01011110 01110001 11111011 10000110 11110110 00000010 01011110 00010110 00000011 01101101 11000000 00001110 01001000 00110000 01001111 10010000;

7) 10101111 10001111 00001011 11101111 10011100 10011101 01101100 01110010 11100001 10110100 10111110 00010001 11100111 00111111 01100010 01011100;

8) 11000000 10001100 11100001 11001010 10001000 00100100 10111010 11111010 01111101 10101000 01000111 01011011 00010010 10111011 00111111 10111110.

Запитання для самоконтролю

1. Що таке генератор псевдовипадкових чисел?
2. Перелічіть недоліки простих арифметичних генераторів.
3. Назвіть найпоширеніші генератори псевдовипадкових чисел.
4. У чому полягає суть лінійного конгруентного методу?
5. Наведіть ітеративну формулу, на якій базується один із поширених датчиків Фібоначчі.
6. Назвіть дві основні групи, на які поділяються тести на випадковість.
7. У чому полягає відмінність статичних тестів на випадковість від графічних?

1.7. Фундаментальні алгоритми на графах і деревах

1.7.1. Короткі теоретичні відомості

Граф – це сукупність непустої множини вершин і множини зв'язків між вершинами (ребер). Графи можуть бути орієнтованим та неорієнтованими.

Орієнтований граф визначається як пара $G = (V, E)$, де V – непуста скінченна множина вершин, а E – множина впорядкованих пар різних вершин, що називаються дугами чи орієнтованими ребрами. E – бінарне відношення до V, тобто підмножина множини $(V \times V)$.

На рис. 1.79 показаний орієнтований граф $G = (V, E)$ з 6-ма вершинами і 9-ма ребрами:

$V = \{1,2,3,4,5,6\}$; $E = \{(1,4), (1,5), (1,1), (4,5), (4,4), (5,3), (3,5), (5,6), (3,6)\}$.

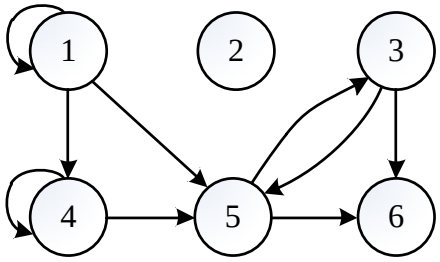


Рисунок 1.79 – Орієнтований граф $G = (V, E)$

У неорієнтованому графі $G = (V, E)$ множина ребер E складається із невпорядкованих пар вершин. Позначають неорієнтоване ребро як (u, v), при цьому для неорієнтованого графа (u, v) і (v, u) позначають одне і те саме ребро. На рис. 1.80 показаний неорієнтований граф з 6-ма вершинами $V = \{u, v, w, x, y, z\}$.

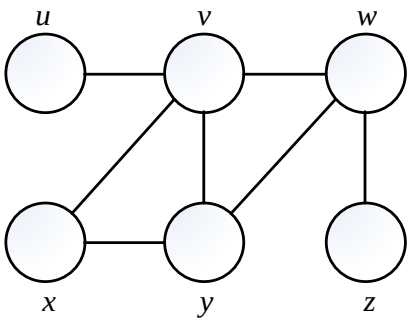


Рисунок 1.80 – Неорієнтований граф

Про ребро (u, v) орієнтованого графа говорять, що воно виходить з вершини u і входить у вершину v.

На рис. 1.79 два ребра, що виходять з вершини 5 – {5,3}, {5,6}, і три, що входять до неї – {1,5}, {4,5} і {3,5}.

Про ребро (u, v) неорієнтованого графа говорять, що воно є інцидентним вершинам u і v (а вершини називають граничними вершинами для ребра).

Ребра називаються суміжними, якщо вони відрізняються та мають спільну граничну вершину.

Вершина називається ізольованою, якщо немає ребер, для яких вона є граничною. На рис. 1.79 вершина 2 – ізольована.

Якщо на графі G є ребро (u, v), говорять, що вершина v суміжна з вершиною u. Для неорієнтованих графів відношення суміжності є симетричним, а для орієнтованих графів це не обов'язково.

На рис. 1.80 вершина u суміжна з вершиною v, а вершина v суміжна з вершиною u.

На рис. 1.79 вершина 5 орієнтованого графа суміжна з вершиною 3, а вершина 3 суміжна з вершиною 5, але вершина 6 суміжна з вершиною 3 (оскільки $(3,6) \in E$), а вершина 3 не суміжна з вершиною 6 (оскільки $(6,3) \notin E$).

Нехай довжина k з вершини u до вершини v визначається як послідовність вершин $\langle v_0, v_1, v_2, \dots, v_k, \rangle$, в якій $v_0 = u$, а $v_k = v$ і $(v_{i-1}, v_i) \in E$ для всіх $i = 1,2, \dots, k$. Таким чином, шлях довжини k складається із k ребер. Цей шлях містить вершини $\langle v_0, v_1, v_2, \dots, v_k, \rangle$ і ребра (v_0, v_1), (v_1, v_2), … , (v_{k-1}, v_k). Вершину v_0 називають початком шляху, вершину v_k його кінцем; говорять, що шлях веде з v_0 до v_k. Якщо для даних вершин u і v існує шлях p із v до u, то говорять, що вершина u досяжна з v по шляху p.

Шлях називається простим, якщо всі вершини в ньому відрізняються одна від одної.

Циклом в орієнтованому графі називається шлях, в якому початкова вершина збігається з кінцевою і який містить хоча б одне ребро.

Граф, в якому немає циклів, називається ациклічним.

Орієнтований граф називається сильно зв'язаним, якщо з будь-якої його вершини досяжна (по орієнтованим шляхам) будь-яка інша.

Дерево – це зв’язаний ациклічний граф. Зв’язність означає наявність шляхів між будь-якою парою вершин, ациклічність – відсутність циклів і те, що між парами вершин є лише один шлях.

Графи застосовуються для моделювання та описання різних систем і процесів, як наприклад, для транспортних маршрутів, відношення об’єктів, потоків вантажів і повідомлень. Для розв’язання різноманітних задач на графах існує велика кількість алгоритмів – це і пошук вершин на графі, і знаходження найкоротших шляхів, побудова мінімального кістякового дерева і т.д.

Розглянемо два базових алгоритми обходу вершин графа, які становлять основу багатьох інших – це пошук у ширину і глибину. Назва цих алгоритмів відображає напрям пошуку, відповідно «в ширину» і «в глибину», які просто реалізуються за допомогою рекурсивних процедур. Дані алгоритми застосовуються як для орієнтованих, так і неорієнтованих графів.

Основна ідея алгоритму пошуку в ширину на графі G – перелічити всі досяжні вершини із вказаної початкової в порядку зростання відстані.

Відстанню між вершинами вважається довжина мінімального шляху (кількість ребер) [1].

У процесі виконання пошуку в ширину вершини графа розфарбовуються в різні кольори та отримують мітку відстані. Спочатку всі вершини білі. Коли вершина вперше виявлена, вона зафарбовується в сірий колір. Вершина стає чорною, коли виявлені всі суміжні з нею вершини.

Алгоритм починається із заданої вершини графа, що розглядається. Очевидно, що цій вершині приписується мітка відстані, яка дорівнює нулю. Вершина зафарбовується в сірий колір. Далі розглядаємо всі суміжні з нею вершини, зафарбовуємо кожну з них у сірий колір і приписуємо мітку 2. Першу вершину зафарбовуємо в чорний колір. Послідовно повторюємо для кожної вершини з міткою 2 процес виявлення білих суміжних вершин. Зафарбовуємо ці виявлені вершини в сірий колір, надаємо їм мітку на одиницю більшу від попередньої. Вершину, у якої всі суміжні вершини стали сірими, зафарбовуємо в чорний колір. Процес повторюється доти, поки всі вершини не стануть чорними.

Алгоритм пошуку в глибину дозволяє побудувати обхід графа, в якому відвідуються всі вершини, доступні з початкової.

У процесі виконання пошуку в глибину вершини графа також зафарбо-

вуються в різні кольори, що свідчать про їх стан:

– спочатку всі вершини білі;

– вершина зафарбовується в сірий колір при її виявленні;

– вершина стає чорною, коли вона повністю оброблена, тобто виявлені (оброблені) всі суміжні з нею вершини (вершина не має білих суміжних з нею вершин).

Кожна вершина має також дві мітки часу:

– перша мітка вказує, коли вершина виявляється (і зафарбовується в сірий колір);

– друга мітка вказує, коли був завершений перегляд списку суміжності даної вершини і вона стала чорною.

Обхід у глибину починається з фіксованої вершини графа. Зафарбовуємо дану вершину в сірий колір і ставимо першу мітку часу, яка дорівнює 1. Далі йдемо вздовж ребер графа (зафарбовуючи відкриті вершини в сірий колір і ставлячи мітки часу їх відкриття), поки не дістанемося до тупикової вершини. Вершина буде вважатися тупиковою, якщо ми вже відкрили всі суміжні з нею вершини.

Після попадання в тупик повертаємося назад вздовж пройденого шляху, поки не виявимо вершину, у якої ще є невиявлені суміжні вершини, а потім рухаємось в цьому новому напрямку. Якщо вершина повністю оброблена (тобто всі суміжні з нею вершини сірі), вона зафарбовується в чорний колір, і ставиться її друга мітка часу.

Процес завершується, коли ми повернулись у початкову точку, а всі суміжні з нею вершини вже виявлені.

Якщо при цьому залишаються невиявлені вершини, то одна з них вибирається як нова вихідна вершина і пошук повторюється з неї [1].

Задача про топологічне сортування формулюється таким чином: розташувати вершини заданого орієнтованого графа без циклів на горизонтальній прямій так, щоб усі ребра йшли зліва направо.

Існує декілька алгоритмів топологічного сортування, наприклад алгоритм Кана, алгоритм Тар’яна, алгоритм Демукрона.

Одним з найпростіших алгоритмів розв’язання поставленої задачі є алгоритм на основі пошуку в глибину. Починаємо пошук з будь-якої вершини, яка не має ребер, що входять до неї. Здійснюємо обхід в глибину за описаним вище алгоритмом, і, коли вершина повністю оброблена (стала чорною), зано-

симо її номер у стек. Після закінчення обходу в глибину номера вершин дістають зі стека. Порядок розташування вершин на горизонтальній прямій відповідає порядку їх добування зі стека.

Для детальнішого ознайомлення з алгоритмами на графах і деревах рекомендується прочитати відповідні розділи робіт [1, 2, 11, 12].

1.7.2. Приклади розв'язання задач

Задача 1

Нехай задано орієнтований граф $G = (V, E)$, в якому виділена вихідна вершина №3 (рис. 1.81). Необхідно знайти довжини найкоротших шляхів (якщо такі існують) від заданої вершини до всіх інших, використовуючи алгоритм пошуку в ширину.

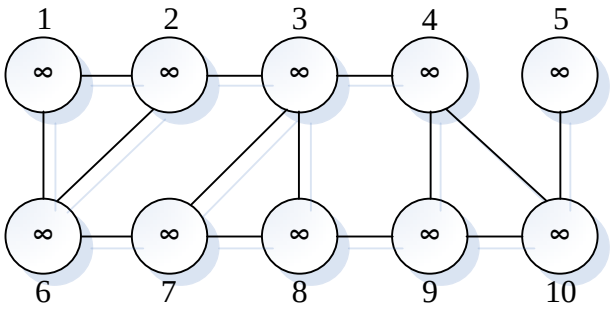


Рисунок 1.81 – Вихідний граф $G = (V, E)$

Розв'язання

Спочатку всі вершини білі і в них записані відстані, що дорівнюють ∞. Починаємо пошук у ширину із заданої вершини №3, записуємо цифру 0 у цю вершину та зафарбовуємо її в сірий колір (рис. 1.82).

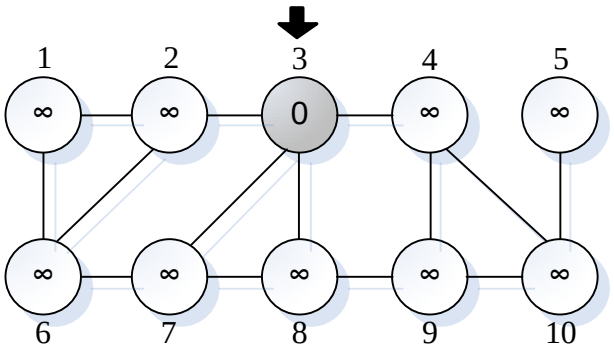


Рисунок 1.82 – Зафарбовування вершини №3 в сірий колір

Створюємо чергу з білих вершин, суміжних з вершиною №3. Це вершини №2, №4, №7 та №8. По черзі в ці вершини записуємо 1, оскільки між кожною з них і початковою вершиною є лише по одному ребру. Зафарбовуємо вершини №2, №4, №7 та №8 в сірий колір. Оскільки колір усіх вершин, суміжних з вершиною №3, став сірим, то зафарбовуємо її в чорний (рис. 1.83).

Вибираємо наступну за чергою сіру вершину. Це вершина №2, в суміжні з нею білі вершини №1 та №6 записуємо цифру 2 (тобто на одиницю більше, ніж у вершині №2), зафарбовуємо їх у сірий колір і додаємо до черги сірих вершин.

На даному етапі черга містить вершини №2, №4, №7, №8, №1 та №6.

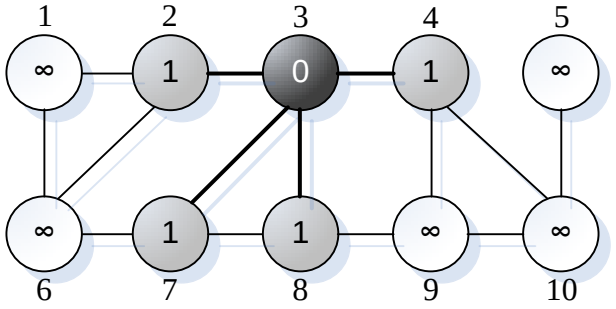


Рисунок 1.83 – Виявлені вершини №2, №4, №7 і №8, вершина №3 – чорна

Вершину №2 зафарбовуємо в чорний колір (рис. 1.84) і виключаємо з черги.

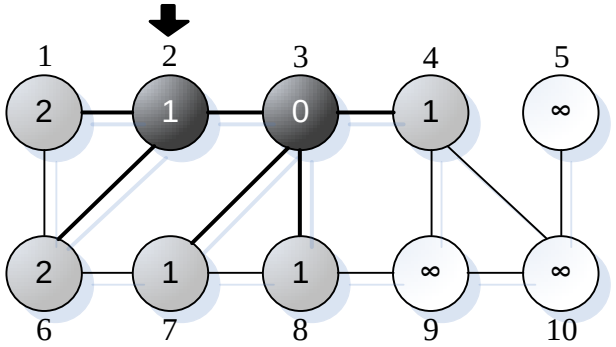


Рисунок 1.84 – Виявлені вершини №1 і №6, вершина №2 – чорна

Розглянемо наступну в черзі вершину. Це вершина №4. В суміжні з нею вершини №9 і №10 запишемо 2 (це на одиницю більше, ніж у вершині №4), зафарбовуємо їх в сірий колір і додаємо до черги. На даному етапі черга міс-

тить вершини №4, №7, №8, №1, №6, №9 та №10.

Вершину №4 зафарбовуємо в чорний колір (рис. 1.85) та виключаємо з черги.

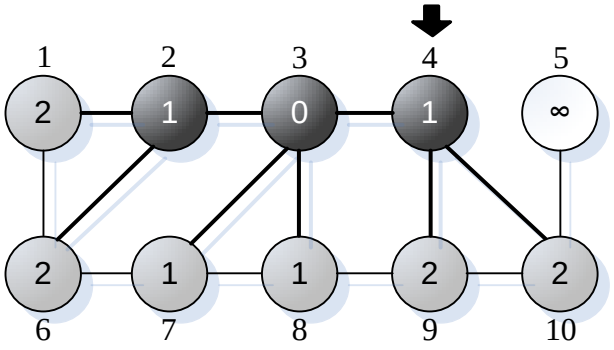


Рисунок 1.85 – Виявлені вершини №9 і №10, вершина №4 – чорна

Переходимо до наступної вершини в черзі. Це вершина №7. Вона не має білих суміжних вершин, тому зафарбовується в чорний колір та виключається з черги.

Далі розглянемо наступну в черзі вершину – вершину №8. Вона також не має білих суміжних вершин, тому зафарбовується в чорний колір та виключається з черги (рис. 1.86).

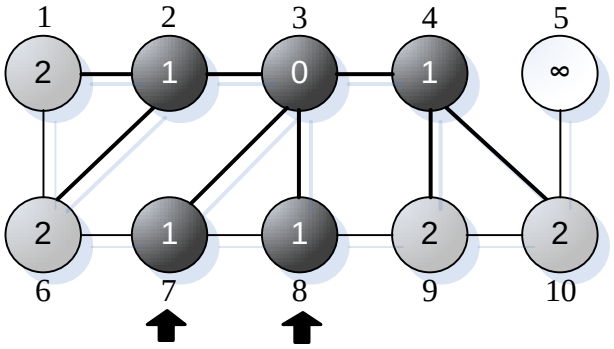


Рисунок 1.86 – Зафарбовування вершин №7 і №8 в чорний колір

До черги на даному етапі входять такі сірі вершини: №1, №6, №9, №10.

Розглянемо вершину №1. Вона не має білих суміжних вершин, тому зафарбовується в чорний колір та виключається з черги.

Аналогічно зафарбовуємо в чорний колір і виключаємо вершини №6 та №9, оскільки вони не мають білих суміжних вершин (рис. 1.87).

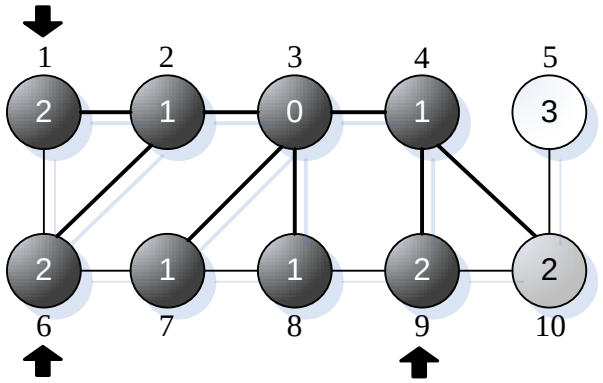


Рисунок 1.87 – Зафарбовування вершин №1, №6, №9 в чорний колір

У черзі залишилась одна вершина – вершина №10. Вона має білу суміжну вершину №5. Записуємо у вершину №5 цифру 3, зафарбовуємо її в сірий колір та додаємо до черги.

Вершину №10 зафарбовуємо в чорний і виключаємо з черги (рис. 1.88).

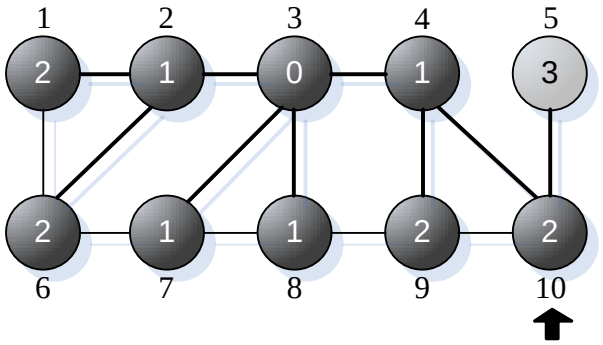


Рисунок 1.88 – Виявлена вершина №5, вершина №10 – чорна

У черзі – вершина №5, вона не має білих суміжних вершин, зафарбовуємо її в чорний колір и виключаємо з черги.

Черга пуста, пошук у ширину закінчено. Результат наведено на рисунку 1.89.

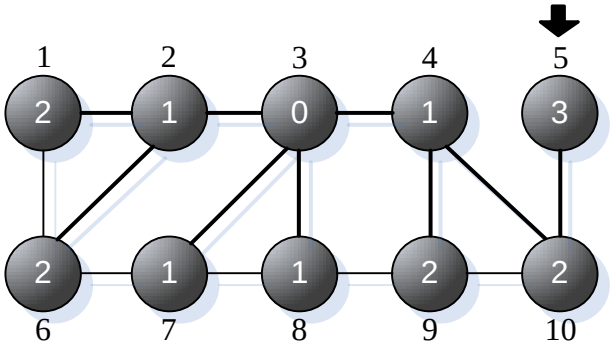


Рисунок 1.89 – Результат пошуку в ширину на графі $G = (V, E)$ з початковою вер-

шиною №3

Задача 2

На рис. 1.90 заданий орієнтований граф $G = (V, E)$. Необхідно здійснити пошук у глибину на даному графі з вершини z.

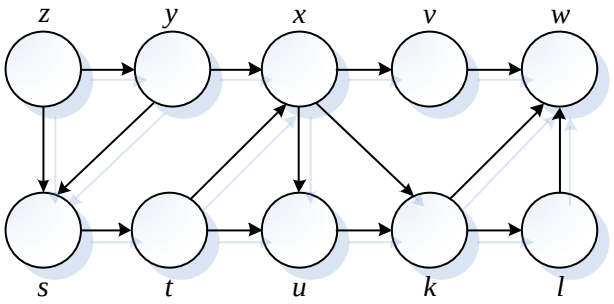


Рисунок 1.90 – Вихідний граф $G = (V, E)$

Розв'язання

Починаємо рухатися графом G з початкової вершини z, ставимо першу мітку часу, яка дорівнює 1 в даній вершині, та зафарбовуємо її в сірий колір (рис. 1.91).

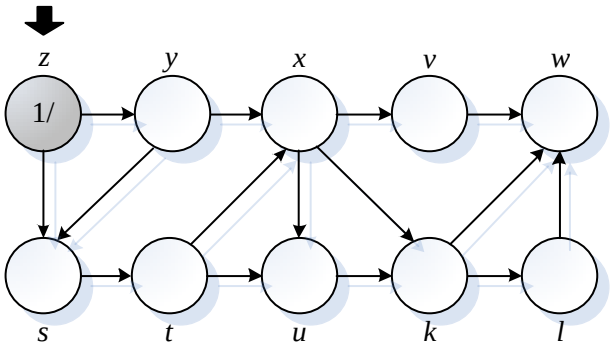


Рисунок 1.91 – Виявлена вершина z

Знаходимо суміжну із z вершину (рис. 1.92).

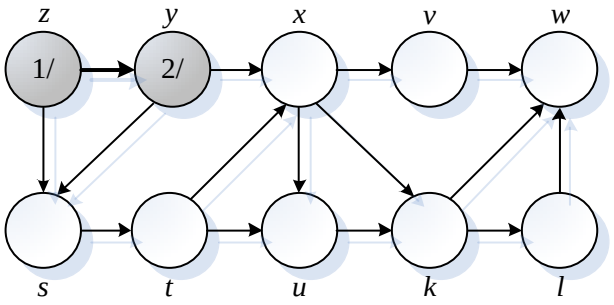


Рисунок 1.92 – Виявлена вершина y

Це вершина y, ставимо першу мітку часу (мітку її виявлення), яка дорівнює 2, тобто до мітки часу вершини z додаємо одиницю. Зафарбовуємо вершину y в сірий колір.

Продовжуємо рухатися «вглибину». Вибираємо вершину x, що є суміжною з y, мітка виявлення вершини дорівнює 3, зафарбовуємо x у сірий колір (рис. 1.93).

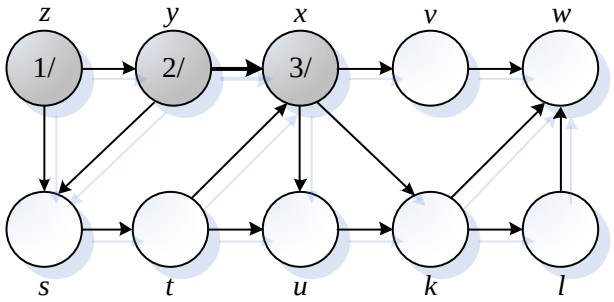


Рисунок 1.93 – Виявлена вершина x

Наступні виявлені вершини – v (рис. 1.94), потім w (рис. 1.95).

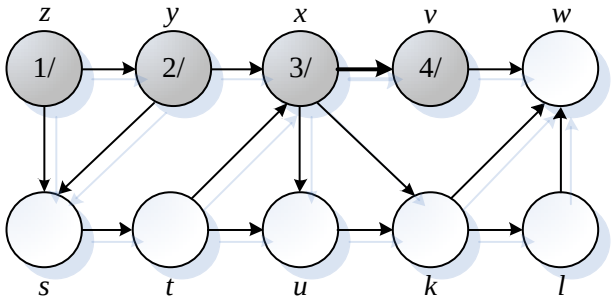


Рисунок 1.94 – Виявлена вершина v

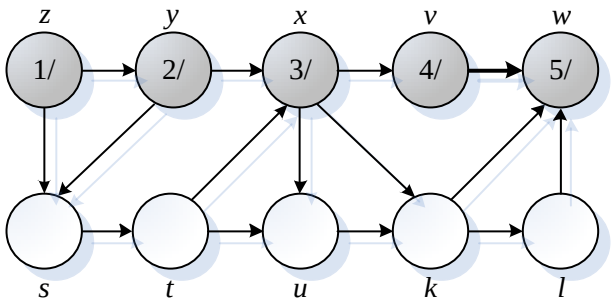


Рисунок 1.95 – Виявлена вершина w

Оскільки вершина w не має суміжних білих вершин, зафарбовуємо її в чорний колір і ставимо другу мітку часу,), яка дорівнює 6 (рис. 1.96).

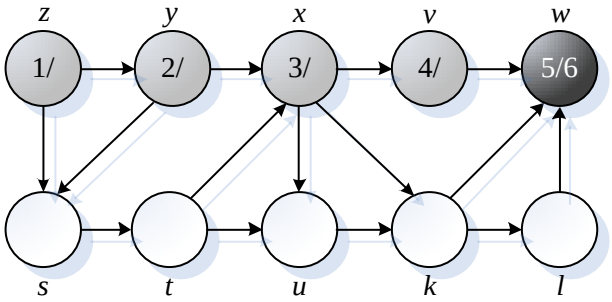


Рисунок 1.96 – Зафарбовування вершини w в чорний колір

Далі «повертаємося» до вершини v, дана вершина також не має білих суміжних, значить зафарбовуємо її в чорний колір і ставимо другу мітку часу), яка дорівнює 7 (рис. 1.97).

Повертаємося до попередньої сірої вершини x (рис. 1.98).

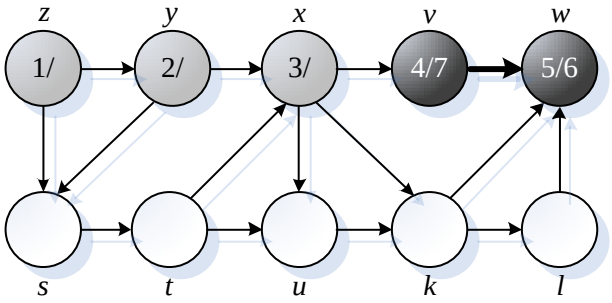


Рисунок 1.97 – Зафарбовування вершини v в чорний колір

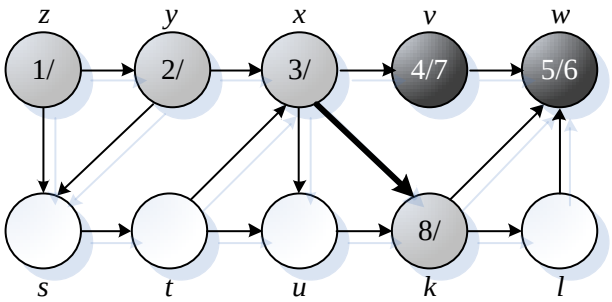


Рисунок 1.98 – Виявлена вершина k

Дана вершина має суміжні білі вершини, вибираємо вершину k, зафарбовуємо її в сірий колір і ставимо першу мітку), яка дорівнює 8.

Вершина k має білу суміжну вершину – це вершина l, зафарбовуємо її в сірий колір і ставимо мітку часу її виявлення – 9 (рис. 1.99).

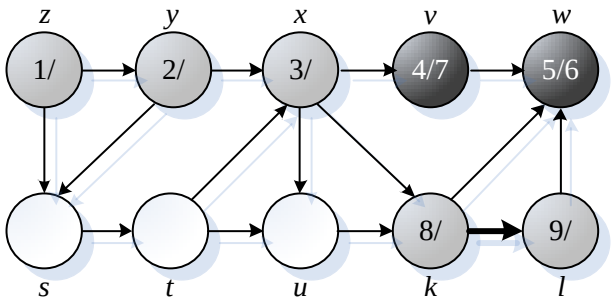


Рисунок 1.99 – Виявлена вершина l

Вершина l не має білих суміжних вершин, тому зафарбовуємо її в чорний колір, ставимо другу мітку часу), яка дорівнює 10 (рис. 1.100), і переходимо в попередню сіру вершину – вершину k.

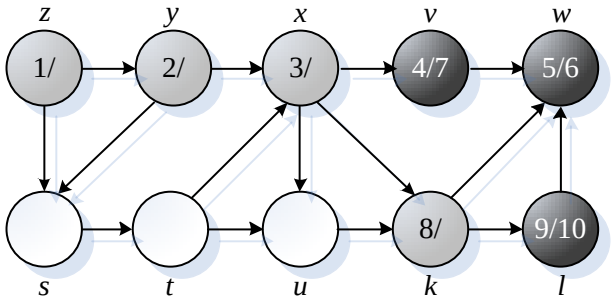


Рисунок 1.100 – Зафарбовування вершини l в чорний колір

Вершина k не має білих суміжних вершин, тому зафарбовуємо її в чорний колір, ставимо другу мітку часу), яка дорівнює 11 (рис. 1.101), і переходимо до попередньої сірої вершини.

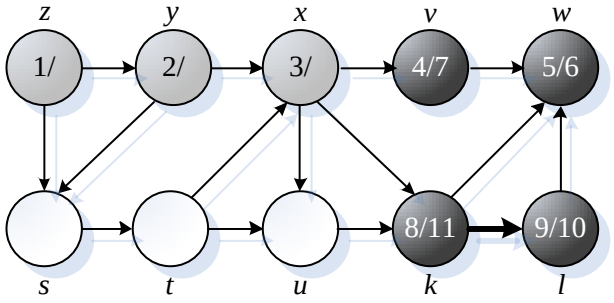


Рисунок 1.101 – Зафарбовування вершини k в чорний колір

Вершина x має білу суміжну вершину, тому переходимо до даної вершини u, зафарбовуємо її в сірий колір і ставимо першу мітку), яка дорівнює 12 (рис. 1.102).

Вершина u не має білих суміжних вершин, тому зафарбовуємо її в чорний колір, ставимо другу мітку часу), яка дорівнює 13, та переходимо до попередньої сірої вершини x (рис. 1.103).

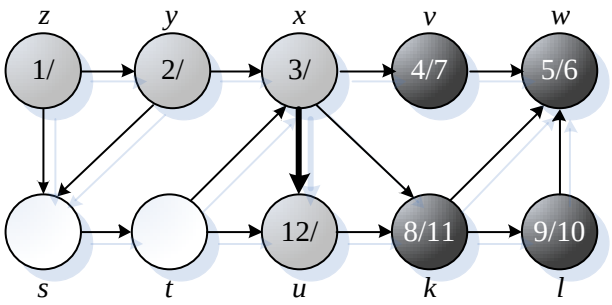


Рисунок 1.102 – Виявлена вершина u

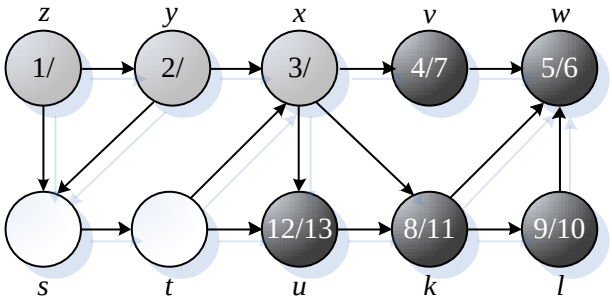


Рисунок 1.103 – Зафарбовування вершини u в чорний колір

Вершина x не має білих суміжних вершин, зафарбовуємо її в чорний колір і ставимо другу мітку), яка дорівнює 14 (рис. 1.104).

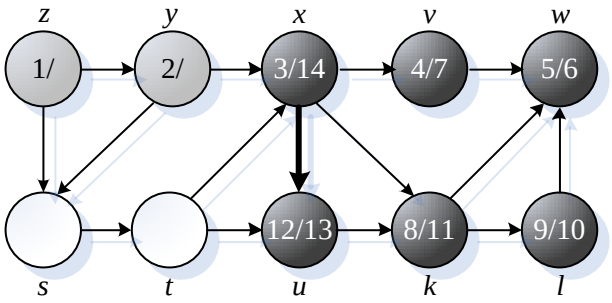


Рисунок 1.104 – Зафарбовування вершини x в чорний колір

Переходимо до сусідньої сірої вершини – y. Вершина y має білу суміжну вершину s. Зафарбовуємо s в сірий колір, ставимо першу мітку часу), яка дорівнює 15 (рис. 1.105).

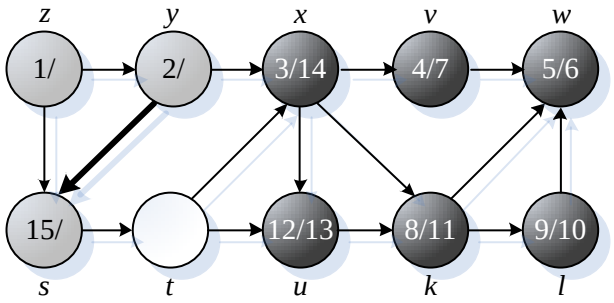


Рисунок 1.105 – Виявлена вершина s

Переходимо до суміжної з k білої вершини t, зафарбовуємо її в сірий колір і ставимо першу мітку – 16 (рис. 1.106).

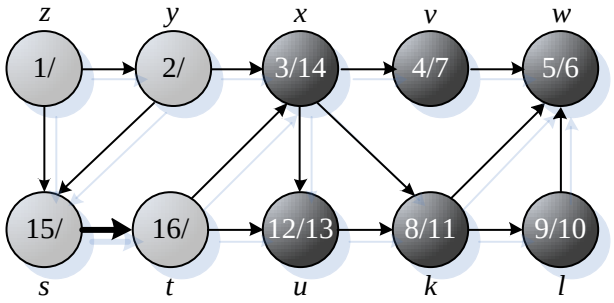


Рисунок 1.106 – Виявлена вершина t

Вершина t не має білих суміжних вершин, тому зафарбовуємо її в чорний колір, ставимо другу мітку часу), яка дорівнює 17 (рис. 1.107), і переходимо до вершини s.

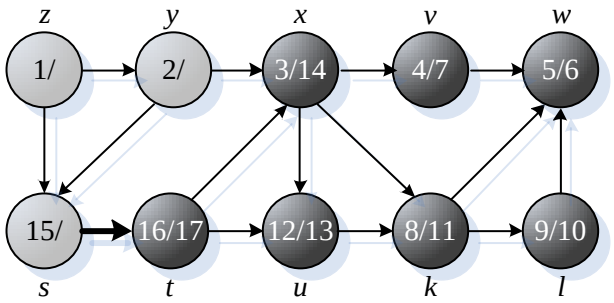


Рисунок 1.107 – Зафарбовування вершини t в чорний колір

Вершина s не має білих суміжних вершин, зафарбовуємо її в чорний колір, ставимо другу мітку часу), яка дорівнює 18 (рис. 1.108), і повертаємося до сірої вершини y.

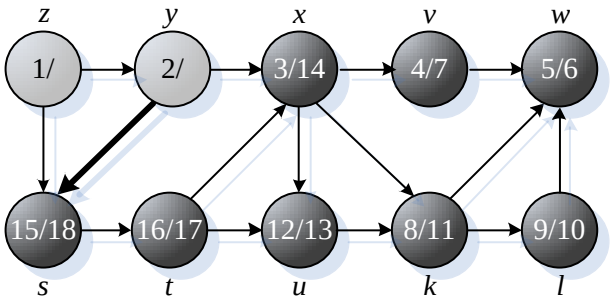


Рисунок 1.108 – Зафарбовування вершини s в чорний колір

Оскільки вершина y не має білих суміжних вершин, то зафарбовуємо її в чорний колір, ставимо другу мітку – 19 (рис. 1.109).

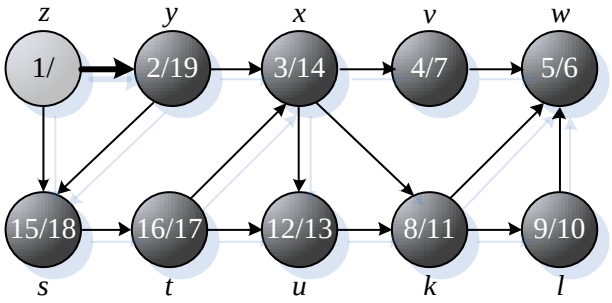


Рисунок 1.109 – Зафарбовування вершини y в чорний колір

Переходимо до вершини z, вона не має білих суміжних вершин, зафарбовуємо її в чорний колір, ставимо другу мітку часу, рівну 20 (рис. 1.110).

Всі вершини мають чорний колір, пошук у глибину завершено.

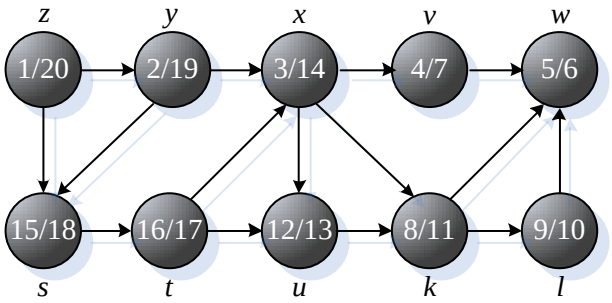


Рисунок 1.110 – Зафарбовування вершини z в чорний колір

Задача 3

Для вихідного графу G, наведеного на рис. 1.111, провести топологічне сортування.

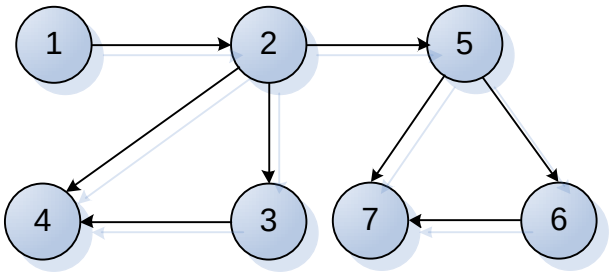


Рисунок 1.111 – Вихідний граф G

Розв'язання

Спочатку необхідно впевнитися, що даний граф не має циклів. Граф, наведений на рис. 1.111, циклів не має. Здійснимо пошук у глибину на заданому графі, обравши за початкову вершину №1. Результат пошуку у глибину показано на рис. 1.112.

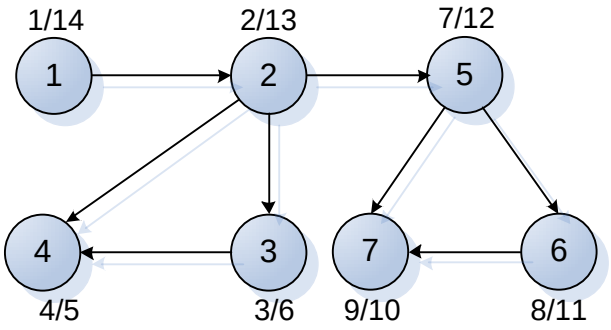


Рисунок 1.112 – Результат пошуку в глибину на графі G

Відповідно до отриманого результату пошуку в глибину розташуємо вершини графа на прямій, починаючи з тої вершини, яка має максимальну другу мітку часу.

Далі розставимо вершини в порядку спадання значення другої мітки. Результат топологічного сортування наведено на рис. 1.113.

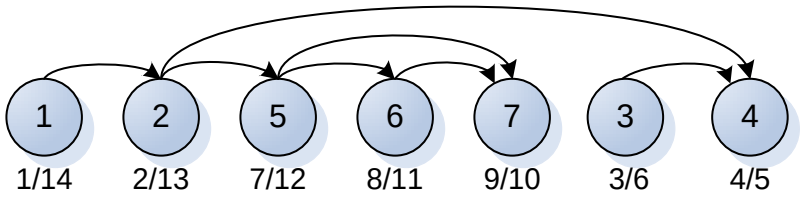


Рисунок 1.113 – Результат топологічного сортування графа G

Як бачимо, всі ребра графа ідуть зліва направо, що свідчить про те, що задачу розв'язано.

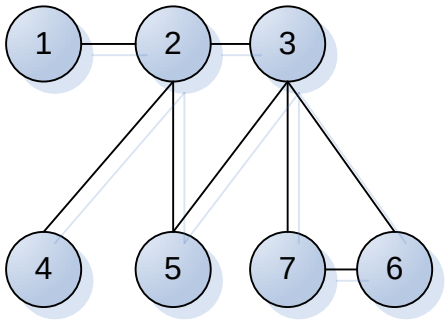
Запитання для самоконтролю

1. Чим пояснюється назва алгоритму «пошук у ширину»?
2. Які відмінності між сірими і чорними вершинами в алгоритмі пошуку в ширину?
3. Опишіть стратегію пошуку в глибину.
4. Які кольори вершин використовуються в алгоритмі пошуку в глибину?
5. Які мітки часу використовуються в алгоритмі пошуку в глибину?
6. Сформулюйте задачу про топологічне сортування графа.

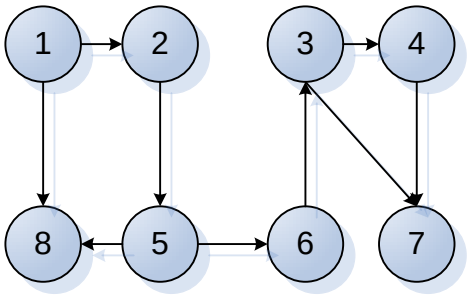
Задачі для аудиторних занять

1. Здійснити пошук у ширину для графів:

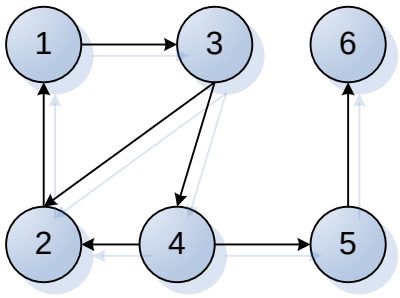
а)



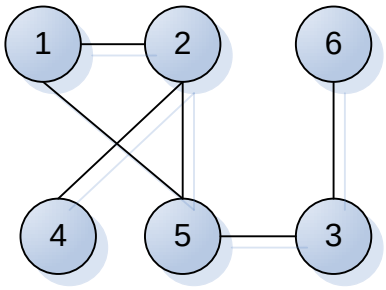
б)



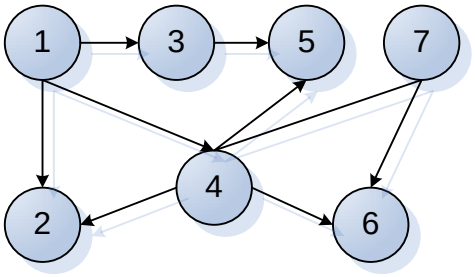
в)



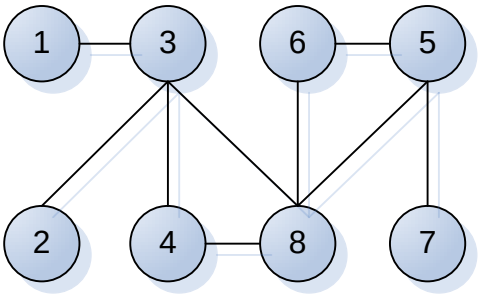
г)



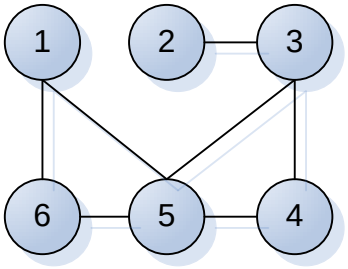
д)



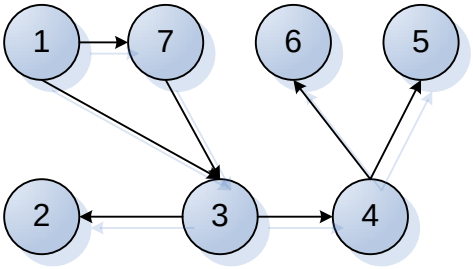
е)



є)

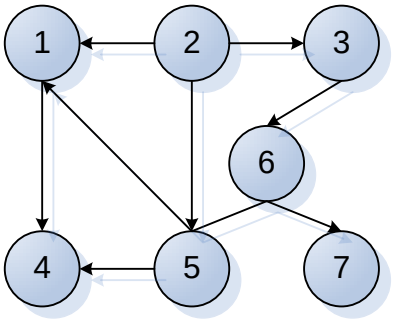


ж)

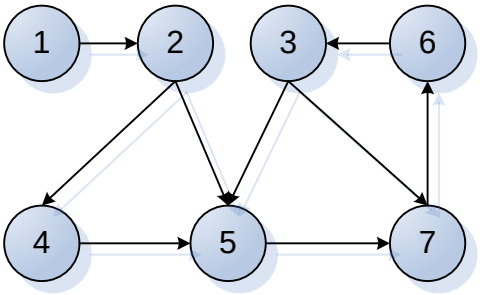


2. Здійснити пошук у глибину для графів:

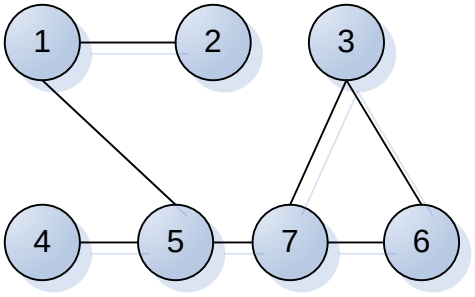
а)



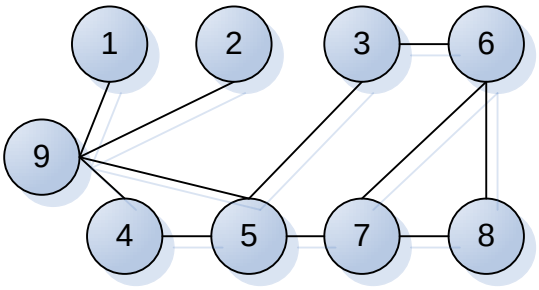
б)



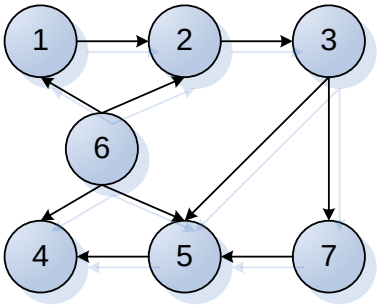
в)



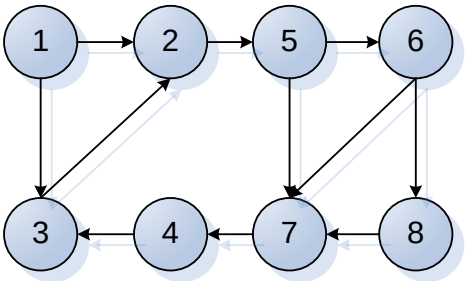
г)



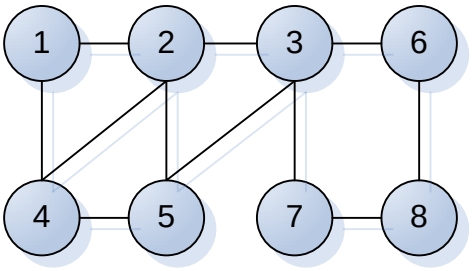
д)



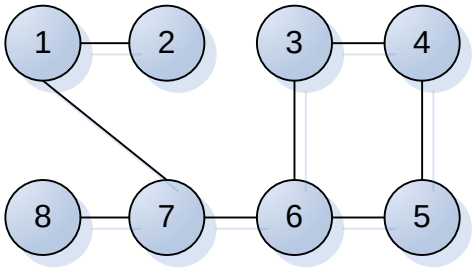
е)



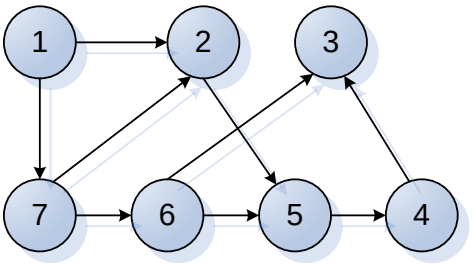
є)



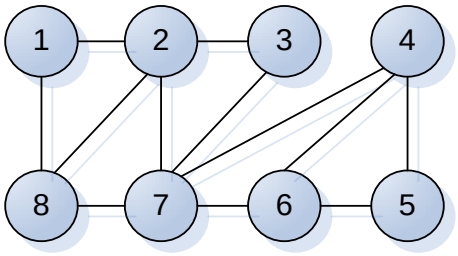
ж)



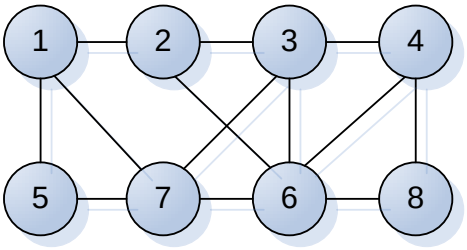
з)



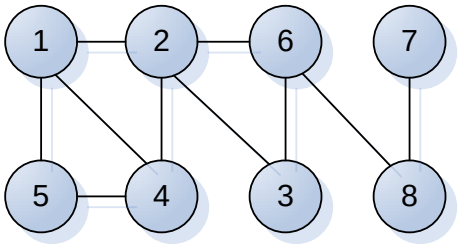
і)



к)

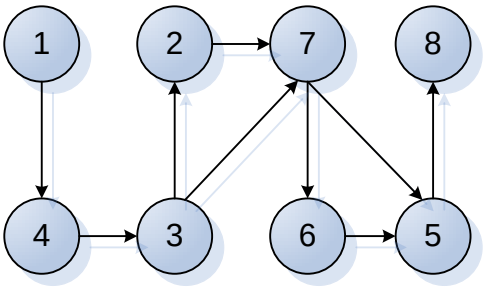


л)

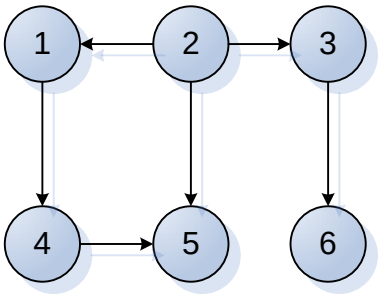


3. Провести топологічне сортування для графів:

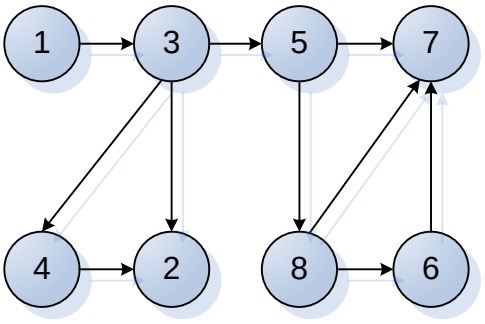
а)



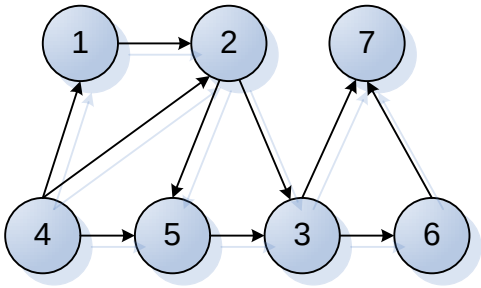
б)

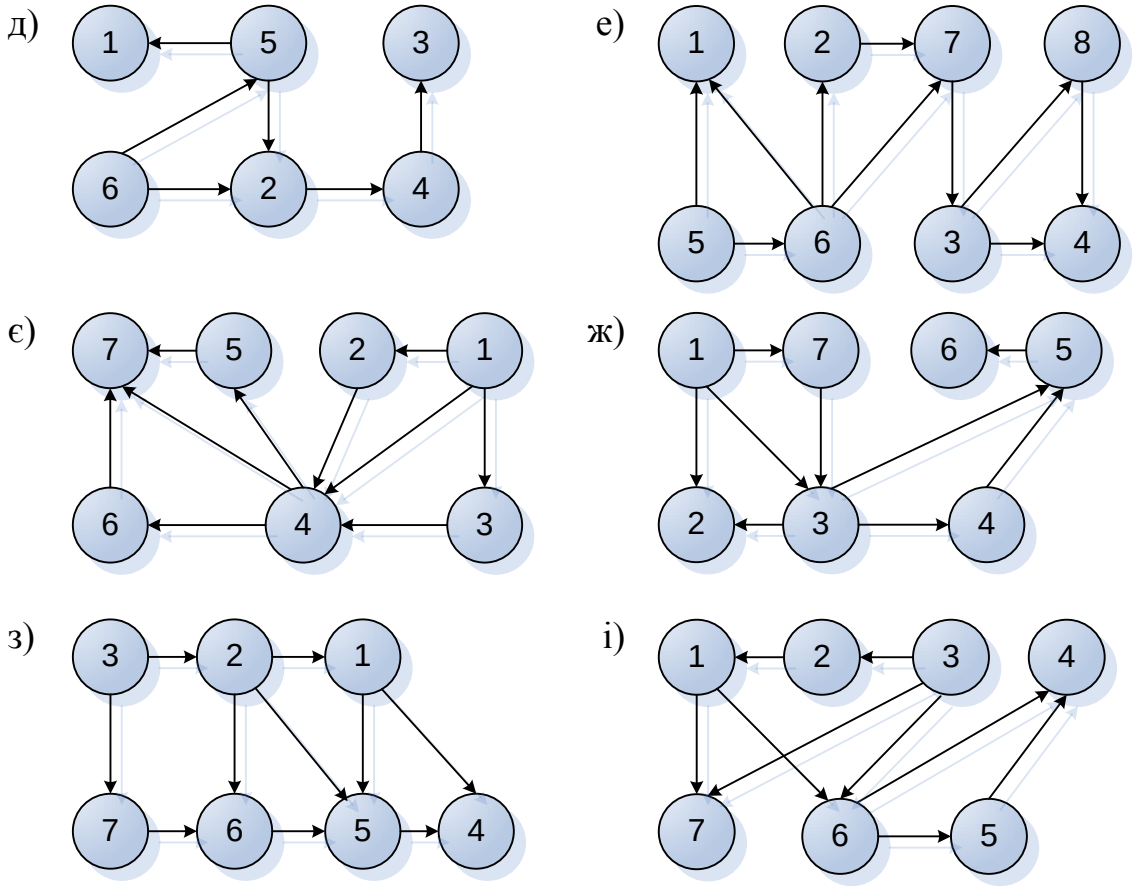


в)



г)





1.8. Геометричні алгоритми

1.8.1. Короткі теоретичні відомості

Алгоритми для розв'язання геометричних задач розглядаються в обчислювальній геометрії [3, 13].

Основна задача обчислювальної геометрії полягає в розробці алгоритмів роботи з геометричними об'єктами, на площині найпростішими з них є точки і відрізки. Точка на декартовій площині задається двома числами: її абсцисою та ординатою. Відрізок з кінцями в точках p_1 і p_2 визначається як множина точок, що подаються у вигляді $p = ap_1 + (1 - a)p_2$, де $0 \leq a \leq 1$.

Важливу роль у багатьох геометричних алгоритмах відіграє визначення напрямку повороту одного вектора до другого. Для розв'язання даної задачі часто використовують поняття векторного (чи, як його називають ще, псевдоскалярного) добутку.

Для обчислення векторного добутку двох векторів $\overrightarrow{p_1}$ і $\overrightarrow{p_2}$, що лежать в одній площині (під яким розуміють площу паралелограма, утвореного точками

$(0,0)$, p_1, p_2 $(p_1 + p_2)$), в роботі [1] пропонується визначення його як визначника матриці:

$$\overrightarrow{p_1} \times \overrightarrow{p_2} = \det\begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = x_1 y_2 - x_2 y_1 = -\overrightarrow{p_2} \times \overrightarrow{p_1}.$$

Якщо порівняти дане визначення векторного добутку з тим, яке дається в курсах аналітичної геометрії, то відповідь на запитання про напрямок повороту є очевидною:

1) якщо $\overrightarrow{p_1} \times \overrightarrow{p_2} > 0$, то від $\overrightarrow{p_2}$ до $\overrightarrow{p_1}$ необхідно повертати за годинниковою стрілкою;

2) якщо $\overrightarrow{p_1} \times \overrightarrow{p_2} < 0$, то від $\overrightarrow{p_2}$ до $\overrightarrow{p_1}$ потрібно повертати проти годинникової стрілки;

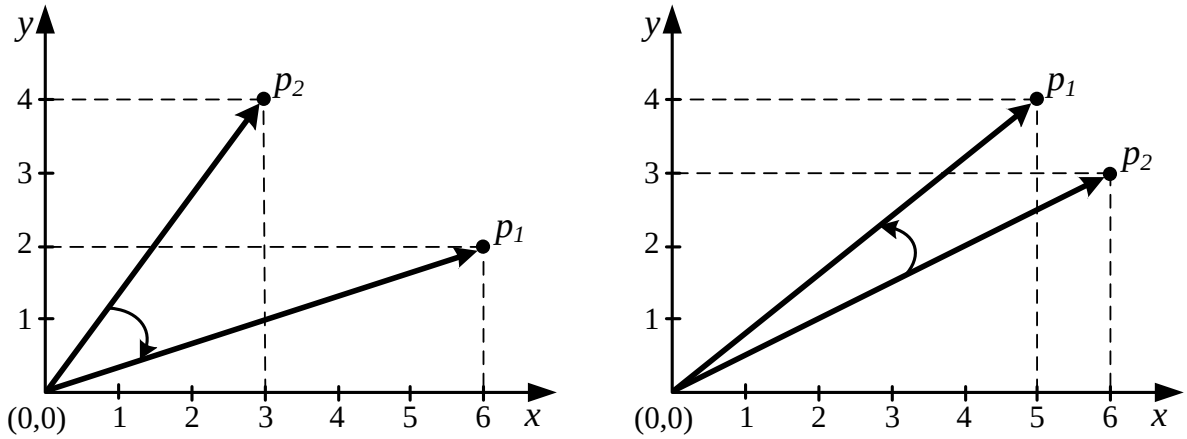
3) якщо векторний добуток рівний нулю, то вектори лежать на одній прямій, тобто вони або направлені в один, або в протилежний бік.

Зазначимо, що при виборі напрямку приймається найкоротший поворот.

Нехай задані два вектори $\overrightarrow{p_1} = (6,2)$ і $\overrightarrow{p_2} = (3,4)$. Обчислимо їх векторний добуток:

$$\overrightarrow{p_1} \times \overrightarrow{p_2} = \det\begin{pmatrix} 6 & 3 \\ 2 & 4 \end{pmatrix} = 6 \times 4 - 3 \times 2 = 18.$$

Оскільки векторний добуток додатній, то від $\overrightarrow{p_2}$ до $\overrightarrow{p_1}$ необхідно повертати за годинниковою стрілкою (рис. 1.114, *а*)).



а *б*

Рисунок 1.114 – Визначення повороту вектора $\overrightarrow{p_2}$ до $\overrightarrow{p_1}$:

а – за годинниковою стрілкою; *б* – проти годинникової стрілки

Нехай задані два вектори $\overrightarrow{p_1} = (5,4)$ і $\overrightarrow{p_2} = (6,3)$. Обчислимо їх векторний добуток:

$$\overrightarrow{p_1} \times \overrightarrow{p_2} = \det\begin{pmatrix} 5 & 6 \\ 4 & 3 \end{pmatrix} = 5 \times 3 - 6 \times 4 = -9$$

Оскільки векторний добуток, то від $\overrightarrow{p_2}$ до $\overrightarrow{p_1}$ необхідно повертати проти годинникової стрілки (рис. 1.114, *б*)).

Задача побудови опуклих оболонок – одна з центральних задач обчислювальної геометрії, що визначається не лише величезною кількістю її додатків (наприклад, у математичній статистиці, розробці ігор, обробці зображень), але і її корисністю для алгоритмів розв'язання інших задач, які використовують побудову опуклої оболонки як елементарну операцію.

Поняття опуклої оболонки визначається таким чином. Нехай задана деяка множина точок на площині. Опуклою оболонкою називається найменший опуклий багатокутник, що містить дані точки (рис. 1.115).

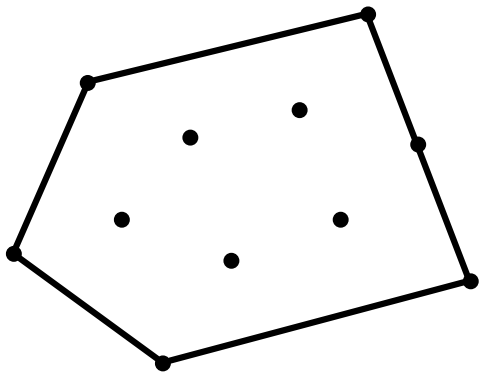


Рисунок 1.115 – Опукла оболонка точок

Відома велика кількість алгоритмів для розв'язання даної задачі, наприклад:

- алгоритм типу «Розділяй і володарюй»;
- алгоритм «швидкої побудови»;
- алгоритм Чана;
- алгоритм обходу Джарвіса;
- алгоритм обходу Грехема та інші.

Розглянемо детальніше алгоритм обходу Джарвіса [14] і алгоритм Грехема [15].

Алгоритм Джарвіса

Задана множина N точок на площині.

Побудова опуклої оболонки даної множини починається з точки, яка гарантовано буде входити до опуклої оболонки. Очевидно, що сама ліва нижня точка підходить під цю умову, позначимо її p_0.

Далі знаходиться наступна точка p_i (де i – крок алгоритму) у випадку обходу проти годинникової стрілки. Вона, очевидно, володіє властивістю, що інші точки лежать зліва від вектора $\overrightarrow{p_{i-1}p_i}$.

Пошук точки, що включається на i-му кроці алгоритму можна здійснити так.

Вибираємо будь-яку точку не з оболонки як претендента на включення, позначимо її p_{best}. Потім послідовно порівнюємо точку p_{best} з точками не з оболонки (а також з початковою точкою оболонки p_0, оскільки нам необхідно в кінці побудови оболонки «зімкнути» її, тобто із поточної точки знову прийти в точку p_0). Для порівняння точок будемо використовувати знак векторного добутку $(p_{best} - p_{i-1}) \times (q_j - p_{i-1})$.

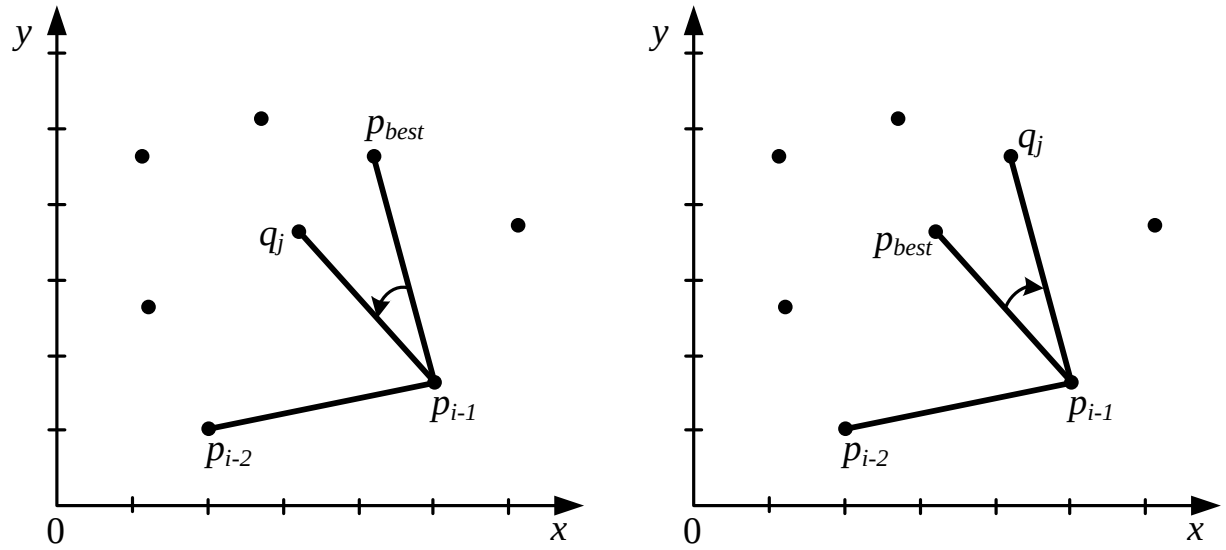
Якщо для одної із точок, що розглядаються, вказаний вираз менший від нуля, то вважаємо її претендентом на включення і продовжуємо перевірку інших точок. Якщо ж значення векторного добутку рівне нулю, то вибираємо ту точку, яка розташована далі від точки p_{i-1}.

На рис. 1.116 показані два випадки, які можуть бути отримані в результаті порівняння точок p_{best} і q_j.

Точки p_{i-2} і p_{i-1} на рис. 116 позначають точки, що належать до опуклої оболонки, знайдені на двох попередніх кроках алгоритму (очевидно, що для кроку 1 точка p_{i-2} на рис. 116 буде відсутня).

Перший випадок (рис. 1.116, *а*) відповідає додатному значенню вказаного вище векторного добутку. Як це можна побачити на рис. 116, в такому випадку необхідно залишити точку p_{best} як претендента на включення до опуклої оболонки.

Другий випадок (рис. 1.116, *б*) відповідає від'ємному значенню векторного добутку, тобто поворот від точки p_{best} до точки q_j відбувається за годинниковою стрілкою, значить, як претендента на включення до опуклої оболонки необхідно розглянути точку q_j, тобто замінити p_{best} точкою q_j.



а *б*

Рисунок 1.116 – Порівняння точок p_{best} і q_j на i-му кроці алгоритму:

а – точка p_{best} залишається; *б* – точка p_{best} замінюється на q_j

Якщо ж значення векторного добутку дорівнює нулю, то точки p_{best}, q_j і p_{i-1} лежать на одній прямій. Тоді як кандидата на включення до опуклої оболонки необхідно з точок p_{best} і q_j вибрати ту, яка розташована далі від точки p_{i-1}.

Продовжуючи дану процедуру, на одному з кроків повернемося до точки p_0. Це буде означати, що опукла оболонка побудована.

Алгоритм обходу Грехема був описаний ним в одній із його перших робіт [15], присвяченій питанню розробки ефективних геометричних алгоритмів.

Алгоритм починає свою роботу з находження початкової точки p_0 із заданої множини точок N (не меншої трьох), яка гарантовано є вершиною опук-

лої оболонки. За таку береться сама ліва із самих нижніх точок множини N.

На *етапі попереднього сортування* всі точки вихідної множини (окрім p_0) сортуються за зростанням полярного кута, утвореного кожною поточною точкою q_j, відносно точки p_0. Якщо дві точки мають однаковий полярний кут, то залишаємо для подальшого розгляду ту точку, відстань від якої до точки p_0 більша, оскільки точка з меншою відстанню до p_0 заздалегідь не потрапляє до оболонки. Відсортовані точки записуються до стека. Зазначимо, що таке сортування можна здійснити за допомогою векторного добутку.

На *етапі побудови опуклої оболонки* алгоритм виконує покрокову обробку відсортованих точок, формуючи оболонку шляхом видалення тих точок, в яких не здійснюється лівий поворот. Це очевидно, оскільки оболонка будується проти годинникової стрілки, починаючи від p_0, значить, рух здійснюється завжди наліво.

На кожному кроці даного етапу перевіряється взаємне розташування трьох точок (спочатку вибираються p_0 в перші дві точки у відсортованому списку). Перевірка полягає у визначенні, чи утворюють точки, які перевіряються, лівий або правий поворот. Таку перевірку також можна здійснити за допомогою векторного добутку.

Будемо говорити, що три точки p_{i-2}, p_{i-1} і p_i утворюють *лівий поворот*, якщо точка p_i знаходиться зліва від прямої, утвореної вектором, направленим з точки p_{i-2} в p_{i-1}.

Якщо точки утворюють лівий поворот, то алгоритм рухається далі за списком, розглядаючи три послідовні точки, починаючи с p_{i-1}.

Якщо точки не утворюють лівий поворот, то середня з них видаляється із списку, і алгоритм «повертається», розглядаючи три послідовні точки, починаючи з p_{i-2}.

Алгоритм буде закінчено, коли всі точки відсортованого списку будуть оброблені.

1.8.2. Приклади розв'язання задач

Задача

На площині задані 7 точок (рис. 1.117). Необхідно побудувати їх опуклу оболонку.

Розв'язання

Випишемо координати всіх заданих точок:

$$q_1 = (1,8),\ q_2 = (4,6),\ q_3 = (6,2),\ q_4 = (10,12),\ q_5 = (11,10),\ q_6 = (12,4),$$
$$q_7 = (15,9).$$

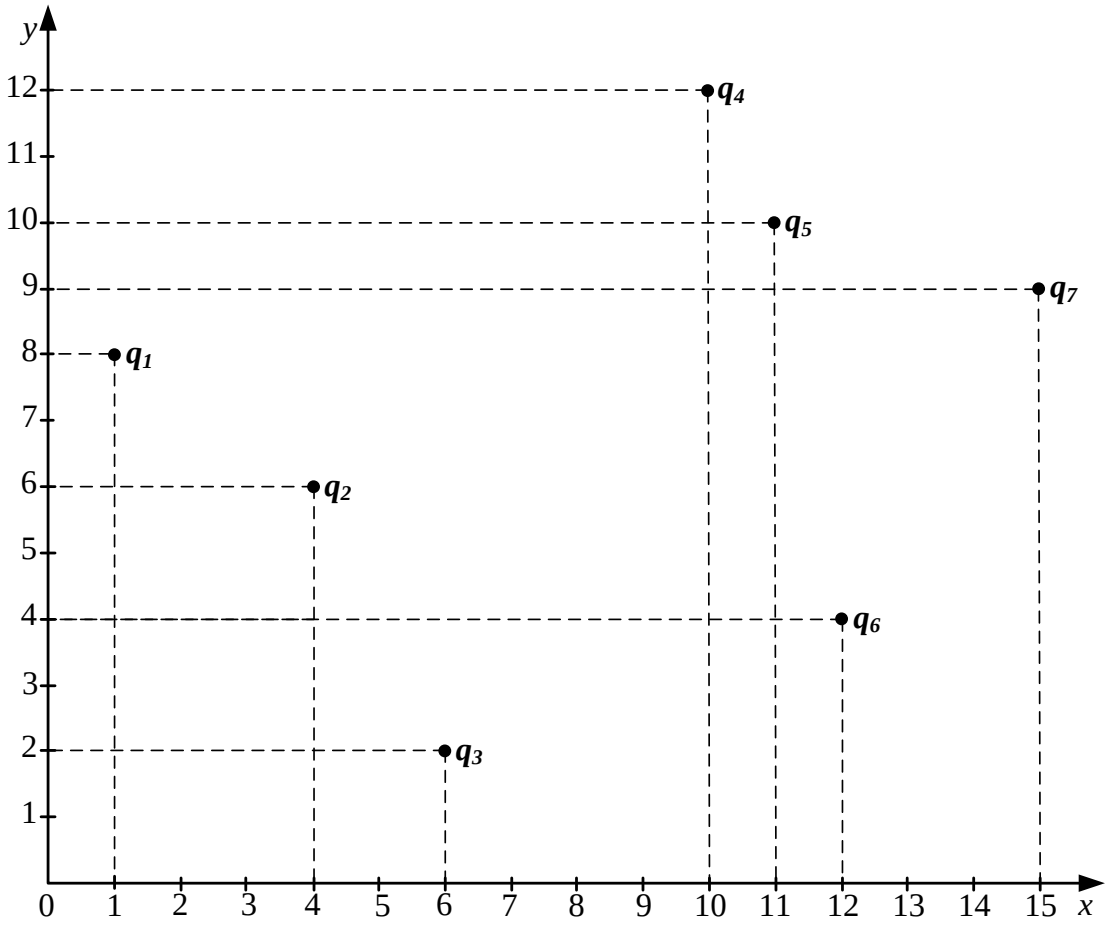


Рисунок 1.117 – Вихідна множина точок

Побудову опуклої оболонки будемо виконувати на основі обходу Джарвіса. Будемо далі позначати як p_i точку, яка на i-му кроці алгоритму включена до опуклої оболонки Q.

Крок 0. Знаходимо початкову точку в обході опуклої оболонки. За таку вибираємо точку, у якої мінімальна ордината, тобто в даному випадку, $q_3 = (6,2)$.

Покладемо $i = 0$, $p_0 = q_3 = (6,2)$.

Включимо точку до оболонки:

$$Q = \{q_3\}.$$

Крок 1. Знаходимо наступну за q_3 точкою на опуклій оболонці.

Покладемо $i = 1$.

Як претендента на включення до опуклої оболонки візьмемо першу точку не з оболонки і позначимо її p_{best} (в даному випадку $p_{best} = q_1$).

Оскільки $p_{best} = q_1$, то порівняння починаємо з другої точки.

Покладемо $j = 2$. Порівняємо $q_j = q_2$ з p_{best}.

Запишемо векторний добуток $(p_{best} - p_0) \times (q_2 - p_0)$:

$$((1,8) - (6,2)) \times ((4,6) - (6,2)) = (-5,6) \times (-2,4).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -5 & -2 \\ 6 & 4 \end{pmatrix} = -20 + 12 = -8.$$

Оскільки векторний добуток від'ємний, замінимо p_{best}:

$$p_{best} = q_2 = (4,6).$$

Покладемо $j = j + 1 = 3$. Окільки точка q_3 – це єдина точка в оболонці, переходимо до наступної.

Покладемо $j = j + 1 = 4$. Порівняємо $q_j = q_4$ з p_{best}.

Запишемо векторний добуток $(p_{best} - p_0) \times (q_4 - p_0)$:

$$((4,6) - (6,2)) \times ((10,12) - (6,2)) = (-2,4) \times (4,10).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -2 & 4 \\ 4 & 10 \end{pmatrix} = -20 - 16 = -36.$$

Оскільки векторний добуток від'ємний, замінимо p_{best}:

$$p_{best} = q_4 = (10,12).$$

Покладемо $j = j + 1 = 5$. Порівняємо $q_j = q_5$ з p_{best}.

Запишемо векторний добуток $(p_{best} - p_0) \times (q_5 - p_0)$:

$$((10,12) - (6,2)) \times ((11,10) - (6,2)) = (4,10) \times (5,8).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} 4 & 5 \\ 10 & 8 \end{pmatrix} = 32 - 50 = -18.$$

Оскільки векторний добуток від'ємний, замінимо p_{best}:

$$p_{best} = q_5 = (11,10).$$

Покладемо $j = j+1 = 6$. Порівняємо $q_j = q_6$ з p_{best}.

Запишемо векторний добуток $(p_{best} - p_0) \times (q_6 - p_0)$:

$$((11,10) - (6,2)) \times ((12,4) - (6,2)) = (5,8) \times (6,2).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} 5 & 6 \\ 8 & 2 \end{pmatrix} = 10 - 48 = -38.$$

Оскільки векторний добуток від'ємний, замінимо p_{best}:

$$p_{best} = q_6 = (12,4).$$

Покладемо $j = j+1 = 7$. Порівняємо $q_j = q_7$ з p_{best}.

Запишемо векторний добуток $(p_{best} - p_0) \times (q_7 - p_0)$:

$$((12,4) - (6,2)) \times ((15,9) - (6,2)) = (6,2) \times (9,7).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} 6 & 9 \\ 2 & 7 \end{pmatrix} = 49 - 18 = 31.$$

Оскільки векторний добуток додатний, то точка p_{best} зберігається.

Усі точки розглянуті, значить, $p_1 = q_6 = (12,4)$.

Включимо знайдену точку до оболонки:

$$Q = \{q_3, q_6\}.$$

Крок 2. Знаходимо наступну за q_6 точку на опуклій оболонці.

Покладемо $i = 2$.

Як претендента на включення візьмемо $p_{best} = q_1$. Порівняння починаємо з другої точки.

Покладемо $j = 2$. Порівняємо $q_j = q_2$ з p_{best}.

Запишемо векторний добуток $(p_{best} - p_1) \times (q_2 - p_1)$:

$$((1,8) - (12,4)) \times ((4,6) - (12,4)) = (-11,4) \times (-8,2).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -11 & -8 \\ 4 & 2 \end{pmatrix} = -22 + 32 = 10.$$

Оскільки векторний добуток додатний, то точка p_{best} зберігається.

Покладемо $j = j + 1 = 3$. Порівняємо $q_j = q_3$ з p_{best}.

Запишемо векторний добуток $(p_{best} - p_1) \times (q_3 - p_1)$:

$$((1,8) - (12,4)) \times ((6,2) - (12,4)) = (-11,4) \times (-6,-2).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -11 & -6 \\ 4 & -2 \end{pmatrix} = 22 + 24 = 44.$$

Оскільки векторний добуток додатний, то точка p_{best} зберігається.

Покладемо $j = j + 1 = 4$. Порівняємо $q_j = q_4$ з p_{best}.

Запишемо векторний добуток $(p_{best} - p_1) \times (q_4 - p_1)$:

$$((1,8) - (12,4)) \times ((10,12) - (12,4)) = (-11,4) \times (-2,8).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -11 & -2 \\ 4 & 8 \end{pmatrix} = -88 + 8 = -80.$$

Оскільки векторний добуток від'ємний, замінимо p_{best}:

$$p_{best} = q_4 = (10,12).$$

Покладемо $j = j + 1 = 5$.

Запишемо векторний добуток $(p_{best} - p_1) \times (q_5 - p_1)$.

$$((10,12) - (12,4)) \times ((11,10) - (12,4)) = (-2,8) \times (-1,6).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -2 & -1 \\ 8 & 6 \end{pmatrix} = -12 + 8 = -4.$$

Оскільки векторний добуток від'ємний, то точку p_{best} замінимо, точкою q_5:

$$p_{best} = q_5 = (11,10).$$

Покладемо $j = j + 1 = 6$. Оскільки точка q_6 уже включена до опуклої оболонки, то переходимо до наступної точки.

Покладемо $j = j + 1 = 7$. Порівняємо $q_j = q_7$ з p_{best}.

Запишемо векторний добуток $(p_{best} - p_1) \times (q_7 - p_1)$:

$$((11,10) - (12,4)) \times ((15,9) - (12,4)) = (-1,6) \times (3,5).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -1 & 3 \\ 6 & 5 \end{pmatrix} = -5 - 18 = -23.$$

Оскільки векторний добуток від'ємний, замінимо p_{best} точкою p_7:

$$p_{best} = q_7 = (15,9).$$

Всі точки розглянуті, значить, $p_2 = q_7 = (15,10)$.

Включимо знайдену точку до оболонки:

$$Q = \{q_3, q_6, q_7\}.$$

Крок 3. Знаходимо наступну після q_7 точку на оболонці.

Покладемо $i = 3$.

Як претендента на включення до опуклої оболонки на 3-му кроці візьмемо першу точку не з оболонки $p_{best} = q_1$.

Покладемо $j = 2$. Порівняємо $q_j = q_2$ з p_{best}.

Запишемо векторний добуток $(p_{best} - p_2) \times (q_2 - p_2)$:

$$((1,8) - (15,9)) \times ((4,6) - (15,9)) = (-14,-1) \times (-11,-3).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -14 & -11 \\ -1 & -3 \end{pmatrix} = 42 - 11 = 31.$$

Оскільки векторний добуток додатний, то точка p_{best} зберігається.

Покладемо $j = j + 1 = 3.$

Запишемо векторний добуток $(p_{best} - p_2) \times (q_3 - p_2)$:

$$((1,8) - (15,9)) \times ((6,2) - (15,9)) = (-14,-1) \times (-9,-7).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -14 & -9 \\ -1 & -7 \end{pmatrix} = 98 - 9 = 89.$$

Оскільки векторний добуток додатний, то точка p_{best} зберігається.

Покладемо $j = j + 1 = 4.$

Запишемо векторний добуток $(p_{best} - p_2) \times (q_4 - p_2)$:

$$((1,8) - (15,9)) \times ((10,12) - (15,9)) = (-14,-1) \times (-5,3).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -14 & -5 \\ -1 & 3 \end{pmatrix} = -42 - 5 = -47.$$

Оскільки векторний добуток від'ємний, то точку p_{best} замінимо точкою q_4:

$$p_{best} = q_4 = (10,12).$$

Покладемо $j = j + 1 = 5$.

Запишемо векторний добуток $(p_{best} - p_2) \times (q_5 - p_2)$:

$$((10,12) - (15,9)) \times ((11,10) - (15,9)) = (-5,3) \times (-4,1).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -5 & -4 \\ 3 & 1 \end{pmatrix} = -5 + 12 = 7.$$

Оскільки векторний добуток додатний, то точка p_{best} зберігається.

Покладемо $j = j + 1 = 6$. Оскільки точка q_6 вже включена до опуклої оболонки, то переходимо до наступної точки.

Покладемо $j = j + 1 = 7$. Точка q_7 уже включена до опуклої оболонки.

Усі точки розглянуто, значить, $p_3 = q_4 = (10,12)$.

Включимо знайдену точку до оболонки:

$$Q = \{q_3, q_{6,} q_7, q_4\}.$$

Крок 4. Знаходимо наступну після q_4 точку на оболонці.

Покладемо $i = 4$ і $p_{best} = q_1 = (1,8)$.

Починаємо порівняння з другої точки, $j = 2$.

Запишемо векторний добуток $(p_{best} - p_3) \times (q_2 - p_3)$:

$$((1,8) - (10,12)) \times ((4,6) - (10,12)) = (-9,-4) \times (-6,-6).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -9 & -6 \\ -4 & -6 \end{pmatrix} = 54 - 24 = 30.$$

Оскільки векторний добуток додатний, то точка p_{best} зберігається.

Покладемо $j = j + 1 = 3$.

Запишемо векторний добуток $(p_{best} - p_3) \times (q_3 - p_3)$:

$$((1,8) - (10,12)) \times ((6,2) - (10,12)) = (-9,-4) \times (-4,-10).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -9 & -4 \\ -4 & -10 \end{pmatrix} = 90 - 16 = 74.$$

Оскільки векторний добуток додатний, то точка p_{best} зберігається.

Покладемо $j = j + 1 = 4$. Оскільки точка q_4 уже включена до опуклої оболонки, то переходимо до наступної точки.

Покладемо $j = j + 1 = 5$.

Запишемо векторний добуток $(p_{best} - p_3) \times (q_5 - p_3)$:

$$((1,8) - (10,12)) \times ((11,10) - (10,12)) = (-9,-4) \times (1,-2).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -9 & 1 \\ -4 & -2 \end{pmatrix} = 18 + 4 = 22.$$

Оскільки векторний добуток додатний, то точка p_{best} зберігається.

Покладемо $j = j + 1 = 6$. Оскільки точка q_6 уже включена до опуклої оболонки, то переходимо до наступної точки.

Покладемо $j = j + 1 = 7$. Точка q_7 уже включена до опуклої оболонки.

Усі точки розглянуті, значить, $p_4 = q_1 = (1,8)$.

Включимо знайдену точку до оболонки:

$$Q = \{q_3, q_{6,} q_7, q_4, q_1\}.$$

Крок 5. Знаходимо наступну після q_1 точку на оболонці.

Покладемо $i = 5$ і $p_{best} = q_2 = (4,6)$.

Порівняємо q_3 з p_{best}.

Запишемо векторний добуток $(p_{best} - p_4) \times (q_3 - p_4)$:

$$((4,6) - (1,8)) \times ((6,2) - (1,8)) = (3,-2) \times (5,-6).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} 3 & 5 \\ -2 & -6 \end{pmatrix} = -18 + 10 = -8.$$

Оскільки векторний добуток від'ємний, то точку p_{best} замінимо точкою q_3:

$$p_{best} = q_3 = (6,2).$$

Покладемо $j = j + 1 = 4$. Оскільки q_4 уже включена до опуклої оболонки, то переходимо до наступної точки.

Покладемо $j = j + 1 = 5$.

Запишемо векторний добуток $(p_{best} - p_4) \times (q_5 - p_4)$:

$$((6,2) - (1,8)) \times ((11,10) - (1,8)) = (5,-6) \times (10,2).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} 5 & 10 \\ -6 & 2 \end{pmatrix} = 10 + 60 = 70.$$

Оскільки векторний добуток додатний, то точка p_{best} зберігається.

Покладемо $j = j + 1 = 6$. Оскільки точка q_6 уже включена до опуклої оболонки, то переходимо до наступної точки.

Покладемо $j = j + 1 = 7$. Точка q_7 уже включена до опуклої оболонки.

Усі точки розглянуто, а оскільки отримана на даному кроці точка-претендент збігається з початковою точкою q_3, то оболонка повністю побудована:

$$Q = \{q_3, q_{6,} q_7, q_4, q_1\}.$$

На рис. 1.118 показаний результат побудови опуклої оболонки для заданого в даній задачі набору точок.

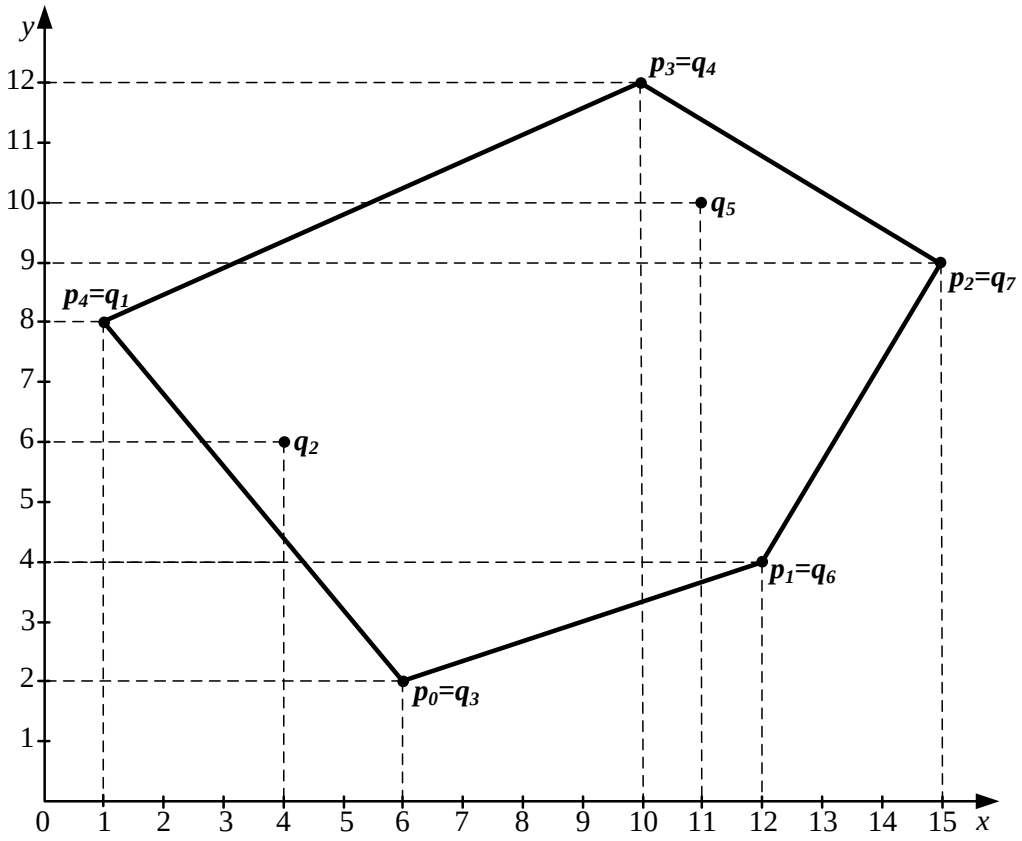


Рисунок 1.118 – Результат побудови опуклої оболонки

Задача 2

На площині задані 5 точок: $p_0 = (1,1)$, $p_1 = (7,2)$, $p_2 = (5,3)$, $p_3 = (6,5)$, $p_4 = (4,6)$ і $p_5 = (3,9)$. Побудувати опуклу оболонку алгоритмом Грехема.

Вказівка. Точки $< p_1, p_2, p_3, p_4, p_5 >$ відсортовані в порядку зростання полярних кутів відносно стартової точки p_0.

Розв'язання

Оскільки задано початкову точку p_0 і відомий відсортований список точок, то переходимо до другого етапу алгоритму – побудови оболонки. Задану множину точок задачі показано на рис. 1.119.

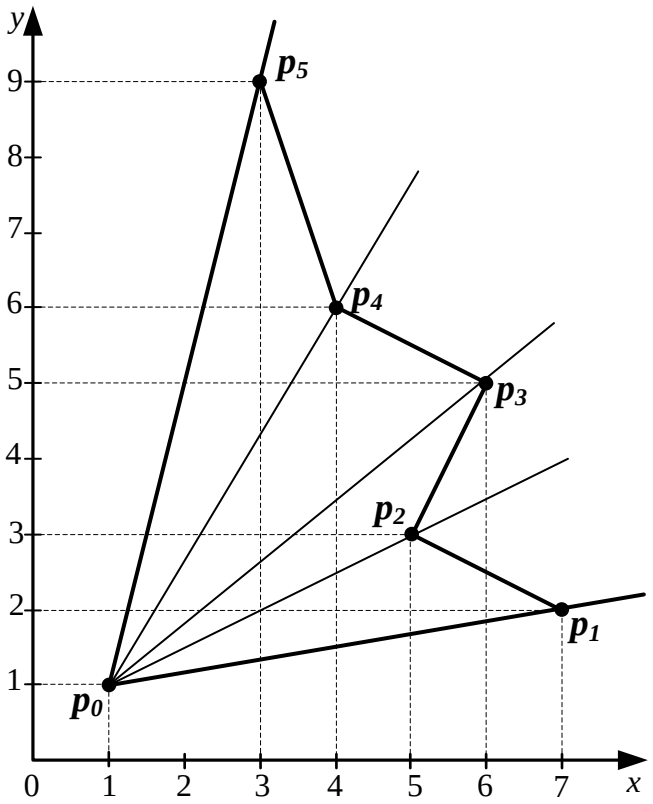


Рисунок 1.119 – Вихідна множина точок

Крок 0. Включаємо до оболонки стартову точку p_0, а також першу точку p_1 з відсортованого списку. Оболонка, що формується:

$$Q = \{p_0, p_1\}.$$

Крок 1. Розглянемо дві точки з оболонки, що формується, p_0, p_1 і наступну нерозглянуту точку з відсортованого списку – p_2. Визначаємо напрям повороту з точки p_1 в точку p_2, для чого записуємо векторний добуток $(p_1 - p_0) \times (p_2 - p_o)$:

$$((7,2)-(1,1)) \times ((5,3)-(1,1)) = (6,1) \times (4,2).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} 6 & 4 \\ 1 & 2 \end{pmatrix} = 12 - 4 = 8.$$

Оскільки векторний добуток додатний, то поворот лівий, тобто з точки p_1 в точку p_2 необхідно йти наліво відносно прямої, яка визначається точками p_0 і p_1. Значить, включаємо до оболонки точку p_2:

$$Q = \{p_0, p_1, p_2\}.$$

Крок 2. Рухаємося вперед по відсортованому списку, тобто розглядаємо дві останні точки p_1, p_2 з оболонки, яку формуємо, і наступну нерозглянуту точку з відсортованого списку – p_3.

Запишемо векторний добуток $(p_2 - p_1) \times (p_3 - p_1)$:

$$((5,3) - (7,2)) \times ((6,5) - (7,2)) = (-2,1) \times (-1,3).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -2 & -1 \\ 1 & 3 \end{pmatrix} = -6 + 1 = -5.$$

Оскільки векторний добуток від'ємний, то поворот правий, тобто з точки p_2 в точку p_3 необхідно йти направо відносно прямої, що визначається точками p_1 і p_2. Значить, виключаємо з оболонки, що формується, точку p_2 і виключаємо p_3:

$$Q = \{p_0, p_1, p_3\}.$$

Крок 3. Розглянемо дві останні точки p_1, p_3 з оболонки, що формується, і наступну нерозглянуту точку p_4 з відсортованого списку.

Запишемо векторний добуток $(p_3 - p_1) \times (p_4 - p_1)$:

$$((6,5) - (7,2)) \times ((4,6) - (7,2)) = (-1,3) \times (-3,4).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -1 & -3 \\ 3 & 4 \end{pmatrix} = -4 + 9 = 5.$$

Оскільки векторний добуток додатний, то поворот лівий, значить, включаємо до оболонки точку p_4:

$$Q = \{p_0, p_1, p_3, p_4\}.$$

Крок 4. Розглянемо дві останні точки p_3, p_4 з оболонки, що формується, і наступну нерозглянуту точку з відсортованого списку – p_5.

Запишемо векторний добуток $(p_4 - p_3) \times (p_5 - p_3)$:

$$((4,6) - (6,5)) \times ((3,9) - (6,5)) = (-2,1) \times (-3,4).$$

Обчислимо векторний добуток через визначник матриці:

$$\det\begin{pmatrix} -2 & -3 \\ 1 & 4 \end{pmatrix} = -8 + 3 = -5.$$

Оскільки векторний добуток від'ємний, то поворот правий, отже, виключаємо з оболонки, що формується, точку p_4 і включаємо p_5:

$$Q = \{p_0, p_1, p_3, p_5\}.$$

Усі точки з відсортованого списку розглянуто, значить, оболонку побудовано. Вона включає наступні точки (рис. 1.120), починаючи зі стартової в порядку обходу проти годинникової стрілки:

$$Q = \{p_0, p_1, p_3, p_5\}.$$

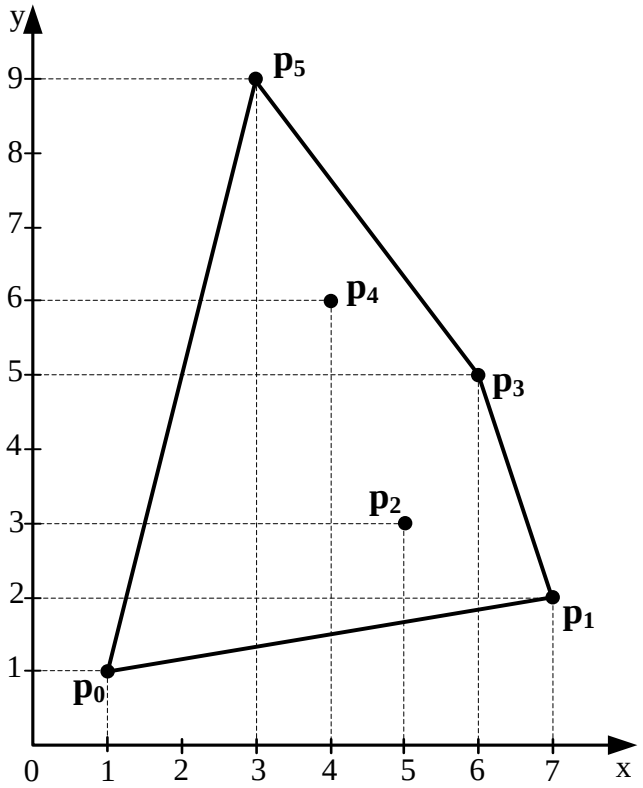


Рисунок 1.120 – Опукла оболонка множини точок задачі 2

Запитання для самоконтролю

1. Дайте визначення опуклої оболонки множини.

2. Назвіть основні алгоритми пошуку опуклої оболонки.

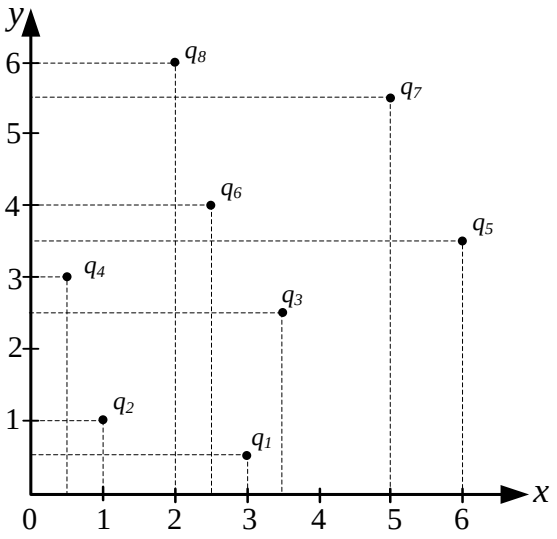
3. У чому полягає ідея алгоритму Джарвіса?

4. Який з алгоритмів (перегляд Грехема чи прохід Джарвіса) потребує менше часу для пошуку опуклої оболонки?

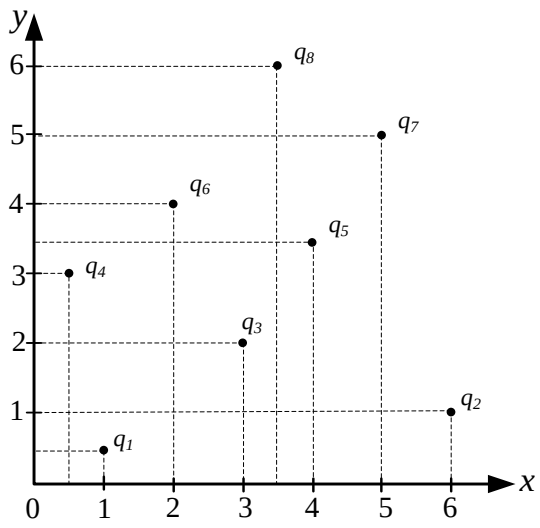
Задачі для аудиторних занять

1. Визначити опуклу оболонку методом Грехема для заданих точок:

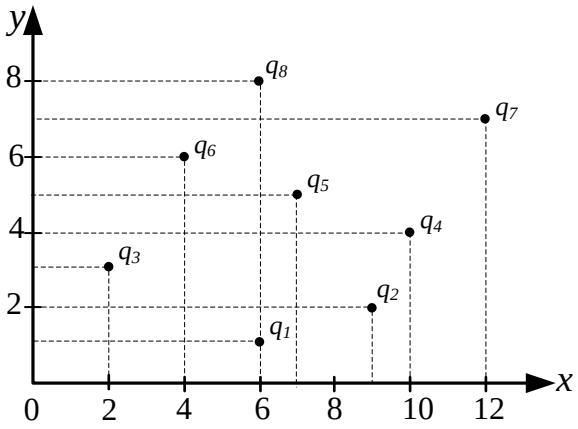
а)



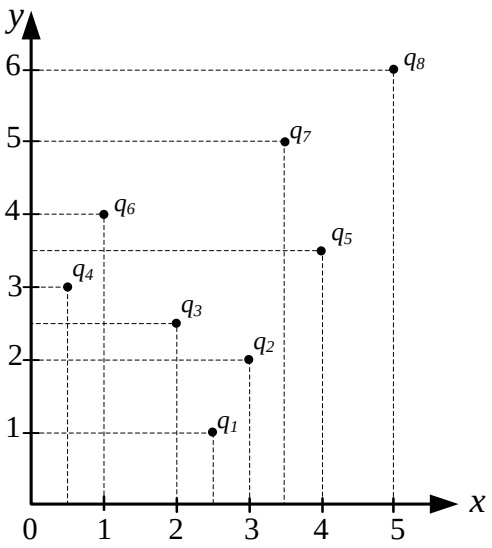
б)



в)



г)



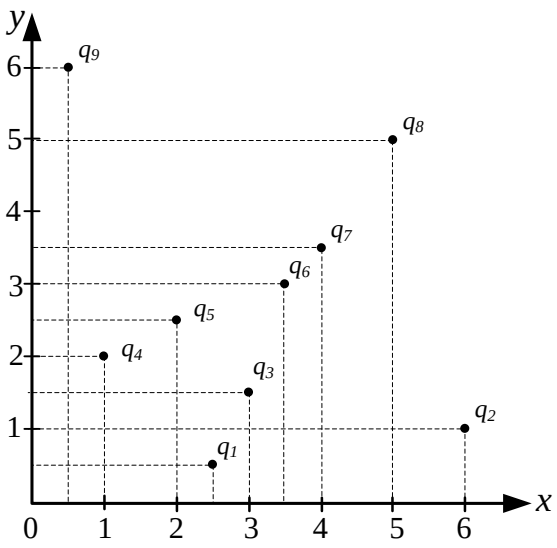
2. Визначити опуклу оболонку методом Джарвіса для заданих точок:

а) $a_1 = (1,5)$, $a_2 = (6,6)$, $a_3 = (7,4)$, $a_4 = (5,1)$, $a_5 = (4,5)$, $a_6 = (8,3)$, $a_7 = (3,3)$;

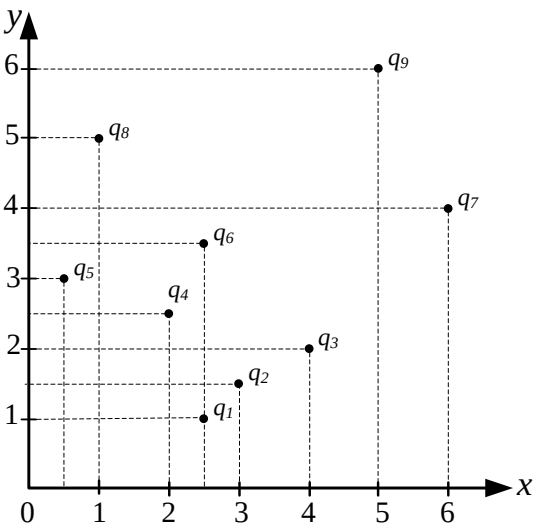
б) $a_1 = (1,1)$, $a_2 = (1,5)$, $a_3 = (4,6)$, $a_4 = (4,4)$, $a_5 = (5,3)$, $a_6 = (6,3)$, $a_7 = (7,4)$, $a_8 = (2,4)$;

в) $a_1 = (3,6)$, $a_2 = (2,1)$, $a_3 = (5,2)$, $a_4 = (5,5)$, $a_5 = (1,3)$, $a_6 = (6,4)$, $a_7 = (8,4)$;

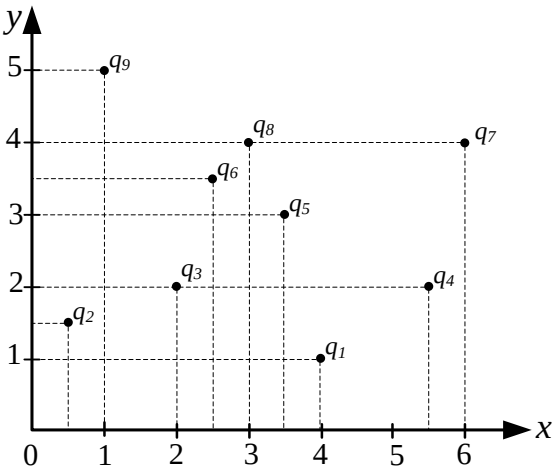
г)



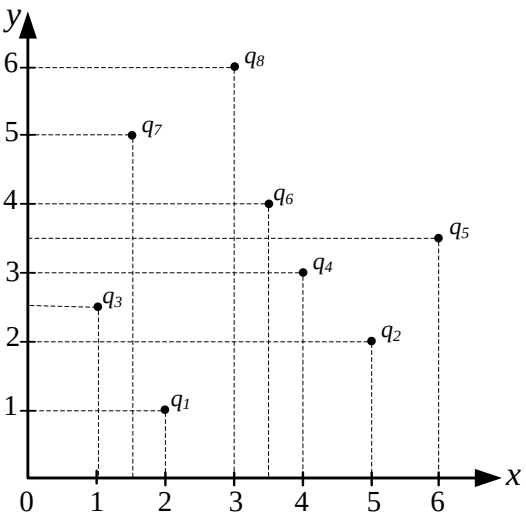
д)



е)



є)



1.9. Тести

Подані нижче тести можна використовувати як у цілому за курсом, так і за окремими темами.

Тести до теми «Вступ до структур даних та алгоритмів»

1. Книга Ніклауса Вірта називається:
   а) «Алгоритми + Структури даних = Програмне забезпечення»;
   б) «Алгоритми + Структури даних = Програми»;
   в) «Алгоритми + Мови = Програмне забезпечення»;
   г) «Алгоритми + Налагодження = Програми».
2. У сучасних комп’ютерах структури даних, як правило, подані у вигля-

ді (з точки зору програміста):

а) організованих проводів;

б) послідовності областей напівпровідників p- і n-типів в транзисторах;

в) послідовностей двійкових цифр;

г) квантових станів частинок;

3. Логічна структура даних – це:

а) послідовна, несуперечлива структура даних;

б) спосіб подання даних у пам’яті;

в) вивчення структури даних без урахування її подання в пам’яті комп’ютера;

г) структура даних, придатна (підходяща) для булевої алгебри.

4. Різниця між простими і інтегрованими структурами даних:

а) прості структури даних оперують простими числами, а інтегровані – цілими;

б) прості структури даних вже реалізовані в більшості сучасних мов програмування;

в) прості структури даних забезпечують швидкий доступ до даних;

г) інтегровані структури даних містять інші структури даних.

5. Різниця між зв’язаними і незв’язаними структурами:

а) зв’язані структури мають явні зв’язки між елементами даних;

б) зв’язані структури мають мережеве з’єднання;

в) зв’язані структури мають ребра;

г) зв’язані структури основані на Гарвардській архітектурі.

6. Прикладом динамічної структури є:

а) ціле число;

б) вектор;

в) двостороння черга;

г) граф.

7. Прикладом простої структури є:

а) ціле число;

б) вектор;

в) двостороння черга;

г) граф.

8. Прикладом статичної структури є:
   а) ціле число;
   б) вектор;
   в) двостороння черга;
   г) граф.
9. Прикладом напівстатичної структури є:
   а) ціле число;
   б) вектор;
   в) двостороння черга;
   г) граф.
10. Прикладом нелінійної структури є:
   а) ціле число;
   б) вектор;
   в) двостороння черга;
   г) граф.
11. Прикладом інтегрованої структури є:
   а) ціле число;
   б) вектор;
   в) двостороння черга;
   г) граф.
12. Перший алгоритм, який дійшов до нас:
   а) Аристотеля;
   б) Евкліда;
   в) Фалеса;
   г) Піфагора.
13. Автор теореми про неповноту символічної логіки:
   а) Георг Гендель;
   б) Георг Гегель;
   в) Ернст Геккель;
   г) Курт Гедель.
14. Який термін не є формалізацією алгоритму:
   а) лямбда-обчислення Черча;
   б) машина Тюрінга;
   в) машина Поста;
   г) машина Беббіджа?

15. Різниця між частковим та повним алгоритмами:

а) частковий алгоритм може містити блоки на людських мовах;

б) частковий алгоритм може вирішити лише частину задач даного класу;

в) частковий алгоритм може працювати швидше для певних вхідних даних;

г) повний алгоритм дає повний розв’язок.

16. «Алгоритм повинен містити скінченне число елементарних здійснених приписів» є вимогою:

а) правильності;

б) скінченності запису;

в) універсальності;

г) скінченності дій.

17. «Алгоритм повинен виконувати скінченне число кроків при вирішенні проблеми» є вимогою:

а) правильності;

б) скінченності запису;

в) універсальності;

г) скінченності дій.

18. «Алгоритм повинен бути однаковим для всіх допустимих вхідних даних» є вимогою:

а) правильності;

б) скінченності запису;

в) універсальності;

г) скінченності дій.

19. «Алгоритм повинен надавати правильний відносно поставленої задачі розв’язок» є вимогою:

а) правильності;

б) скінченності запису;

в) універсальності;

г) скінченності дій.

20. NP-повна задача:

а) не може бути розв’язана за допомогою алгоритму;

б) може бути розв’язана тільки повним алгоритмом;

в) не може бути перевіреною за допомогою поліноміального алгорит-

му;

г) забирають значний час для великої розмірності вхідних даних.

Тести до теми «Базові структури даних»

1. Словником називається структура даних, яка підтримує такі операції:

а) додавання елементів, вибір і видалення найменшого елемента;

б) перетин і об’єднання елементів;

в) додавання і видалення елементів, перевірка належності до множини даного елемента;

г) пошук найменшого і найбільшого елемента, додавання і видалення елементів.

2. Типова операція з множинами $Successor(S; x)$:

а) повертає вказівник на наступний за елементом x елемент ни S;

б) повертає вказівник на елемент, що безпосередньо передує ту x;

в) видаляє з множини S елемент, який безпосередньо передує елементу x.

3. Стеки та черги – це:

а) статичні множини;

б) динамічні множини;

в) нелінійні структури;

г) інтегровані структури.

4. Лінійний список, в якому можна видалити тільки той елемент, який був у нього доданий останнім, називається:

а) чергою;

б) деком;

в) стеком;

г) масивом.

5. Структура даних, яка організована за принципом FIFO (перший прийшов – перший пішов), називається:

а) чергою;

б) деком;

в) стеком;

г) масивом.

6. Структура даних, яка організована за принципом LIFO (останній прийшов – перший пішов), називається:

а) чергою;

б) деком;

в) стеком;

г) масивом.

7. Які позиції стека доступні для занесення нових елементів:

а) вершина і дно;

б) вершина;

в) дно;

г) будь-яка, крім дна.

8. Стеком називається структура даних, організована за принципом:

а) «перший прийшов – перший пішов»;

б) «останній прийшов – перший пішов»;

в) «перший прийшов – останній пішов»;

г) «останній прийшов – останній пішов».

9. Який елемент можна видалити з черги:

а) будь-який;

б) один з крайніх елементів;

в) елемент, який доданий першим;

г) елемент, який доданий останнім?

10. Яке поле відсутнє в односторонньо зв’язаному списку:

а) $next$;

б) $prev$?

11. Як розташовані елементи у впорядкованому списку:

а) в порядку зменшення ключів;

б) в порядку зростання ключів;

в) обидва варіанти правильні?

12. В кільцевому списку:

а) поле $prev$ голови списку вказує на хвіст списку, поле $next$ хвоста списку відсутнє;

б) поле $next$ хвоста списка вказує на голову списку, поле $prev$ голови списку відсутнє;

в) поле $prev$ голови списку вказує на хвіст списку, а поле $next$ хвоста списку вказує на голову списку.

13. У гіршому випадку пошук в хеш-таблиці може займати:

а) $O(n^2)$;

б) $O(\lg n)$;

в) $O(n)$.

14. Побудова хеш-функції методом ділення з остачею відбувається за формулою (m – число можливих хеш-значень, k – ключ):

а) $h(k) = \frac{k}{m}$;

б) $h(k) = k \ mod \ m$;

в) $h(k) = k \ div \ m$;

г) $h(k) = m \ div \ k$.

15. Яка формула використовується для побудови хеш-функції методом множення (m – число можливих хеш-значень, k – ключ, A – константа, $0 < A < 1$):

а) $h(k) = \mathrm{A}km$;

б) $h(k) = k \ mod \ \mathrm{A}m$;

в) $h(k) = \ (kA \ mod \ m)$;

г) $h(k) = [m \ (kA \ mod \ 1)]$?

16. Висота червоно-чорного дерева не більша за:

а) $O(n)$;

б) $O(\lg n)$;

в) $O(n^2)$.

17. Яка з нижче перерахованих властивостей не є RB-властивістю:

а) кожна вершина – або червона, або чорна;

б) кожен листок ($nill$) – чорний;

в) якщо вершина червона, обидва її потомки чорні;

г) якщо вершина червона, обидва її потомки червоні;

д) всі шляхи, що йдуть донизу від кореня до листків, мають однакову кількість чорних вершин?

Тести до теми «Математичні основи аналізу алгоритмів»

1. Запис $T(n) = \theta(n^2)$ означає:

а) знайдуться такі константи $c_1, c_2 > 0$, що $c_1 n^2 \leq T(n) \leq c_2 n^2$ при

всіх $n \geq 0$;

б) знайдуться такі константи $c_1, c_2 > 0$ і таке n_0, що $c_1 n^2 \leq T(n) \leq c_2 n^2$ при всіх $n \geq n_0$;

в) знайдеться така константа $c > 0$ і таке n_0, що $T(n) \geq c_2 n^2$ при всіх $n \geq n_0$;

г) знайдеться така константа $c \geq 0$ і таке n_0, що $cn^2 \leq T(n)$ при всіх $n \geq n_0$.

2. Запис $f(n) = O(\mathrm{g}(n))$ означає:

а) знайдуться такі константи $c_1, c_2 > 0$ і таке n_0, що $c_1 g(n) \leq f(n) \leq c_2 g(n)$ для всіх $n \geq n_0$;

б) знайдеться така константа $c > 0$ і таке n_0, що $0 \leq f(n) \leq cg(n)$ для всіх $n \geq n_0$;

в) знайдеться така константа $c > 0$ і таке n_0, що $0 \leq cg(n) \leq f(n)$ для всіх $n \geq n_0$.

3. Запис $f(n) = \theta(\mathrm{g}(n))$ означає, що

а) знайдуться такі константи $c_1, c_2 > 0$ і таке n_0, що $c_1 g(n) \leq f(n) \leq c_2 g(n)$ для всіх $n \geq n_0$;

б) знайдеться така константа $c > 0$ і таке n_0, що $0 \leq f(n) \leq cg(n)$ для всіх $n \geq n_0$;

в) знайдеться така константа $c > 0$ і таке n_0, що $0 \leq cg(n) \leq f(n)$ для всіх $n \geq n_0$.

4. Запис $f(n) = \Omega(\mathrm{g}(n))$ означає, що

а) знайдуться такі константи $c_1, c_2 > 0$ і таке n_0, що $c_1 g(n) \leq f(n) \leq c_2 g(n)$ для всіх $n \geq n_0$;

б) знайдеться така константа $c > 0$ і таке n_0, що $0 \leq f(n) \leq cg(n)$ для всіх $n \geq n_0$;

в) знайдеться така константа $c > 0$ і таке n_0, що $0 \leq cg(n) \leq f(n)$ для всіх $n \geq n_0$.

5. Запис $f(n) = o(\mathrm{g}(n))$ означає:

а) якщо для будь-якого додатного ε знайдеться таке n_0, що $f(n) \leq g(n) \leq \varepsilon f(n)$ при всіх $n \geq n_0$;

б) якщо для будь-якого додатного ε знайдеться таке n_0, що $\mathrm{g}(n) \leq f(n) \leq \varepsilon \mathrm{g}(n)$ при всіх $n \geq n_0$;

в) якщо для будь-якого додатного ε знайдеться таке n_0, що

$0 \le f(n) \le \varepsilon \mathrm{g}(n)$ при всіх $n \ge n_0$;

г) якщо для будь-якого додатного ε знайдеться таке n_0, що $0 \le \varepsilon \mathrm{g}(n) \le f(n)$ при всіх $n \ge n_0$.

6. Запис $f(n) = w(\mathrm{g}(n))$ означає:

а) якщо для будь-якого додатного ε знайдеться таке n_0, що $f(n) \le g(n) \le \varepsilon f(n)$ при всіх $n \ge n_0$;

б) якщо для будь-якого додатного ε знайдеться таке n_0, що $g(n) \le f(n) \le \varepsilon \mathrm{g}(n)$ при всіх $n \ge n_0$;

в) якщо для будь-якого додатного ε знайдеться таке n_0, що $0 \le f(n) \le \varepsilon \mathrm{g}(n)$ при всіх $n \ge n_0$;

г) якщо для будь-якого додатного ε знайдеться таке n_0, що $0 \le \varepsilon \mathrm{g}(n) \le f(n)$ при всіх $n \ge n_0$.

7. Запис $f(n) = O(\mathrm{g}(n))$ означає:

а) верхню оцінку;

б) нижню оцінку;

в) верхню і нижню оцінки.

8. Запис $f(n) = \theta(\mathrm{g}(n))$ означає:

а) верхню оцінку;

б) нижню оцінку;

в) верхню і нижню оцінки.

9. Запис $f(n) = \Omega(\mathrm{g}(n))$ означає:

а) верхню оцінку;

б) нижню оцінку;

в) верхню і нижню оцінки.

10. Як за швидкістю зростання співвідносяться функції $f(n) = a^n$ і $\mathrm{g}(n) = n!$:

а) $f(n) = o(\mathrm{g}(n))$;

б) $\mathrm{g}(n) = o(f(n))$?

11. Як за швидкістю зростання співвідносяться функції $p(n) = \sum_{i=0}^{d} a_i n_i$ і $\mathrm{g}(n) = n!$:

а) $p(n) = o\big(\mathrm{g}(n)\big)$;

б) $\mathrm{g}(n) = o(p(n)$?

12. Як за швидкістю зростання співвідносяться функції $p(n) = \sum_{i=0}^{d} a_i n_i$ і $\mathrm{g}(n) = a^n$:

а) $p(n) = o\big(\mathrm{g}(n)\big)$;

б) $\mathrm{g}(n) = o(p(n)$?

13. Як за швидкістю зростання співвідносяться функції $f(n) = \log_a n$ і $g(n) = n!$:

а) $f(n) = o\big(\mathrm{g}(n)\big)$;

б) $g(n) = o\big(f(n)\big)$?

14. Як за швидкістю зростання співвідносяться функції $p(n) = \sum_{i=0}^{d} a_i n_i$ і $f(n) = \log_a n$:

а) $p(n) = o\big(f(n)\big)$;

б) $f(n) = o\big(p(n)\big)$?

15. Які позначення використовуються для асимптотичних оцінок складності:

а) $O, \Omega, \Theta, o, v, w$;

б) $O, \Omega, \Theta, W, o, w$;

в) O, Ω, Θ, o;

г) $O, \Omega, \Theta, o, q, w$?

Тести до теми «Алгоритми сортування, злиття та пошуку»

1. Математичне очікування часу роботи якого алгоритму дає $O(n \lg n)$:

а) сортування вставками;

б) швидке сортування;

в) сортування вибором?

2. Час роботи процедури об’єднання двох впорядкованих масивів становить:

а) $\Theta(n)$;

б) $\Theta(n)^2$;

в) $\Theta(n \lg n)$;

г) $O(n^2)$;

д) $O(n \lg n)$.

3. Алгоритм лінійного пошуку працює в гіршому випадку за час:

а) $O(n)$;

б) $O(1)$;

в) $O(n^2)$;

г) $\Theta(n \lg n)$.

4. Бінарний пошук проводиться:

а) в масиві цілих невід'ємних чисел;

б) масиві цілих чисел;

в) відсортованому масиві;

г) не відсортованому масиві.

5. Час роботи алгоритму бінарного пошуку:

а) $\Theta(n)$;

б) $\Theta(n)^2$;

в) $O(\lg n)$;

г) $O(n^2)$;

д) $O(n \lg n)$.

6. Яке з тверджень є правильним:

а) метод двійкового пошуку можна використовувати для задачі пошуку x на відрізку $[a, b]$, для деякої монотонно зростаючої функції $f(x)$, де $f(x) = y$;

б) метод двійкового пошуку можна використовувати для задачі пошуку x на відрізку $[0, 1]$, для деякої функції $f(x)$, що набуває невід'ємних значень, де $f(x) = y$;

в) є правильними твердження а) і б);

г) твердження а) і б) неправильні?

7. Говорять, що алгоритм сортування оснований на порівняннях, якщо:

а) він використовує внутрішню структуру елементів, що сортуються, та порівнює їх і після деякої кількості порівнянь дає відповідь;

б) він ніяк не використовує внутрішню структуру елементів, що сортуються, а лише порівнює їх і після деякої кількості порівнянь дає відповідь.

8. Дерево дозволу для алгоритму сортування вставками, утворене послідовністю з чотирьох елементів, має листків не менше:

а) 8;

б) 16;

в) 20;

г) 24;

д) 28.

9. Що таке висота дерева дозволу:

а) максимальна довжина шляху в цьому дереві від кореня до листка;

б) мінімальна довжина шляху в цьому дереві від кореня до листка;

в) максимальна кількість листків;

г) максимальна кількість вузлів?

10. Висота будь-якого дерева дозволу алгоритму сортування n елементів дорівнює:

а) $\Omega(n \lg n)$;

б) $O(n^2)$;

в) $\Theta(n \lg n)$;

г) $\Omega\left(\frac{n^2}{2}\right)$;

д) $O(\lg n)$.

11. Сортування підрахунком можна використовувати, якщо кожний з n елементів послідовності, яку необхідно відсортувати:

а) є цілим числом з відомого діапазону (що не перевищує наперед відоме k);

б) є будь-яким числом з відомого діапазону (що не перевищує наперед відоме k);

в) є будь-яким числом;

г) є цілим додатним числом з відомого діапазону (що не перевищує наперед відоме k);

д) є цілим невід'ємним числом з відомого діапазону (що не перевищує наперед відоме k).

12. Алгоритм сортування підрахунком має властивість стійкості, що означає:

а) якщо у вхідному масиві є декілька рівних чисел, то у вихідному масиві їх відносне розташування не важливе;

б) якщо у вхідному масиві є декілька рівних чисел, то у вихідному масиві вони розташовані в тому ж порядку, що й у вхідному;

в) у вхідному масиві немає рівних чисел.

13. Час роботи алгоритму вставками залежить:

а) тільки від розміру вихідного масиву;

б) від розміру вихідного масиву і порядку його елементів.

14. Алгоритм сортування вставками:

а) не використовує додатковий масив та змінні;

б) використовує додатковий масив та змінні;

в) використовує лише додатковий масив;

г) використовує лише додаткові змінні, але не використовує додатковий масив.

15. Виконується сортування 5-ти елементів. Скільки операцій порівняння буде виконано:

а) 7;

б) 30;

в) 15;

г) 10;

д) залежить від того, як спочатку був відсортований масив?

16. Який пошук є ефективнішим:

а) лінійний;

б) бінарний;

в) алгоритми однакові за ефективністю?

Тести до теми «Алгоритмічні стратегії»

1. Динамічне програмування:

а) корисне, коли підзадачі, які зустрічаються на різних шляхах розв’язання задачі, не повторюються;

б) корисне, коли на різних шляхах багатократно зустрічаються одні й ті самі задачі.

2. У задачі про маршрут на прямокутному полі числа на дошці можуть бути:

а) додатними;

б) цілими невід’ємними;

в) від’ємними;

г) будь-якими.

3. Рекурентна формула для розв'язання задачі про маршрут на прямокутному полі, що використовується для клітинок i, j $(i > 1, j > 1)$:

а) $m[i,j] = a[i,j] + \min\{m[i-1,j], m[i,j-1]\}$;

б) $m[i,j] = a[i,j] + \max\{m[i-1,j], m[i,j-1]\}$;

в) $m[i,j] = \max\{m[i-1,j], m[i,j-1]\}$;

г) немає правильного варіанта.

4. Послідовність також вважається своєю підпослідовністю. Дане твердження є:

а) правильним;

б) неправильним.

5. Чи має задача про найбільшу спільну підпослідовність властивість оптимальності для підзадач?

а) так;

б) ні?

6. Задача про вибір заявок полягає в тому, щоб:

а) відібрати максимальну кількість заявок;

б) відібрати максимальну кількість сумісних одна з одною заявок;

в) відібрати заявки таким чином, щоб час простою аудиторії був мінімальним;

г) відібрати заявки таким чином, щоб сумарний час їх виконання був максимальним.

7. Жадібний алгоритм:

а) дає оптимальне розв'язання для всіх задач;

б) дає оптимальне розв'язання для деяких задач;

в) дає наближене розв'язання.

8. Для задачі про вибір заявок жадібний алгоритм дає:

а) оптимальне рішення;

б) локально-оптимальне рішення;

в) наближене рішення.

9. Укажіть на правильну відповідь для задачі про вибір заявок, яка розв'язується жадібним алгоритмом, якщо вихідна множина заявок подана таким чином: $a_1 = [1,4)$; $a_{12} = [1,3)$; $a_3 = [5,6)$; $a_4 = [7,9)$; $a_5 = [8,9)$; $a_6 = [10,11)$:

а) $a_1 a_3 a_4 a_6$;

б) $a_2a_3a_4a_6$;

в) $a_1a_3a_5a_6$;

г) є правильними а) та б);

д) є правильними а) та в);

д) немає правильної відповіді.

10. Задана послідовність $ADDFGABC$. Яка з перелічених послідовностей не є підпослідовністю даної:

а) $ADFGBC$;

б) $ADDFGABC$;

в) $DDFGABCA$;

г) {}?

11. Амортизаційний аналіз – це засіб аналізу алгоритмів, які виконують:

а) неповторювані операції;

б) типові операції;

в) будь-які операції (повторювані або ні).

12. При амортизаційному аналізі повинна виконуватись умова:

а) для будь-якої послідовності операцій фактичний сумарний час тривалості всіх операцій не перевищує суму їх облікових вартостей;

б) для будь-якої послідовності операцій фактичний сумарний час тривалості всіх операцій більший від суми їх облікових вартостей;

в) для будь-якої послідовності операцій фактичні сумарний час тривалості всіх операцій менший від суми їх облікових вартостей.

13. Яка з операцій видаляє вершину із стека:

а) $Push\ (S)$;

б) $Push\ (S, x)$;

в) $Pop\ (S)$;

г) $Pop\ (S, x)$;

d) $Multipush\ (S, k)$?

14. У методі передоплати кожній операції присвоюється своя облікова вартість, причому ці вартості:

а) менші від реальних;

б) більші від реальних;

в) як більші, так і менші від реальних.

15. Чи є послідовність BCD найбільшою спільною підпослідовністю по-

слідовностей $BACD$ та $ABCAD$?

а) так;

б) ні.

Тести до теми «Основи теорії обчислюваності, машина Тюрінга»

1. Алгоритмічна нерозв'язність задачі того чи іншого класу означає:

а) неможливість розв'язання будь-якої конкретної задачі даного класу;

б) неможливість розв'язання всіх задач даного класу тим самим прийомом.

2. Першою фундаментальною теоретичною роботою, пов'язаною з доказом алгоритмічної нерозв'язності, була робота:

а) Алана Тюрінга;

б) Курта Геделя;

в) Еміля Поста;

г) Дональда Кнута.

3. До якої класичної задачі прийнято зводити розглянуту задачу при доведенні її алгоритмічної нерозв'язності:

а) неповноти символічних логік;

б) десятої проблеми Гільберта;

в) задачі зупину?

4. Множина A натуральних чисел m зводиться до іншої множини B натуральних чисел, якщо існує всюди визначена функція, яку можна обчислити, $N \to N$ з такою властивістю:

а) $x \in B \Rightarrow f(x) \in A$, для всіх $x \in N$;

б) $x \in A \Rightarrow f(x) \in B$, для всіх $x \in N$;

в) $x \in A \Leftrightarrow f(x) \in B$, для всіх $x \in N$;

г) $x \in B \Leftrightarrow f(x) \in A$, для всіх $x \in N$.

5. B зводиться до A, якщо існує алгоритм , який дозволяє множину B за умови, що йому наданий доступ до оракула, що відповідає на запитання про множину A. Даний вид приведення називається:

а) приведення за Тюрінгом;

б) приведення за Постом;

в) приведення за Генделем;

г) приведення за Герцбергом.

6. Множина натуральних чисел X називається розв'язною множиною, якщо існує алгоритм, який:

а) за будь-яким натуральним n визначає, чи належить воно до множини X;

б) ставить у відповідність кожному $x \in X$ обчислювану функцію;

в) будь-яка множина натуральних чисел є розв'язною.

7. Літера «m» в назві m-звідної функції множини A до B походить від терміну:

а) max-over-reducibility;

б) max-one-reducibility;

в) min-over-reducibility;

г) min-one-reducibility;

д) many-over-reducibility;

е) many-one-reducibility.

8. Проблема еквівалентності алгоритмів:

а) за двома довільно заданими алгоритмами визначити, чи видаватимуть вони однаковий вихідний результат на заданих вихідних даних;

б) за двома довільно заданими алгоритмами визначити, чи видаватимуть вони однаковий вихідний результат на будь-яких вихідних .

9. Як називається множина станів, в яких може знаходитись пристрій керування машини Тюрінга:

а) зовнішній алфавіт;

б) внутрішній алфавіт;

в) конфігурація;

г) таблиця станів?

10. Що таке конфігурація машини Тюрінга:

а) внутрішній та зовнішній алфавіти, початковий символ, заключний стан;

б) вміст стрічки, поточна позиція головки, таблиця станів;

в) послідовність станів усіх комірок стрічки і стан пристрою керування?

11. Як називається стандартна машина Тюрінга, якщо вона здатна виконувати лише команди виду: $q_\alpha 0 q_\beta u T$ і $q_\alpha 1 q_\beta 1 T$, де $u = \{0,1\}$, $T = \{Л.П.С\}$:

а) нестираючою;

б) універсальною;

в) багатострічковою;

г) квантовою?

12. Яка з операцій не належить до основних операцій композиції машин Тюрінга:

а) множення;

б) піднесення до степеня;

в) додавання;

г) ітерація?

13. Машина Тюрінга називається стандартною, якщо:

а) її зовнішній алфавіт складається з 0 та 1;

б) вона здійснює операції додавання та множення;

в) вона при зсуві стрічки може попередньо змінювати стан комірки, що сприймається.

14. Десята проблема Гільберта: Нехай заданий багаточлен n-го степеня p _____, чи існує алгоритм, який визначає, чи має рівняння $p = 0$ розв'язок _____?

а) _____, в цілих числах;

б) з цілими коефіцієнтами, в цілих числах;

в) з дійсними коефіцієнтами, в цілих числах;

г) з довільними коефіцієнтами, в цілих числах;

г) слова не пропущені.

15. Яка загальна причина поєднує такі проблеми: розподіл дев'яток у запису числа π, обчислення досконалих чисел, десята проблема Гільберта:

а) логічна нерозв'язність;

б) відсутність загального методу розв'язання задач?

Тести до теми «Генератори псевдовипадкових чисел»

1. Який з методів не використовується для генерування псевдовипадкових послідовностей:

а) лінійний конгруентний метод;

б) нелінійний конгруентний метод;

в) регістр зсуву з узагальненим зворотним зв'язком;

г) регістр зсуву з лінійним зворотним зв'язком?

2. Укажіть формулу, за якою обчислюються члени лінійної рекурентної послідовності (лінійний конгруентний метод):

а) $X_{k+1} = X_k \bmod(am + c)$;

б) $X_{k+1} = aX_k \bmod m^c$;

в) $X_{k+1} = (aX_k + c) \bmod m$;

г) $X_{k+1} = aX_k \bmod cm$.

3. Лінійний конгруентний метод:

а) застосовується в простих випадках;

б) має криптографічну стійкість;

в) застосовується в складних випадках.

4. Послідовність чисел, які обчислюються за допомогою лінійного конгруентного методу, є періодичною з періодом:

а) рівним m;

б) що не перевищує m;

в) більшим за m.

5. Яким чином необхідно вибрати m при реалізації лінійного конгруентного методу:

а) $m = \lg 2^e$;

б) $m = 2^{\lg 2^E}$;

в) $m = \lg 2^e$;

г) $m = 2^e$?

6. Укажіть на формулу, на якій оснований датчик Фібоначчі:

а) $X_k = \begin{cases} X_{k-a} - X_{k-b}, если\ \ X_{k-a} \geq X_{k-b} \\ X_{k-a} - X_{k-b} + 1, если\ \ X_{k-a} < X_{k-b} \end{cases}$;

б) $X_k = \begin{cases} X_{k-a} - X_{k-b}, если\ \ X_{k-a} > X_{k-b} \\ X_{k-a} - X_{k-b} + 1, если\ \ X_{k-a} \leq X_{k-b} \end{cases}$;

в) $X_k = \begin{cases} X_{k-a} - X_{k-b} + 1, если\ \ X_{k-a} \geq X_{k-b} \\ X_{k-a} - X_{k-b}, если\ \ X_{k-a} < X_{k-b} \end{cases}$;

г) $X_k = \begin{cases} X_{k-a} + X_{k-b}, если\ \ X_{k-a} \geq X_{k-b} \\ X_{k-a} - X_{k-b} + 1, если\ \ X_{k-a} < X_{k-b} \end{cases}$.

7. У методі Фібоначчі із затримкою числа X_k:

а) дійсні числа з діапазону $[-1, 1]$;

б) дійсні числа з діапазону $[0, 1)$;

в) дійсні числа з діапазону $[0, 1]$;

г) додатні цілі числа;

д) цілі числа.

8. Лаги a і b в методі Фібоначчі із затримкою:

а) додатні числа;

б) цілі додатні числа;

в) невід'ємні числа;

г) цілі невід'ємні числа.

9. Генератор псевдовипадкових чисел – це алгоритм, що породжує послідовність чисел, елементи якої:

а) незалежні одне від одного;

б) майже незалежні одне від одного і підпорядковуються заданому розподілу;

в) майже незалежні одне від одного і не підпорядковуються заданому розподілу.

10. Детерміновані алгоритми:

а) можуть генерувати повністю випадкові числа;

б) можуть лише апроксимувати деякі властивості випадкових чисел.

11. Укажіть на значення ймовірності, яке повинно бути отримано в ході частотного блочного тесту, щоб двійкова послідовність була істинно випадковою:

а) $p \geq 0{,}01$;

б) $p < 0{,}01$;

в) $p \geq 0{,}001$;

г) $p < 0{,}001$.

12. Суть частотного побітового тесту:

а) визначення співвідношення між нулями і одиницями у всій двійковій послідовності;

б) визначення найдовшого ряду одиниць всередині всієї послідовності;

в) визначення найдовшого ряду нулів всередині всієї послідовності.

13. Укажіть на тест пакета NIST, який необхідно провести першим (усі

наступні проводяться за умови, що пройдений даний):

а) частотний блочний тест;

б) на найдовшу послідовність одиниць в блоці;

в) частотний побітовий тест;

г) на послідовність однакових бітів.

14. Тести на випадковість поділяються на дві групи:

а) статичні та ймовірнісні;

б) статичні та графічні;

в) статичні, графічні та ймовірнісні.

15. Яке значення m необхідно взяти, щоб частотний блочний тест став частотним побітовим тестом:

а) $m = 2^e$;

б) $m = \lg 2^e$;

в) $m = 2$;

г) $m = 1$?

Тести до теми «Фундаментальні алгоритми на графах та деревах»

1. За допомогою списків суміжних вершин можна подати:

а) орієнтований граф;

б) неорієнтований граф;

в) як орієнтований граф, так і неорієнтований граф.

2. Довжина всіх списків суміжних вершин в орієнтованому і неорієнтованому графах з n ребрами:

а) більша для орієнтованого графа;

б) більша для неорієнтованого графа;

в) однакова.

3. Який спосіб подання є кращим для щільного графа:

а) набір списків суміжних вершин;

б) матриця суміжності;

в) обидва варіанти рівнозначні?

4. Яка структура даних краще підійде для зберігання розрідженого графа:

а) матриця інцидентності;

б) масив;

в) список?

5. Цикл в орієнтованому графі називається простим:

а) якщо він містить усі вершини графа не більше ніж один раз;

б) якщо в ньому немає однакових вершин (окрім першої та останньої);

в) якщо він містить усі ребра графа не більше ніж один раз.

6. Вершинам графа можна поставити у відповідність:

а) множини;

б) відношення між об’єктами;

в) об’єкти;

г) зв’язки.

7. Неорієнтований граф:

а) може містити ребра-цикли;

б) не може містити ребра-цикли.

8. Дві мітки часу в кожної вершини використовуються:

а) у пошукові в глибину;

б) у пошукові в ширину;

в) у пошуках у глибину і в ширину.

9. В алгоритмі в ширину сірі вершини:

а) не можуть мати суміжні не знайдені вершини;

б) можуть мати суміжні не знайдені вершини.

10. У пошуку в ширину відстанню вважається:

а) довжина максимального шляху з початкової вершини до поточної;

б) довжина мінімального шляху з початкової вершини до поточної;

в) кількість проміжних вершин між початковою і поточною вершинами.

11. Алгоритм пошуку в глибину ставить на вершинах:

а) мітку часу знаходження вершини;

б) час, коли вершина зафарбовується в чорний колір;

в) дві мітки часу: перша – коли вершина була знайдена, друга – коли вершина стала чорною;

г) дві мітки часу: перша – коли вершина стала сірою, друга – коли вершина стала чорною;

д) три мітки часу: перша – коли вершина була знайдена, друга – коли вершина стала сірою, третя – коли вершина стала чорною.

12. Алгоритм пошуку в глибину використовується:

а) в орієнтованому графі;

б) неорієнтованому графі;

в) орієнтованому та неорієнтованому графах.

13. Час процедури пошуку в глибину становить:

а) $\Theta(V)$;

б) $\Theta(V + E)$;

в) $\Theta(E)$;

г) $\Theta(V^2)$;

д) $\Theta(E^2)$.

14. Задача топологічного сортування використовується для:

а) орієнтованого графа з циклами і без;

б) орієнтованого графа без циклів;

в) неорієнтованого графа;

г) повного графа.

15. Топологічне сортування виконується за час:

а) $\Theta(V + E)$;

б) $O(V^2)$;

в) $\Omega(V + E)$;

г) $O(V \times V + E)$.

Тести до теми «Геометричні алгоритми»

1. Векторний добуток двох векторів p_1 і p_2 визначається:

а) як площа паралелограма, утвореного точками $(0, 0)$, p_1, p_2 і $(p_1 \times p_2)$;

б) як визначник матриці $p_1 \times p_2 = \det\begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix}$;

в) обидва варіанти а) і б) є правильними;

г) немає правильної відповіді.

2. Обмежуючим прямокутником геометричної фігури називають:

а) будь-який з прямокутників зі сторонами, паралельними осям координат, що містить дану фігуру;

б) найбільший з прямокутників зі сторонами, паралельними осям координат, що містить дану фігуру;

в) найменший з прямокутників зі сторонами, паралельними осям координат, що містить дану фігуру;

г) найменший з прямокутників, що містить дану фігуру;

д) найбільший з прямокутників, що містить дану фігуру.

3. Якщо значення $p_1 \times p_2$ є додатним, то найкоротший поворот p_2 відносно (0, 0), що з'єднує його з p_1, проходить:

а) за годинниковою стрілкою;

б) проти годинникової стрілки.

4. Якщо значення $p_1 \times p_2$ є від'ємним, то найкоротший поворот p_2 відносно (0, 0), що з'єднує його з p_1, проходить:

а) за годинниковою стрілкою;

б) проти годинникової стрілки.

5. В який бік необхідно повернути в точці p_1, рухаючись по ламаній $p_0 p_1 p_2$, якщо векторний добуток $(p_2 - p_0) \times (p_1 - p_0)$ є додатним:

а) направо;

б) наліво;

в) рухатись прямо;

г) повернути кругом;

д) або рухатись прямо, або повернути кругом?

6 В який бік необхідно повернути в точці p_1, рухаючись по ламаній $p_0 p_1 p_2$, якщо векторний добуток $(p_2 - p_0) \times (p_1 - p_0)$ дорівнює нулю?

а) направо;

б) наліво;

в) рухатись прямо;

г) повернути кругом;

д) або рухатись прямо, або повернути кругом.

7. В який бік необхідно повернути в точці p_1, рухаючись по ламаній $p_0 p_1 p_2$, якщо векторний добуток $(p_2 - p_0) \times (p_1 - p_0)$ є від'ємним:

а) направо;

б) наліво;

в) рухатись прямо;

г) повернути кругом;

д) або рухатись прямо, або повернути кругом?

8. Опуклою оболонкою скінченної множини точок Q називається _____ опуклий багатокутник:

а) найбільший;

б) найменший;

в) будь-який.

9. Опукла оболонка скінченної множини точок Q містить усі точки з даної множини, при цьому:

а) деякі з точок можуть бути всередині багатокутника, деякі будуть його вершинами;

б) всі точки знаходяться всередині багатокутника;

в) деякі з точок можуть бути всередині багатокутника, деякі – на його сторонах, деякі будуть його вершинами.

10. Перегляд за Грехемом потребує часу:

а) $O(n^2)$;

б) $O(n \lg n)$;

в) $O(n \lg h)$;

г) $O(nh)$,

де n – число точок, а h – число вершин опуклої оболонки.

11. Прохід за Джарвісом потребує часу:

а) $O(n^2)$;

б) $O(n \lg n)$;

в) $O(n \lg h)$;

г) $O(nh)$,

де n – число точок, а h – число вершин опуклої оболонки.

12. Прохід за Джарвісом починається з точки:

а) з найбільшою ординатою (якщо їх декілька – то з найменшою серед них абсцисою);

б) з найменшою ординатою (якщо їх декілька – то з найменшою серед них абсцисою);

в) з найбільшою ординатою (якщо їх декілька – то з найбільшою серед

них абсцисою);

г) з найменшою ординатою (якщо їх декілька – то з найбільшою серед них абсцисою).

13. Умова $[(p_3-p_1)\times(p_2-p_1)][(p_4-p_1)\times(p_2-p_1)]\le 0$ означає:

а) точки p_3 і p_4 лежать по різні боки від прямої p_1p_2;

б) точки p_1 і p_2 лежать по різні боки від прямої p_3p_4;

в) відрізки p_1p_2 і p_3p_4 перетинаються;

г) немає правильної відповіді.

14. Умова $[(p_1-p_3)\times(p_4-p_3)][(p_2-p_3)\times(p_4-p_3)]\le 0$ означає:

а) точки p_3 і p_4 лежать по різні боки від прямої p_1p_2;

б) точки p_1 і p_2 лежать по різні боки від прямої p_3p_4;

в) відрізки p_1p_2 і p_3p_4 перетинаються;

г) немає правильної відповіді.

15. Який порядок обходу вершин використовується в проході за Джарвісом:

а) проти годинникової стрілки;

б) за годинниковою стрілкою;

в) можна рухатись в будь-який бік?

1.10. Індивідуальні домашні розрахункові завдання

1.10.1. Вимоги до виконання та оформлення розрахункових завдань

На початку навчального семестру кожен студент отримує комплексне індивідуальне розрахункове завдання.

Конкретні дані для кожного завдання необхідно брати строго у відповідності до свого варіанта (порядковий номер студента в журналі академічної групи).

Індивідуальне завдання друкується чи акуратно та рівно пишеться від руки на білих аркушах формату А4. Орієнтація аркуша вертикальна. Текст з одного боку аркуша. Поля мають бути не менше 2 см зліва, 1 см для інших.

Кожне завдання починається з нового аркуша. Спочатку вказується номер завдання. Далі наводиться повне і точне формулювання завдання та розв’язання. В розв’язанні повинні бути послідовно і з необхідними роз’ясненнями та ілюстраціями подані всі етапи розв’язання завдання.

Індивідуальне завдання повинно містити титульний аркуш заданого вигляду, і після підписання всіх завдань зшивається з лівого боку скріпками чи іншим способом, що гарантує його цілісність.

У разі невідповідності вказаним вимогам розрахункове завдання на перевірку не приймається.

Здача проводиться протягом семестру у відведений викладачем для цього час. Строк здачі – до початку залікового тижня.

1.10.2. Варіанти індивідуальних завдань

Варіант 1

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for k = 1 to n
    for i = 1 to n
        for j = 1 to n
            if a[i, k] + a[k, j] < a[i, j]
                a[i, j] := a[i, k] + a[k, j];
```

б)
```
d = b * b - 4 * a * c
s_d = sqrt(d)
x1 = (-b - s_d) / (2 * a)
x2 = (-b + s_d) / (2 * a).
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = log^k n,$$

$$f_2(n) = n^k.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^2 + 2n = O(n^2),$$

$$\log n = O(n).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $addaadfbcc$ та $cdabdfadcc$;

2) $daaefeefad$ та $deecaaaefa$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[12, 15); [1, 3); [14, 16); [4, 5); [10, 13); [7, 13); [8, 10); [5, 6); [7, 10); [5, 8);$

2) $[5, 6); [10, 16); [14, 18); [10, 12); [13, 16); [1, 5); [9, 11); [7, 12); [7, 8); [9, 12).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два

числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 5 9 1 7 1 7 4 8 2 4;

2) 9 6 2 0 5 0 8 7 4 5.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 20, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 17-ма вершинами та 24-ма ребрами, не більше 5-ти ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_{10}, y_{10}$. Знайти опуклу оболонку цієї множини точок.

Варіант 2

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for j = 1 to n
    for i = 1 to j - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t
        end;
```

б)
```
for k = 1 to n
  for i = 1 to n
    for j = 1 to n
      if a[i, k] + a[k, j] < a[i, j]
        a[i, j] := a[i, k] + a[k, j].
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = 3^k,$$
$$f_2(n) = \log(n^n).$$

Довести дані твердження, користуючись визначенням O-позначення:

$$\log n = O(n),$$
$$n^7 = O(e^n).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $daaefeefad$ та $deecaaaefa$;

2) $fbeaedbdcb$ та $bedfbbdecb$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[5,6);\ [10,16);\ [14,18);\ [10,12);\ [13,16);\ [1,5);\ [9,11);\ [7,12);\ [7,8);\ [9,12);$

2) $[14,17);\ [1,3);\ [10,11);\ [5,6);\ [6,9);\ [10,12);\ [13,18);\ [14,20);\ [7,11);\ [3,5).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 9 6 2 0 5 0 8 7 4 5;

2) 8 0 5 4 6 4 4 4 3 7.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 24, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 10-ма вершинами та 18-ма ребрами, не більше 4-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_{12}, y_{12}$. Знайти опуклу оболонку цієї множини точок.

Варіант 3

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for j = 1 to n
    for i = 1 to n - 1
        if a[i] > a[i + 1]
            t = a[i];
            a[i] = a[i + 1]
            a[i + 1] = t;
```

б)
```
d = b * b - 4 * a * c
s_d = sqrt(d)
x1 = (-b - s_d) / (2 * a)
x2 = (-b + s_d) / (2 * a).
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = log^k n,$$

$$f_2(n) = 3^k.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^2 + 2n = O(n^2),$$

$$n^{45} = O(n!).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $addaadfbcc$ та $cdabdfadcc$;

2) $dfaedfcfba$ та $abbaedbfba$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[12, 15)$; $[1, 3)$; $[14, 16)$; $[4, 5)$; $[10, 13)$; $[7, 13)$; $[8, 10)$; $[5, 6)$; $[7, 10)$; $[5, 8)$;

2) $[1, 3)$; $[4, 6)$; $[7, 12)$; $[13, 15)$; $[13, 16)$; $[8, 9)$; $[11, 15)$; $[7, 11)$; $[11, 12)$; $[3, 5)$.

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 5 9 1 7 1 7 4 8 2 4;

2) 2 9 1 9 7 2 0 5 7 8.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною,

що дорівнює 16, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 15-ма вершинами та 18-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \dots\ x_8, y_8$. Знайти опуклу оболонку цієї множини точок.

Варіант 4

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for i = 1 to n
   a[i] = i + i * i;
```

б)
```
for i = 1 to m do
     for j = 1 to (n div 2)
          t = a[i, j]
          a[i, j] = a[i, n - j]
          a[i, n - j] = t.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = 3^n,$$
$$f_2(n) = k^n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$\log n = O(n),$$
$$n^4 + n^2 = O(n^5).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $daaefeefad$ та $deecaaaefa$;

2) $fabcfbcdea$ та $dabfaccaad$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[5, 6);\ [10, 16);\ [14, 18);\ [10, 12);\ [13, 16);\ [1, 5);\ [9, 11);\ [7, 12);\ [7, 8);\ [9, 12);$

2) $[12, 15);\ [1, 3);\ [4, 6);\ [12, 13);\ [10, 12);\ [9, 11);\ [11, 17);\ [6, 12);\ [5, 8);\ [3, 5).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 9 6 2 0 5 0 8 7 4 5;

2) 5 5 8 1 7 3 5 3 3 1.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 20 різними генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 16-ма вершинами та 25-ма ребрами, не більше 5-ти ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_{10}, y_{10}$. Знайти опуклу оболонку цієї множини точок.

Варіант 5

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for i = 1 to m do
    for j = 1 to (n div 2) do begin
        t = a[i, j]
        a[i, j] = a[i, n - j]
        a[i, n - j] = t;
```

б)
```
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            if a[i, k] + a[k, j] < a[i, j]
                a[i, j] = a[i, k] + a[k, j].
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = log^k n,$$

$$f_2(n) = k^n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^2 + 2n = O(n^2),$$

$$n^7 = O(e^n).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $addaadfbcc$ та $cdabdfadcc$;

2) $fbeaedbdcb$ та $bedfbbdecb$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[12, 15);\ [1, 3);\ [14, 16);\ [4, 5);\ [10, 13);\ [7, 13);\ [8, 10);\ [5, 6);\ [7, 10);\ [5, 8);$

2) $[14, 17);\ [1, 3);\ [10, 11);\ [5, 6);\ [6, 9);\ [10, 12);\ [13, 18);\ [14, 20);\ [7, 11);\ [3, 5).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 5 9 1 7 1 7 4 8 2 4;

2) 8 0 5 4 6 4 4 4 3 7.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 24, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 14-ма вершинами та 16-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_{12}, y_{12}$. Знайти опуклу оболонку цієї множини точок.

Варіант 6

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for i = 1 to m
    if a[i] mod 2 = 0
        for j = 1 to m
```

```
b[i][j] = a[i];
```

б)
```
d = b * b - 4 * a * c
s_d = sqrt(d)
x1 = (-b - s_d) / (2 * a)
x2 = (-b + s_d) / (2 * a).
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = 3^n,$$
$$f_2(n) = \sqrt{n}.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$\log n = O(n);$$
$$n \log n = O(n^2).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $daaefeefad$ та $deecaaaefa$;

2) $dacdfedadd$ та $ddcdbffafc$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[5,6);\ [10,16);\ [14,18);\ [10,12);\ [13,16);\ [1,5);\ [9,11);\ [7,12);\ [7,8);\ [9,12)$;

2) $[9,13);\ [14,17);\ [1,3);\ [4,6);\ [4,7);\ [13,18);\ [7,11);\ [3,7);\ [5,8);\ [12,14)$.

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 9 6 2 0 5 0 8 7 4 5;

2) 4 5 1 9 2 6 4 6 5 4.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 16, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 17-ма вершинами та 20-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2; \ldots\ x_8, y_8$. Знайти опуклу оболонку цієї множини точок.

Варіант 7

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
a_pow = a
result = 1
while k > 0
   if k mod 2 = 1
        result = result * a_pow
   a_pow = a_pow * a_pow
   k = k div 2;
```

б)
```
for k = 1 to n
     for i = 1 to n
          for j = 1 to n
               if a[i, k] + a[k, j] < a[i, j]
                    a[i, j] = a[i, k] + a[k, j].
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = log^k n,$$
$$f_2(n) = k^n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^2 + 2n = O(n^2),$$
$$n^4 + n^2 = O(n^5).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $addaadfbcc$ та $cdabdfadcc$;

2) $fabcfbcdea$ та $dabfaccaad$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[12, 15);\ [1, 3);\ [14, 16);\ [4, 5);\ [10, 13);\ [7, 13);\ [8, 10);\ [5, 6);\ [7, 10);\ [5, 8);$

2) $[12, 15);\ [1, 3);\ [4, 6);\ [12, 13);\ [10, 12);\ [9, 11);\ [11, 17);\ [6, 12);\ [5, 8);\ [3, 5).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 5 9 1 7 1 7 4 8 2 4;

2) 5 5 8 1 7 3 5 3 3 1.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 20, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 17-ма вершинами та 24-ма ребрами, не більше 5-ти ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_{10}, y_{10}$. Знайти опуклу оболонку цієї множини точок.

Варіант 8

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
total = 0
    for i = 1 to m
      k = a[i]
      while k > 0
        if k mod 2 = 1
          total = total + 1
      k = k div 2;
```

б)
```
d = b * b - 4 * a * c
  s_d = sqrt(d)
  x1 = (-b - s_d) / (2 * a)
  x2 = (-b + s_d) / (2 * a).
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = 3^n;$$

$$f_2(n) = \sin n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$\log n = O(n);$$

$$e^{2n} = O(9^n).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $daaefeefad$ та $deecaaaefa$;

2) $aacaaeaeca$ та $adacaeefcb$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[5, 6)$ $[10, 16)$ $[14, 18)$ $[10, 12)$ $[13, 16)$ $[1, 5)$ $[9, 11)$ $[7, 12)$ $[7, 8)$ $[9, 12)$;

2) $[12, 15)$ $[1, 3)$ $[9, 14)$ $[14, 19)$ $[8, 11)$ $[1, 7)$ $[8, 9)$ $[9, 11)$ $[13, 19)$ $[3, 7)$.

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 9 6 2 0 5 0 8 7 4 5;

2) 0 5 9 5 4 4 0 1 4 1.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 24, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 10-ма вершинами та 18-ма ребрами, не більше 4-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_{12}, y_{12}$. Знайти опуклу оболонку цієї множини точок.

Варіант 9

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
i_from = 1
i_to = n
while i_from < i_to
  i_med = (i_from + i_to) div 2
```

```
if a[i_med] > goal
     i_to = i_med - 1
else if a[i_med] < goal
     i_from = i_med + 1
else
     break;
```

б)
```
d = b * b - 4 * a * c
s_d = sqrt(d)
x1 = (-b - s_d) / (2 * a)
x2 = (-b + s_d) / (2 * a).
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = log^k n;$$

$$f_2(n) = \sqrt{n}.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^2 + 2n = O(n^2);$$

$$n \log n = O(n).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $addaadfbcc$ та $cdabdfadcc$;

2) $dacdfedadd$ та $ddcdbffafc$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[12, 15)$ $[1, 3)$ $[14, 16)$ $[4, 5)$ $[10, 13)$ $[7, 13)$ $[8, 10)$ $[5, 6)$ $[7, 10)$ $[5, 8)$;

2) $[9, 13)$ $[14, 17)$ $[1, 3)$ $[4, 6)$ $[4, 7)$ $[13, 18)$ $[7, 11)$ $[3, 7)$ $[5, 8)$ $[12, 14)$.

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 5 9 1 7 1 7 4 8 2 4;

2) 4 5 1 9 2 6 4 6 5 4.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною,

що дорівнює 16, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 15-ма вершинами та 18-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \dots\ x_8, y_8$. Знайти опуклу оболонку цієї множини точок.

Варіант 10

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
d = b * b - 4 * a * c
s_d = sqrt(d)
x1 = (-b - s_d) / (2 * a)
x2 = (-b + s_d) / (2 * a);
```

б)
```
for j = 1 to n
    for i = 1 to j - 1
        if a[i] > a[i + 1]
            t := a[i]
            a[i] := a[i + 1]
            a[i + 1] := t.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = 3^n;$$
$$f_2(n) = tg\ n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$\log n = O(n);$$
$$n^3 + n \log^2 n \ = O(n^3).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $daaefeefad$ та $deecaaaefa$;

2) $abbcfebfbf$ та $bdbfcebddf$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) [5, 6); [10, 16); [14, 18); [10, 12); [13, 16); [1, 5); [9, 11); [7, 12); [7, 8); [9, 12);

2) $[1, 2);\ [12, 15);\ [2, 6);\ [10, 11);\ [12, 14);\ [7, 12);\ [13, 17);\ [7, 10);\ [2, 5);\ [11, 13)$.

Завдання 4. Написати програму для машини Тюрінга, яка додає два

числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 9 6 2 0 5 0 8 7 4 5;

2) 6 3 5 7 2 2 5 5 3 0.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 20, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 16-ма вершинами та 25-ма ребрами, не більше 5-ти ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_{10}, y_{10}$. Знайти опуклу оболонку цієї множини точок.

Варіант 11

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for j = 1 to n
    for i = 1 to n - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t;
```

б)
```
d = b * b - 4 * a * c
s_d = sqrt(d)
x1 = (-b - s_d) / (2 * a)
x2 = (-b + s_d) / (2 * a).
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = \log^k n;$$

$$f_2(n) = \sin n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$\log n = O(n);$$

$$e^n + n^{21} = O(e^n).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $daaefeefad$ та $deecaaaefa$;

2) $defebadaff$ та $badeffcedf$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[5,6);\ [10,16);\ [14,18);\ [10,12);\ [13,16);\ [1,5);\ [9,11);\ [7,12);\ [7,8);\ [9,12);$

2) $[11,16);\ [6,9);\ [1,3);\ [10,11);\ [4,5);\ [10,13);\ [5,7);\ [8,10);\ [14,15);\ [13,18).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 9 6 2 0 5 0 8 7 4 5;

2) 2 1 4 9 9 3 4 2 2 7.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 24, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 14-ма вершинами та 16-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_{12}, y_{12}$. Знайти опуклу оболонку цієї множини точок.

Варіант 12

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
d = b * b - 4 * a * c
```

```
s_d = sqrt(d)
x1 = (-b - s_d) / (2 * a)
x2 = (-b + s_d) / (2 * a);
```

б)
```
for i = 1 to n
    a[i] = i + i * i.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = 3^n,$$

$$f_2(n) = \sin^k n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^2 + 2n = O(n^2),$$

$$e^{2n} = O(9^n).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $addaadfbcc$ та $cdabdfadcc$;

2) $aacaaeaeca$ та $adacaeefcb$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[12, 15);\ [1, 3);\ [14, 16);\ [4, 5);\ [10, 13);\ [7, 13);\ [8, 10);\ [5, 6);\ [7, 10);\ [5, 8);$

2) $[12, 15);\ [1, 3);\ [9, 14);\ [14, 19);\ [8, 11);\ [1, 7);\ [8, 9);\ [9, 11);\ [13, 19);\ [3, 7).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 5 9 1 7 1 7 4 8 2 4;

2) 0 5 9 5 4 4 0 1 4 1.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 16, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 17-ма вершинами та 20-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2; \ldots\ x_8, y_8$. Знайти опуклу оболонку цієї множини точок.

Варіант 13

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for i = 1 to m
    for j = 1 to (n div 2)
        t = a[i, j]
        a[i, j] = a[i, n - j]
        a[i, n - j] = t;
```

б)
```
d = b * b - 4 * a * c
s_d = sqrt(d)
x1 = (-b - s_d) / (2 * a)
x2 = (-b + s_d) / (2 * a).
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = \log^k n,$$
$$f_2(n) = \operatorname{tg} n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$\log n = O(n),$$
$$0{,}9^n + n^2 = O(n^2).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $daaefeefad$ та $deecaaaefa$;

2) $fdcacaaffc$ та $cfedafafab$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[5,6);\ [10,16);\ [14,18);\ [10,12);\ [13,16);\ [1,5);\ [9,11);\ [7,12);\ [7,8);\ [9,12);$

2) $[5,9);\ [6,9);\ [9,14);\ [12,14);\ [5,6);\ [10,16);\ [6,10);\ [1,4);\ [13,14);\ [9,11).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 9 6 2 0 5 0 8 7 4 5;

2) 5 4 6 9 6 3 8 0 3 0.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 20, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 17-ма вершинами та 24-ма ребрами, не більше 5-ти ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_{10}, y_{10}$. Знайти опуклу оболонку цієї множини точок.

Варіант 14

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for i = 1 to m
    if a[i] mod 2 = 0
    for j = 1 to m
    b[i][j] = a[i];
```

б)
```
d = b * b - 4 * a * c
s_d = sqrt(d)
x1 = (-b - s_d) / (2 * a)
x2 = (-b + s_d) / (2 * a).
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = 3^n,$$

$$f_2(n) = e^n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^2 + 2n = O(n^2),$$

$$n^3 + nlog\,^2 n = O(n^3).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $addaadfbcc$ та $cdabdfadcc$;

2) $abbcfebfbf$ та $bdbfcebddf$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[12,15);\ [1,3);\ [14,16);\ [4,5);\ [10,13);\ [7,13);\ [8,10);\ [5,6);\ [7,10);\ [5,8);$

2) $[1,2);\ [12,15);\ [2,6);\ [10,11);\ [12,14);\ [7,12);\ [13,17);\ [7,10);\ [2,5);\ [11,13).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 5 9 1 7 1 7 4 8 2 4;

2) 6 3 5 7 2 2 5 5 3 0.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 24, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 10-ма вершинами та 18-ма ребрами, не більше 4-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2; \ldots\ x_{12}, y_{12}$. Знайти опуклу оболонку цієї множини точок.

Варіант 15

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
a_pow = a
result = 1
while k > 0
   if k mod 2 = 1 then
        result = result * a_pow
   a_pow = a_pow * a_pow
   k = k div 2;
```

б)
```
for j = 1 to n
    for i = 1 to n - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = \log^k n,$$
$$f_2(n) = \sin^k n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^{45} = O(n!),$$
$$n^7 = O(e^n).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $dfaedfcfba$ та $abbaedbfba$;

2) $fbeaedbdcb$ та $bedfbbdecb$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[1,3);\ [4,6);\ [7,12);\ [13,15);\ [13,16);\ [8,9);\ [11,15);\ [7,11);\ [11,12);\ [3,5)$;

2) $[14,17);\ [1,3);\ [10,11);\ [5,6);\ [6,9);\ [10,12);\ [13,18);\ [14,20);\ [7,11);\ [3,5)$.

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 2 9 1 9 7 2 0 5 7 8;

2) 8 0 5 4 6 4 4 4 3 7.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 16, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 15-ма вершинами та 18-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати по-

шук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2; \ldots\ x_8, y_8$. Знайти опуклу оболонку цієї множини точок.

Варіант 16

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
total = 0
   for i = 1 to m
          k = a[i]
     while k > 0
          if k mod 2 = 1
               total = total + 1
          k = k div 2;
```

б)
```
for i = 1 to n do
     a[i] = i + i * i.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = \log(n^n),$$
$$f_2(n) = k^n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^2 + 2n = O(n^2),$$
$$e^n + n^{21} = O(e^n).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $addaadfbcc$ та $cdabdfadcc$

2) $defebadaff$ та $badeffcedf$

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[12, 15);\ [1, 3);\ [14, 16);\ [4, 5);\ [10, 13);\ [7, 13);\ [8, 10);\ [5, 6);\ [7, 10);\ [5, 8);$

2) $[11, 16);\ [6, 9);\ [1, 3);\ [10, 11);\ [4, 5);\ [10, 13);\ [5, 7);\ [8, 10);\ [14, 15);\ [13, 18).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 5 9 1 7 1 7 4 8 2 4;

2) 2 1 4 9 9 3 4 2 2 7.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 20, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 16-ма вершинами та 25-ма ребрами, не більше 5-ти ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2; \ldots\ x_{10}, y_{10}$. Знайти опуклу оболонку цієї множини точок.

Варіант 17

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
i_from = 1
i_to = n
while i_from < i_to
   i_med = (i_from + i_to) div 2
   if a[i_med] > goal
       i_to = i_med - 1
   else if a[i_med] < goal
       i_from = i_med + 1
   else
       break;
```

б)
```
for j = 1 to n
    for i = 1 to j - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши,

чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = \log^k n,$$

$$f_2(n) = e^n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^{45} = O(n!),$$

$$n^4 + n^2 = O(n^5).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $dfaedfcfba$ та $abbaedbfba$;

2) $fabcfbcdea$ та $dabfaccaad$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[1,3);\ [4,6);\ [7,12);\ [13,15);\ [13,16);\ [8,9);\ [11,15);\ [7,11);\ [11,12);\ [3,5);$

2) $[12,15);\ [1,3);\ [4,6);\ [12,13);\ [10,12);\ [9,11);\ [11,17);\ [6,12);\ [5,8);\ [3,5).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 2 9 1 9 7 2 0 5 7 8;

2) 5 5 8 1 7 3 5 3 3 1.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 24, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 14-ма вершинами та 16-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати пошук в глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_{12}, y_{12}$. Знайти опуклу оболонку цієї множини точок.

Варіант 18

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
a_pow = a
result = 1
while k > 0
  if k mod 2 = 1
    result = result * a_pow
  a_pow = a_pow * a_pow
  k = k div 2;
```

б)
```
for j = 1 to n
    for i = 1 to n - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = \log(n^n),$$
$$f_2(n) = \sqrt{n}.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^2 + 2n = O(n^2),$$
$$0{,}9^n + n^2 = O(n^2).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $addaadfbcc$ та $cdabdfadcc$;

2) $fdcacaaffc$ та $cfedafafab$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[12, 15);\ [1, 3);\ [14, 16);\ [4, 5);\ [10, 13);\ [7, 13);\ [8, 10);\ [5, 6);\ [7, 10);\ [5, 8);$

2) $[5, 9);\ [6, 9);\ [9, 14);\ [12, 14);\ [5, 6);\ [10, 16);\ [6, 10);\ [1, 4);\ [13, 14);\ [9, 11).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 5 9 1 7 1 7 4 8 2 4;

2) 5 4 6 9 6 3 8 0 3 0.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 16, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 17-ма вершинами та 20-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \dots\ x_8, y_8$. Знайти опуклу оболонку цієї множини точок.

Варіант 19

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
total = 0
  for i = 1 to m
    k = a[i]
    while k > 0
        if k mod 2 = 1
            total = total + 1
        k = k div 2;
```

б)
```
for k = 1 to n
    for i = 1 to n
        for j = 1 to n
            if a[i, k] + a[k, j] < a[i, j]
                 a[i, j] = a[i, k] + a[k, j].
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = k^n,$$

$$f_2(n) = \sqrt{n}.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^{45} = O(n!),$$

$$n\log n = O(n^2).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $dfaedfcfba$ та $abbaedbfba$;

2) $dacdfedadd$ та $ddcdbffafc$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[1,3);\ [4,6);\ [7,12);\ [13,15);\ [13,16);\ [8,9);\ [11,15);\ [7,11);\ [11,12);\ [3,5);$

2) $[9,13);\ [14,17);\ [1,3);\ [4,6);\ [4,7);\ [13,18);\ [7,11);\ [3,7);\ [5,8);\ [12,14).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 2 9 1 9 7 2 0 5 7 8;

2) 4 5 1 9 2 6 4 6 5 4.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 20, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 17-ма вершинами та 24-ма ребрами, не більше 5-ти ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2; \ldots\ x_{10}, y_{10}$. Знайти опуклу оболонку цієї множини точок.

Варіант 20

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for j = 1 to n
    for i = 1 to n - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t;
```

б)
```
i_from = 1
i_to = n
while i_from < i_to
  i_med = (i_from + i_to) div 2
  if a[i_med] > goal
      i_to = i_med - 1
  else if a[i_med] < goal
      i_from = i_med + 1
  else
      break.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = \log(n^n),$$
$$f_2(n) = \sin n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^7 = O(e^n),$$
$$n^4 + n^2 = O(n^5).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $fbeaedbdcb$ та $bedfbbdecb$;

2) $fabcfbcdea$ та $dabfaccaad$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[14, 17);\ [1, 3);\ [10, 11);\ [5, 6);\ [6, 9);\ [10, 12);\ [13, 18);\ [14, 20);\ [7, 11);\ [3, 5);$

2) $[12, 15);\ [1, 3);\ [4, 6);\ [12, 13);\ [10, 12);\ [9, 11);\ [11, 17);\ [6, 12);\ [5, 8);\ [3, 5).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 8 0 5 4 6 4 4 4 3 7;

2) 5 5 8 1 7 3 5 3 3 1.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною,

що дорівнює 24, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 10-ма вершинами та 18-ма ребрами, не більше 4-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \dots\ x_{12}, y_{12}$. Знайти опуклу оболонку цієї множини точок.

Варіант 21

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for j = 1 to n
    for i = 1 to j - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t;
```

б)
```
for j = 1 to n
    for i = 1 to n - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = k^n,$$
$$f_2(n) = \sin n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n\log n = O(n^2),$$
$$n^3 + n\log^2 n = O(n^3).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $dacdfedadd$ та $ddcdbffafc$;

2) $abbcfebfbf$ та $bdbfcebddf$.

Завдання 3. Розв’язати задачу про заявки жадібним алгоритмом.

1) $[9, 13);\ [14, 17);\ [1, 3);\ [4, 6);\ [4, 7);\ [13, 18);\ [7, 11);\ [3, 7);\ [5, 8);\ [12, 14);$

2) $[1,2);\ [12,15);\ [2,6);\ [10,11);\ [12,14);\ [7,12);\ [13,17);\ [7,10);\ [2,5);\ [11,13).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 4 5 1 9 2 6 4 6 5 4;

2) 6 3 5 7 2 2 5 5 3 0.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 16, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 15-ма вершинами та 18-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2; \ldots\ x_8, y_8$. Знайти опуклу оболонку цієї множини точок.

Варіант 22

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for j = 1 to n
    for i = 1 to j - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t;
```

б)
```
for i = 1 to n
    a[i] = i + i * i.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = \log(n^n),$$

$$f_2(n) = \operatorname{tg} n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^{45} = O(n!),$$

$$e^{2n} = O(9^n).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $dfaedfcfba$ та $abbaedbfba$;

2) $aacaaeaeca$ та $adacaeefcb$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[1,3);\ [4,6);\ [7,12);\ [13,15);\ [13,16);\ [8,9);\ [11,15);\ [7,11);\ [11,12);\ [3,5);$

2) $[12,15);\ [1,3);\ [9,14);\ [14,19);\ [8,11);\ [1,7);\ [8,9);\ [9,11);\ [13,19);\ [3,7).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 2 9 1 9 7 2 0 5 7 8;

2) 0 5 9 5 4 4 0 1 4 1.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 20, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 16-ма вершинами та 25-ма ребрами, не більше 5-ти ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_{10}, y_{10}$. Знайти опуклу оболонку цієї множини точок.

Варіант 23

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for j = 1 to n
    for i = 1 to j - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t;
```

б)
```
for i = 1 to m
    for j = 1 to (n div 2)
        t = a[i, j]
        a[i, j] = a[i, n - j]
        a[i, n - j] = t.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = \sqrt{n},$$
$$f_2(n) = \sin n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^7 = O(e^n),$$
$$n\log n = O(n^2).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $fbeaedbdcb$ та $bedfbbdecb$;

2) $dacdfedadd$ та $ddcdbffafc$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[14, 17)$; $[1, 3)$; $[10, 11)$; $[5, 6)$; $[6, 9)$; $[10, 12)$; $[13, 18)$; $[14, 20)$; $[7, 11)$; $[3, 5)$;

2) $[9, 13)$; $[14, 17)$; $[1, 3)$; $[4, 6)$; $[4, 7)$; $[13, 18)$; $[7, 11)$; $[3, 7)$; $[5, 8)$; $[12, 14)$.

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 8 0 5 4 6 4 4 4 3 7;

2) 4 5 1 9 2 6 4 6 5 4.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 24, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 14-ма вершинами та 16-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \dots\ x_{12}, y_{12}$. Знайти опуклу оболонку цієї множини точок.

Варіант 24

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for j = 1 to n
    for i = 1 to j - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t;
```

б)
```
for i = 1 to m
    if a[i] mod 2 = 0
        for j = 1 to m
            b[i][j] = a[i].
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = \log(n^n),$$

$$f_2(n) = \sin^k n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^{45} = O(n!),$$

$$n^3 + n\log n = O(n^3).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $dfaedfcfba$ та $abbaedbfba$;

2) $abbcfebfbf$ та $bdbfcebddf$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[1,3);\ [4,6);\ [7,12);\ [13,15);\ [13,16);\ [8,9);\ [11,15);\ [7,11);\ [11,12);\ [3,5);$

2) $[1,2);\ [12,15);\ [2,6);\ [10,11);\ [12,14);\ [7,12);\ [13,17);\ [7,10);\ [2,5);\ [11,13).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 2 9 1 9 7 2 0 5 7 8;

2) 6 3 5 7 2 2 5 5 3 0.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 16, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 17-ма вершинами та 20-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2; \ldots\ x_8, y_8$. Знайти опуклу оболонку цієї множини точок.

Варіант 25

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for j = 1 to n
    for i = 1 to j - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t;
```

б)
```
a_pow = a
result = 1
while k > 0
  if k mod 2 = 1
```

```
        result = result * a_pow
    a_pow = a_pow * a_pow
    k = k div 2.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = k^n,$$

$$f_2(n) = \operatorname{tg} n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^4 + n^2 = O(n^5),$$

$$n \log n = O(n^2).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $fabcfbcdea$ та $dabfaccaad$;

2) $dacdfedadd$ та $ddcdbffafc$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[12, 15);\ [1, 3);\ [4, 6);\ [12, 13);\ [10, 12);\ [9, 11);\ [11, 17);\ [6, 12);\ [5, 8);\ [3, 5);$

2) $[9, 13);\ [14, 17);\ [1, 3);\ [4, 6);\ [4, 7);\ [13, 18);\ [7, 11);\ [3, 7);\ [5, 8);\ [12, 14).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 5 5 8 1 7 3 5 3 3 1;

2) 4 5 1 9 2 6 4 6 5 4.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 20, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 17-ма вершинами та 24-ма ребрами, не більше 5-ти ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_{10}, y_{10}$. Знайти опуклу оболонку цієї множини точок.

Варіант 26

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for j = 1 to n
    for i = 1 to j - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t;
```

б)
```
total = 0;
  for i = 1 to m
       k = a[i];
       while k > 0
           if k mod 2 = 1
               total = total + 1
           k = k div 2.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = \log(n^n),$$
$$f_2(n) = e^n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^{45} = O(n!),$$
$$e^n + n^{21} = O(e^n).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $dfaedfcfba$ та $abbaedbfba$;

2) $defebadaff$ та $badeffcedf$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[1,3);\ [4,6);\ [7,12);\ [13,15);\ [13,16);\ [8,9);\ [11,15);\ [7,11);\ [11,12);\ [3,5);$

2) $[11,16);\ [6,9);\ [1,3);\ [10,11);\ [4,5);\ [10,13);\ [5,7);\ [8,10);\ [14,15);\ [13,18).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 2 9 1 9 7 2 0 5 7 8;

2) 2 1 4 9 9 3 4 2 2 7.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 24, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 10-ма вершинами та 18-ма ребрами, не більше 4-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_{12}, y_{12}$. Знайти опуклу оболонку цієї множини точок.

Варіант 27

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for j = 1 to n
    for i = 1 to j - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t;
```

б)
```
for j = 1 to n
    for i = 1 to j - 1
        if a[i] > a[i + 1]
            t = a[i]
            a[i] = a[i + 1]
            a[i + 1] = t.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = \sqrt{n},$$

$$f_2(n) = \operatorname{tg} n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^7 = O(e^n),$$
$$e^{2n} = O(9^n).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $fbeaedbdcb$ та $bedfbbdecb$;

2) $aacaaeaeca$ та $adacaeefcb$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[14,17);\ [1,3);\ [10,11);\ [5,6);\ [6,9);\ [10,12);\ [13,18);\ [14,20);\ [7,11);\ [3,5);$

2) $[12,15);\ [1,3);\ [9,14);\ [14,19);\ [8,11);\ [1,7);\ [8,9);\ [9,11);\ [13,19);\ [3,7).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 8 0 5 4 6 4 4 4 3 7;

2) 0 5 9 5 4 4 0 1 4 1.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 16, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 15-ма вершинами та 18-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_8, y_8$. Знайти опуклу оболонку цієї множини точок.

Варіант 28

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for i = 1 to m
    if a[i] mod 2 = 0
        for j = 1 to m
            b[i][j] = a[i];
```

б)
```
for i = 1 to m do
    for j = 1 to (n div 2)
        t = a[i, j]
        a[i, j] = a[i, n - j]
        a[i, n - j] = t.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = k^n,$$

$$f_2(n) = \sin^k n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^{45} = O(n!),$$

$$0.9^n + n^2 = O(n^2).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $dfaedfcfba$ та $abbaedbfba$;

2) $aacaaeaeca$ та $adacaeefcb$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[1,3);\ [4,6);\ [7,12);\ [13,15);\ [13,16);\ [8,9);\ [11,15);\ [7,11);\ [11,12);\ [3,5);$

2) $[5,9);\ [6,9);\ [9,14);\ [12,14);\ [5,6);\ [10,16);\ [6,10);\ [1,4);\ [13,14);\ [9,11).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 2 9 1 9 7 2 0 5 7 8;

2) 5 4 6 9 6 3 8 0 3 0.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 20, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 16-ма вершинами та 25-ма ребрами, не більше 5-ти ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2; \ldots\ x_{10}, y_{10}$. Знайти опуклу оболонку цієї множини точок.

Варіант 29

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
a_pow = a
result = 1
while k > 0
  if k mod 2 = 1
      result = result * a_pow
  a_pow = a_pow * a_pow
  k = k div 2;
```

б)
```
total = 0
for i = 1 to m
   k = a[i]
   while k > 0
       if k mod 2 = 1
           total = total + 1
       k = k div 2.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши, чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = \sin n,$$
$$f_2(n) = \operatorname{tg} n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^4 + n^2 = O(n^5),$$
$$e^{2n} = O(9^n).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $fabcfbcdea$ та $dabfaccaad$;

2) $aacaaeaeca$ та $adacaeefcb$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[12, 15);\ [1, 3);\ [4, 6);\ [12, 13);\ [10, 12);\ [9, 11);\ [11, 17);\ [6, 12);\ [5, 8);\ [3, 5);$

2) $[12, 15);\ [1, 3);\ [9, 14);\ [14, 19);\ [8, 11);\ [1, 7);\ [8, 9);\ [9, 11);\ [13, 19);\ [3, 7).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два

числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 5 5 8 1 7 3 5 3 3 1;

2) 0 5 9 5 4 4 0 1 4 1.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 24, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 14-ма вершинами та 16-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2; \ldots\ x_{12}, y_{12}$. Знайти опуклу оболонку цієї множини точок.

Варіант 30

Завдання 1. Визначити складність алгоритмів, записаних у вигляді псевдокоду:

а)
```
for k = 1 to n
    for i = 1 to n
        for j = 1 to n
            if a[i, k] + a[k, j] < a[i, j]
                a[i, j] = a[i, k] + a[k, j];
```

б)
```
total = 0
  for i = 1 to m
       k = a[i]
       while k > 0
           if k mod 2 = 1
               total = total + 1
           k = k div 2.
```

Порівняти попарно швидкості зростання заданих функцій, визначивши,

чи можна дану пару записати як $f(n) = O(g(n))$, при цьому $n, k > 1$:

$$f_1(n) = k^n,$$
$$f_2(n) = n^n.$$

Довести дані твердження, користуючись визначенням O-позначення:

$$n^7 = O(e^n),$$
$$n^3 + n\log^2 n = O(n^3).$$

Завдання 2. Визначити найбільшу спільну підпослідовність:

1) $fbeaedbdcb$ та $bedfbbdecb$;

2) $abbcfebfbf$ та $bdbfcebddf$.

Завдання 3. Розв'язати задачу про заявки жадібним алгоритмом.

1) $[14, 17);\ [1, 3);\ [10, 11);\ [5, 6);\ [6, 9);\ [10, 12);\ [13, 18);\ [14, 20);\ [7, 11);\ [3, 5);$

2) $[1, 2);\ [12, 15);\ [2, 6);\ [10, 11);\ [12, 14);\ [7, 12);\ [13, 17);\ [7, 10);\ [2, 5);\ [11, 13).$

Завдання 4. Написати програму для машини Тюрінга, яка додає два числа. Перевірити її роботу на числах a, b. Для визначення чисел необхідно додати 10 до номера варіанта та взяти першу цифру суми як a, другу – як b.

Завдання 5. Виконати сортування послідовностей:

1) 8 0 5 4 6 4 4 4 3 7;

2) 6 3 5 7 2 2 5 5 3 0.

Методи сортування:

1) бульбашкою;

2) купою;

3) злиттям;

4) швидким сортуванням.

Завдання 6. Згенерувати дві псевдовипадкові послідовності довжиною, що дорівнює 16, двома генераторами (початкові параметри вибрати довільно).

Завдання 7. Побудувати орієнтований граф без циклів з 17-ма вершинами та 20-ма ребрами, не більше 3-х ребер при кожній вершині. Виконати пошук у глибину, ширину та провести топологічне сортування.

Завдання 8. Взяти будь-яку послідовність із завдання 6 та використати її числа як координати $x_1, y_1;\ x_2, y_2;\ \ldots\ x_8, y_8$. Знайти опуклу оболонку цієї множини точок.

2. ЛАБОРАТОРНИЙ ПРАКТИКУМ

2.1. Загальні рекомендації до виконання лабораторного практикуму

Цілі виконання лабораторних робіт з курсу «Алгоритми і структури даних» такі:

– формування практичних навичок організації та використання при вирішенні завдань динамічних структур даних;

– вивчення найбільш поширених алгоритмів розв’язання задач з використанням складних структур даних.

План проведення лабораторних занять подано в табл. 2.1.

Таблиця 2.1 – План проведення лабораторних занять

№	Тема лабораторної роботи	Кількість годин
1	Базові структури даних. Черги, стеки і списки	2
2	Базові структури даних. Хеш-функції	2
3	Базові структури даних. Червоно-чорні дерева	4
4	Алгоритми сортування	2
5	Генератори псевдовипадкових чисел	2
6	Геометричні алгоритми	2
7	Фундаментальні алгоритми на графах і деревах	2

На етапі підготовки до лабораторної роботи студенти повинні, використовуючи матеріали лекцій і рекомендовану літературу, поглибити свої знання за темою лабораторної роботи.

Викладач перед проведенням заняття проводить контрольне опитування студентів і визначає ступінь їх готовності до роботи. Потім викладач видає студентам індивідуальний варіант завдання на лабораторну роботу.

Завдання можна виконувати мовою програмування на вибір: C; C++; Java; C#; Python; Ruby.

Програму можна демонструвати у будь-якій операційній системі. Рекомендовано розробляти застосунок так, щоб його можна було зібрати та виконувати на широкому класі систем (цього елементарно досягнути на перелічених мовах програмування). Рекомендовано використовувати будь-який дис-

трибутив GNU/Linux або FreeBSD.

Вибір середовища розробки може бути продиктований звичками студента. Доцільно використати декілька середовищ для різних робіт.

Перелічимо кілька важливих вільних або безкоштовних редакторів та середовищ розробки: Vim; Emacs; Eclipse; NetBeans; IntelliJ IDEA; KDevelop; MonoDevelop.

Звіт до лабораторної роботи формується в такому порядку:

1. Титульний аркуш.

Титульний лист оформляється відповідно до зразка, прийнятого на кафедрі.

2. Завдання на лабораторну роботу.

Наводиться описання завдання відповідно до виданого варіанта.

3. Мета роботи.

Мета роботи показує, для чого виконується робота, наприклад, для отримання чи закріплення якихось навичок, вивчення яких-небудь закономірностей і т.п.

4. Основні теоретичні положення.

Наводиться короткий опис основних теоретичних положень, опис структур даних, методів і алгоритмів, необхідних для виконання роботи.

5. Описання розробленого застосунку.

Описується структура застосунку (перераховуються файли, в яких містяться частини коду застосунку, та розв'язувані ними задачі, а також наводиться схема взаємодії даних файлів). Наводиться вихідний код (лістинг) застосунку з коментарями.

6. Результати.

Наводяться результати розрахунків, графіки отриманих залежностей, інші необхідні дані.

7. Висновки.

Оцінюється ступінь відповідності отриманих результатів розрахунку і експериментів до теоретичних даних. Дається пояснення отриманих у ході виконання роботи залежностей і результатів.

8. Список використаних інформаційних джерел.

Звіт до лабораторної роботи виконується на аркушах білого паперу формату А4 в друкованому чи рукописному вигляді.

При оформленні звіту використовується наскрізна нумерація сторінок,

враховуючи титульний аркуш як першу сторінку. Номер сторінки на титульному аркуші не ставиться. Номери сторінок ставляться зверху справа.

Лабораторна робота 1

Тема: Базові структури даних. Черги, стеки і списки.

Мета роботи: ознайомитися з базовими структурами даних (список, черга, стек) та отримати навички програмування алгоритмів, що їх обробляють.

Короткі теоретичні відомості

Стеки і черги – це динамічні множини, в яких елемент, що видаляється з множини операцією $Delete$, що не задається довільно, а визначається структурою множини [1].

Стек ($stack$) – динамічна структура даних, в якій всі включення, виключення і доступ виконуються лише з одного кінця за принципом LIFO (від англ. *«Last In – First Out»* – останнім прийшов, першим пішов). Зі стека можна видалити тільки той елемент, який був у нього доданий останнім.

Черга ($queue$) – динамічна структура даних, в якій всі включення виконуються на одному кінці, а всі виключення (і доступ) – на іншому (за принципом FIFO (від англ. *«First In – First Out»* – першим прийшов, першим пішов). З черги, можна видалити тільки той елемент, який знаходився в черзі довше від усіх.

Одним із зручних способів реалізації динамічних множин є списки.

У зв'язаному списку (або просто списку; *linked list*) елементи лінійно впорядковані, але порядок визначається не номерами, як у масиві, а вказівниками, що входять до складу елементів списку. Кожний елемент списку подається схематично ланкою в ланцюзі, який складається з інформаційного поля $Info$, що містить власне елемент списку, і поля $Next$, куди записується посилання на наступний елемент.

Лінійний однонаправлений список – це структура даних, яка складається з елементів одного типу, послідовно пов'язаних між собою за допомогою вказівників. Кожен елемент списку має вказівник на наступний елемент. Останній елемент списку вказує на $NULL$. Елемент, на який немає вказівника, є першим (головним) елементом списку. Тут посилання в кожному вузлі вказує на наступний вузол у списку.

Елемент двобічно зв'язаного списку ($doubly\ linked\ list$) – це запис, що містить три поля: Key (ключ) і два вказівника $Next$ (наступний) і $Prev$ (попередній). Крім цього, елементи списку можуть містити додаткові дані.

У впорядкованому ($sorted$) списку елементи розташовані в порядку зростання ключів, таким чином, у голови списку ключ найменший, а у хвоста списку – найбільший, на відміну від неврегульованого ($unsorted$) списку.

У кільцевому списку (circular list) поле $Prev$ голови списку вказує на хвіст списку, а поле $Next$ хвоста списку вказує на голову списку.

Для виконання лабораторної роботи необхідно вивчити лекційний матеріал, а для детальнішого вивчення даної теми рекомендується ознайомитися з такими главами в книгах:

1) Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн «Алгоритмы. Построение и анализ», глава 10 «Элементарные структуры данных» (підрозділи 10.1–10.3) [1];

2) А. Ахо, Д. Хопкрофт, Д. Ульман «Структуры данных и алгоритмы», глава 2 «Основные абстрактные типы данных» (підрозділи 2.1–2.4) [2];

3) Д. Кнут «Искусство программирования, том 1. Основные алгоритмы, 3-е издание», глава 2, підрозділ 2.2 «Линейные списки» [4].

Запитання для самоконтролю

1. За яким принципом працює стек?
2. Як позначається операція додавання елемента в стек?
3. Як позначається операція видалення верхнього елемента зі стека?
4. За яким принципом працює черга?
5. Які види списків ви знаєте?
6. Що являє собою двосторонній зв'язаний список?

Завдання на лабораторну роботу

Розробити програму, яка читає з клавіатури послідовність N цілих чисел ($1 < N < 256$), жодне з яких не повторюється, зберігає їх до структури даних (згідно з завданням) та видає на екран такі характеристики:

- сума елементів послідовності;

- середнє арифметичне елементів послідовності;
- три мінімальні та максимальні елементи;
- елемент послідовності з номером $[N/2]$.

Усі перелічені характеристики потрібно визначити із заповненої структури даних. Дозволено використовувати лише ті операції, що притаманні заданій структурі. Наприклад, заборонено отримувати доступ до елемента із довільною позицією у черзі, яку реалізовано на базі масиву.

Використовувати готові реалізації структур даних (наприклад, STL) заборонено.

Указати складність алгоритмів для пошуку кожної характеристики.

Формат вхідних даних. Перший рядок – число N, далі N цілих чисел, розділених пробілами.

Приклад

Вхід.

```
9
1 9 2 8 3 7 4 6 5
```

Вихід.

```
45
5
1 2 3
9 8 7
8
```

Варіанти завдань

Використати такі структури даних:

1. Черга.
2. Стек.
3. Однобічно зв’язаний список.
4. Двобічно зв’язаний список.
5. Кільцевий список.
6. Масив із довільним доступом.

Лабораторна робота 2

Тема: Базові структури даних. Хеш-таблиці.

Мета роботи: ознайомитися з хеш-функціями та хеш-таблицями і отримати навички програмування алгоритмів, що їх обробляють.

Короткі теоретичні відомості

Часто виникає необхідність використання динамічних множин, що підтримують тільки «словникові операції» додавання, пошуку і видалення елемента. У цьому випадку часто застосовують так зване хешування; відповідна структура даних називається «хеш-таблиця». У гіршому випадку пошук в хеш-таблиці може займати стільки ж часу, скільки пошук у списку ($O(n)$), але на практиці хешування є досить ефективним.

У той час, як при прямій адресації елементу з ключем k відводиться позиція номер k, при хешуванні цей елемент записується в позицію номер $h(k)$ в хеш-таблиці (*hashtable*) $T[0..m-1]$, де $h: U \to 0,1,...,m-1$ – деяка функція, яка називається хеш-функцією (*hash function*).

Гарна хеш-функція повинна (наближено) задовольняти припущенням рівномірного хешування: для чергового ключа всі хеш-значення повинні бути рівноймовірними.

Тривіальне хешування полягає у використанні хеш-функції, що повертає власний аргумент.

Побудова хеш-функції методом ділення із залишком (division method) полягає в тому, що ключу k ставиться у відповідність до залишку від ділення k на m:

$$h(k) = k \bmod m,$$

де m – число можливих хеш-значень.

Гарні результати зазвичай виходять, якщо вибрати за m просте число, що є далеким від ступенів двійки.

Побудова хеш-функції методом множення (*multiplication method*) полягає в такому.

Нехай кількість хеш-значень дорівнює m. Зафіксуємо константу A в інтервалі (0,1) і покладемо $h(k) = [m(kA \bmod 1]$, де $kA \bmod 1$ – дрібна частина kA.

Перевага методу множення в тому, що якість хеш-функції мало залежить від вибору m. Звичайно за m вибирають ступінь двійки, оскільки в більшості комп'ютерів множення на таке m реалізується як зсув слова.

Хешування Пірсона (*Pearson hashing*) – алгоритм, запропонований Пітером Пірсоном для процесорів з 8-бітними регістрами, завданням якого є

швидке обчислення хеш-коду для рядка довільної довжини. На вхід функція отримує слово W, що складається з n символів, кожен розміром 1 байт, і повертає значення в діапазоні від 0 до 255. Значення хеш-коду залежить від кожного символу вхідного слова.

Алгоритм можна описати таким кодом мовою Python, який отримує на вхід рядок *w* і використовує таблицю перестановок T:

```
def hash_pearson(w)
h = 0
  for c in w
    index = h ^ ord(c)
    h = T[index]
  return h
```

Для виконання лабораторної роботи необхідно вивчити лекційний матеріал, а для повнішого вивчення даної теми рекомендується ознайомитися з такими розділами в книгах:

1) Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн «Алгоритмы: построение и анализ», глава 11 «Хеширование и хеш-таблицы» [1];

2) А. Ахо, Д. Хопкрофт, Д. Ульман «Структуры данных и алгоритмы», глава 4 «Основные операторы множеств», підрозділ 4.7 «Структуры данных, основанные на хеш-таблицах» [2];

3) Д. Кнут «Искусство программирования», том 3, «Сортировка и поиск», глава 6, підрозділ 6.4 «Хеширование» [5];

4) Седжвик Р. «Фундаментальные алгоритмы на C++. Анализ. Структуры данных. Сортировка. Поиск», глава 14 «Хеширование» [6];

5) Вирт, Никлаус «Алгоритмы и структуры данных. Новая версия для Оберона+CD», глава 5 «Хеширование» [7].

Запитання для самоконтролю

1. Дайте визначення хеш-функції.
2. Як виглядає хеш-таблиця?
3. Що таке колізія?
4. Які способи розв’язання колізій ви знаєте?
5. У чому полягає суть технології зчеплення елементів?
6. Які умови повинна задовольняти гарна хеш-функція?
7. Які методи побудови хеш-функції ви знаєте?

Завдання на лабораторну роботу

Розробити програму, яка читає з клавіатури словник пар ключ-значення, послідовність ключів та відшукує ключі в словнику. Словник треба зберегти в хеш-таблиці та видати на екран значення, що відповідають переліченим ключам.

Формат вхідних даних. Перший рядок – пара цілих чисел $N, M\,(1 < N, M < 256)$; далі N рядків з парами ключ-значення, що розділені пробілом, та M рядків, на кожному з яких розташований ключ, який треба відшукати. Ключ – ціле або дійсне число, або рядок залежно від варіанта завдання; значення – рядок. Усі рядки до 100 символів.

Приклад.

Вхід:

```
3 2
England London
France Paris
Germany Berlin
France
England
```

Вихід:

```
Paris
London
```

Використовувати готові реалізації структур даних (наприклад, STL) заборонено, але можна використати реалізацію рядків (наприклад, std::string у C++).

Варіанти завдань

1. Ключ – ціле число; тривіальне хешування.
2. Ключ – ціле число; хешування за допомогою ділення.
3. Ключ – ціле число; мультиплікативне хешування.
4. Ключ – дійсне число; мультиплікативне хешування.
5. Ключ – рядок; хешування за остачею суми символів.
6. Ключ – рядок; хешування Пірсона.

Лабораторна робота 3

Тема: Базові структури даних. Червоно-чорні дерева.

Мета роботи: ознайомитися з червоно-чорними деревами та отримати навички програмування алгоритмів, що їх обробляють.

Короткі теоретичні відомості

Двійкові дерева традиційно використовуються для зберігання динамічних даних. Якщо дерево збалансоване, то виконувати пошук, вставку і видалення можна набагато швидше, ніж у зв'язаному списку. Якщо ж дерево велике і незбалансоване, то швидкість його обробки може бути не набагато більшою, ніж у зв'язаному списку.

Червоно-чорні дерева – один з типів збалансованих дерев пошуку, в яких передбачені операції балансування гарантують, що висота дерева не перевищить $O(lg\ n)$.

Червоно-чорне дерево (*red-black tree*) – це двійкове дерево пошуку, вершини якого розділені на червоні (*red*) і чорні (*black*). При цьому повинні виконуватися певні вимоги, які гарантують, що глибини будь-яких двох листків відрізняються не більше ніж у два рази, тому дерево можна назвати збалансованим (*balanced*).

Двійкове дерево пошуку називається червоно-чорним деревом, якщо воно має такі властивості (*RB-властивості, red-black properties*).

1. Кожна вершина – або червона, або чорна.

2. Кожен листок (*nil*) – чорний.

3. Якщо вершина червона, обидві її дитини чорні.

4. Всі шляхи, що йдуть вниз від кореня до листя, містять однакову кількість чорних вершин [1].

Кожна вершина червоно-чорного дерева має поля $color$ (колір), key (ключ), $left$ (ліва дитина), $right$ (права дитина) і p (батько). Якщо у вершини відсутня дитина або батько, відповідне поле містить nil. Для зручності вважають, що значення nil, які зберігаються в полях $left$ і $right$, є посиланнями на додаткові (фіктивні) листя дерева.

Операції $Tree - Insert$ і $Tree - Delete$ виконуються на червоно-чорному дереві за час $O(lg\ n)$, але вони змінюють дерево, і результат може не мати RB-властивості. Щоб відновити ці властивості, треба перефарбувати деякі вершини і змінити структуру дерева.

Зміни в структурі вказівників виконуються за допомогою поворотів (*rotations*), які являються локальними операціями в дереві пошуку, що зберігають властивість бінарного дерева пошуку. Детально процес повороту та модифікації червоно-чорного дерева описано в [1].

Для виконання лабораторної роботи необхідно вивчити лекційний матеріал, а для більш повного вивчення даної теми рекомендується ознайомитися з такими розділами в книгах:

1) Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн «Алгоритмы: построение и анализ», глава 13 «Красно-черные деревья» [1];

2) Седжвик Р. «Фундаментальные алгоритмы на C++. Анализ. Структуры данных. Сортировка. Поиск», підрозділ 13.4 «Красно-черные деревья, или RB-деревья» [6].

Запитання для самоконтролю

1. Що таке червоно-чорні дерева?
2. Перечисліть властивості червоно-чорних дерев.
3. Опишіть операцію обертання на двійковому дереві пошуку.
4. Як відбувається додавання вершини до червоно-чорного дерева?

Завдання на лабораторну роботу

Розробити програму, яка читає з клавіатури числа $N, M\ (1 < N, M < 256)$; послідовність N ключів (цілих, дійсних чисел або рядків (до 255 символів) залежно від варіанта завдання); послідовність M ключів. Програма зберігає першу послідовність до червоно-чорного дерева.

Кожного разу, коли до дерева додається новий елемент, потрібно вивести статистику (згідно з варіантом завдання).

1. Мінімальний елемент та його колір.
2. Максимальний елемент та його колір.

Після побудови дерева для кожного елемента x з другої послідовності потрібно вивести результати таких операцій над деревом (згідно з варіантом завдання).

1. Колір елемента x.
2. $Successor(x)$ та його колір.
3. $Predecessor(x)$ та його колір.

Використовувати готові реалізації червоно-чорного дерева (наприклад, STL) заборонено, але можна використати реалізацію рядків (наприклад, std::string у C++).

Варіанти завдань

1. 1.1.
2. 1.2.
3. 1.3.
4. 2.1.
5. 2.2.
6. 2.3.

Лабораторна робота 4

Тема: Алгоритми сортування.

Мета роботи: ознайомитися з алгоритмами сортування та отримати навички їх програмування.

Короткі теоретичні відомості про алгоритми, що використовуються в лабораторній роботі, наведені в підрозділі **1.3. Алгоритми сортування** даного посібника.

Рекомендується також для детальнішого вивчення цієї теми ознайомитися з відповідними розділами у рекомендованій літературі [1, 2, 7, 8, 11].

Запитання для самоконтролю

1. Який принцип використовується в сортуванні злиттям?
2. Які алгоритми сортування працюють за час $O(n^2)$?
3. Які алгоритми сортування на практиці мають математичне очікування часу роботи $O(n \log n)$?
4. Для яких послідовностей чисел застосовується сортування підрахунком?

Завдання на лабораторну роботу

Розробити програму, яка читає з клавіатури числа $N, M\,(1 < N, M < 256)$; послідовність N цілих чисел; послідовність M цілих чисел. Програма зберігає першу послідовність до масиву та виконує сортування. Потім програма ви-

водить відсортовану послідовність на екран та виконує бінарний пошук кожного елемента другої послідовності x: для кожного x повідомити, чи є він у першій послідовності, якщо є, то на якому місці.

Приклад

Вхід.

```
10 3
1 9 2 8 3 7 4 6 5
8 3 20
```

Вихід.

```
1 2 3 4 5 6 7 8 9
8 8
3 3
20 Not found
```

Варіанти завдань

1. Сортування бульбашкою.
2. Сортування включенням.
3. Швидке сортування.
4. Сортування злиттям.
5. Сортування купою.
6. Сортування підрахунком.

Лабораторна робота 5

Тема: Генератори псевдовипадкових чисел.

Мета роботи: ознайомитися з генераторами випадкових чисел та методами перевірки випадковості.

Короткі теоретичні відомості про алгоритми і тести, що використовуються в лабораторній роботі, наведені в підрозділі **1.6. Генератори псевдовипадкових чисел** даного посібника.

Рекомендується також для повнішого вивчення цієї теми ознайомитися з відповідними розділами у рекомендованій літературі [11, 16].

Значення функцій $erfc$ і Q рекомендується обчислювати, використовуючи готові бібліотеки.

1) C++: $erfc$ доступна у файлі $cmath$ (функції $erfcf()$ і $erfcl()$), Q – у файлі boost/math/special_functions/gamma.hpp (функція gamma_q);

2) Java: $erfc$ реалізована в методі $Erf.erfc()$, Q – в методі Gamma.regularizedGammaQ (пакет org.apache.commons.math3.special);

3) Python: $erfc$ доступна як math.erfc(), Q – як scipy.special.gammainc як mpmath.gammainc.

Запитання для самоконтролю

1. Викладіть суть частотного побітового тесту.

2. У чому полягає суть тесту на найдовшу послідовність одиниць у блоці?

3. Які значення лагів рекомендується вибирати в датчику Фібоначчі із запізненнями?

4. З яким періодом періодична послідовність чисел побудована за допомогою лінійного конгруентного методу?

Завдання на лабораторну роботу

1. Визначити послідовність з перших 30 чисел, використовуючи лінійний конгруентний метод для різних параметрів (a, c) (при $m = 65536$):

1) (53879, 39, 27769);

2) (30578, 38, 30348);

3) (62774, 12, 63830);

4) (42679, 57, 25325);

5) (14626, 94, 11641);

6) (24095, 91, 64872);

7) (57278, 54, 46575);

8) (14144, 60, 58014).

2. Обчислити послідовність з 30 чисел, що генерується методом Фібоначчі з затримкою, використовуючи такі вихідні дані – пару (a, b):

1) (17, 5);

2) (17, 13);

3) (17, 7);

4) (17, 11);

5) (23, 5);

6) (23, 13);

7) (23, 7);

8) (23, 11).

Початкові значення згенерувати лінійним конгруентним методом відповідно до даних завдання 1.

3. Перевірити на випадковість послідовності, отримані в завданнях 1 і 2, використовуючи всі наступні тести:

а) частотний побітовий тест;

б) частотний блочний тест (довжина блока $M = 3$);

в) тест на послідовність однакових бітів;

г) тест на найдовшу послідовність одиниць в блоці.

Лабораторна робота 6

Тема: Геометричні алгоритми.

Мета роботи: ознайомитися з основними геометричними алгоритмами та отримати навички їх програмування.

Короткі теоретичні відомості про алгоритми, що використовуються в лабораторній роботі, наведені в підрозділі **1.8. Геометричні алгоритми** даного посібника.

Для виконання лабораторної роботи необхідно вивчити лекційний матеріал, а для детальнішого вивчення даної теми рекомендується ознайомитися з главою 33 «Вычислительная геометрия» книги Т. Кормена, Ч. Лейзерсона, Р. Ривеста, К. Штайна «Алгоритмы: построение и анализ» [1].

Запитання для самоконтролю

1. Як обчислити векторний добуток двох векторів через визначник матриці?

2. Як визначити напрям повороту двох послідовних відрізків за векторним добутком?

3. Охарактеризуйте два основних етапи перевірки перетину відрізків.

4. Чи може векторний добуток двох векторів дорівнювати нулю? Якщо так, то як можуть бути розташовані ці вектори?

Завдання на лабораторну роботу

Розробити програму, яка читає з клавіатури число $N\left(1 < N < 256\right)$ та N пар дійсних чисел – координати точок на площині. Програма виконує один за алгоритмів згідно з варіантом.

Варіанти завдань

1. Точки описують ламану. Потрібно для кожної нової ланки вказати, направо чи наліво здійснено поворот.

2. Точки описують ламану. Потрібно для кожної нової ланки вказати, чи перетинає вона будь-яку з попередніх.

3. Точки є вершинами багатокутника в порядку обходу. Вивести площу багатокутника.

4. Точки є вершинами багатокутника. Повідомити, за чи проти годинникової стрілки здійснено обхід.

5. Точки є вершинами багатокутника в порядку обходу. Повідомити, чи є він опуклим.

6. Побудувати опуклу оболонку заданих точок алгоритмом Грехема.

7. Побудувати опуклу оболонку заданих точок алгоритмом Джарвіса.

Лабораторна робота 7

Тема: Фундаментальні алгоритми на графах і деревах.

Мета роботи: ознайомитися зі способами подання графів та отримати навички програмування алгоритмів, що їх обробляють.

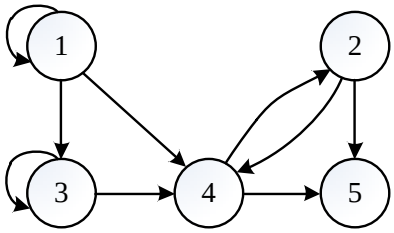
Короткі теоретичні відомості про алгоритми, які використовуються в лабораторній роботі, наведені в підрозділі **1.7. Фундаментальні алгоритми на графах і деревах** даного посібника.

Рекомендується також для детальнішого вивчення даної теми ознайомитися з відповідними розділами в рекомендованій літературі [1, 2, 8, 11, 12].

Для подання графів у даній лабораторній роботі необхідно використовувати матрицю суміжності чи список суміжності залежно від завдання.

Матриця суміжності графа G з n вершинами є квадратною матрицею A розміру $n \times n$, в якій значення її елемента a_{ij} дорівнює числу ребер з i-ї в j-ту вершину графа (рис. 2.1).

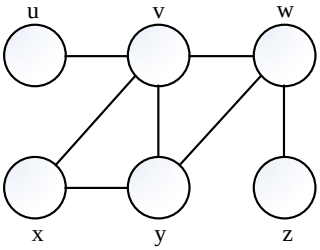
Подання графа $G = (V;\ E)$ у вигляді списків суміжних вершин використовує масив Adj із списків – по одному на вершину [1]. Для кожної вершини список суміжних вершин містить в довільному порядку всі суміжні до неї вершини (рис. 2.2).



$A =$

	1	2	3	4	5
1	1	0	1	1	0
2	0	0	0	1	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	0	0

а

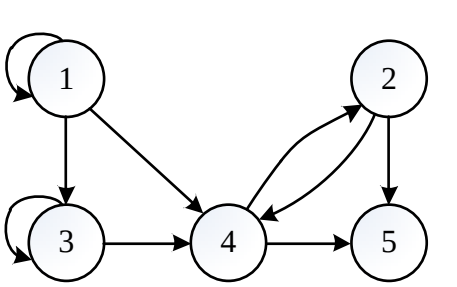


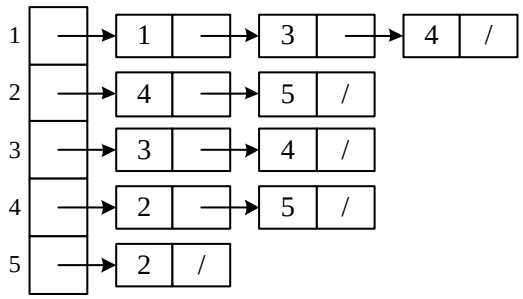
$A =$

	u	v	w	x	y	z
u	0	1	0	0	0	0
v	1	0	1	1	1	0
w	0	1	0	0	1	1
x	0	1	0	0	1	0
y	0	1	1	1	0	0
z	0	0	1	0	0	0

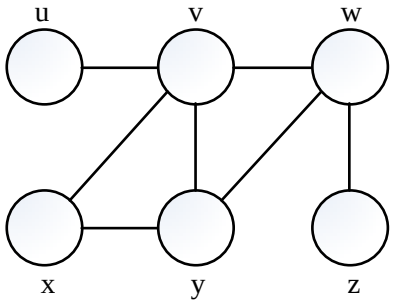
б

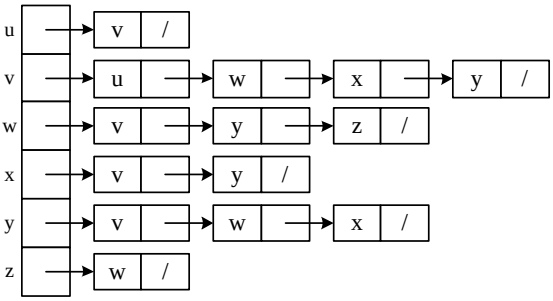
Рисунок 2.1 – Матриця суміжності графа G:
а – орієнтований граф; *б* – неорієнтований граф





а





б

Рисунок 2.2 – Подання графа за допомогою списку суміжних вершин:
а – орієнтований граф; *б* – неорієнтований граф

Запитання для самоконтролю

1. Дайте визначення орієнтованого графа.
2. Який граф називається ациклічним?
3. Які вершини називаються суміжними?
4. Які способи подання графа ви знаєте?

Завдання на лабораторну роботу

Розробити програму, яка читає з клавіатури числа $N, M \left(1 < N, M < 256\right)$ – кількість вершин та ребер графа; послідовність M пар цілих чисел – ребра графа. Програма зберігає граф та виконує над ним алгоритм згідно з варіантом.

Варіанти подання графів

1. Матриця суміжності.
2. Список суміжності.

Варіанти алгоритмів

1. Визначити, чи граф є деревом. Використати пошук у ширину. На екран потрібно вивести вершини у порядку обходу. Для кожної вказати час прибуття та предка в дереві обходу.

2. Визначити, чи граф є деревом. Використати пошук у глибину. На екран потрібно вивести вершини у порядку обходу. Для кожної вказати час початку розгляду, кінця розгляду та предка в дереві обходу.

3. Визначити, чи граф є зв’язаним. Використати пошук у ширину. На екран потрібно вивести вершини у порядку обходу. Для кожної вказати час прибуття та предка у дереві обходу.

4. Визначити, чи граф є зв’язаним. Використати пошук у глибину. На екран потрібно вивести вершини у порядку обходу. Для кожної вказати час початку розгляду, кінця розгляду та предка в дереві обходу.

5. Топологічне сортування. На екран потрібно вивести ті ж дані, що і для пошуку в глибину, а також результат сортування.

6. Відшукати будь-який цикл у графі. Використати пошук у глибину.

Варіанти завдань

1. 1.1.
2. 1.2.
3. 1.3.

4. 1.4.
5. 1.5.
6. 1.6.
7. 2.1.
8. 2.2.
9. 2.3.
10. 2.4.
11. 2.5.
12. 2.6.

3. КУРСОВІ ПРОЕКТИ

3.1. Мета і задачі виконання курсового проекту

Курсовий проект виконується з метою закріплення знань з дисципліни «Алгоритми і структури даних» і отримання навичок розробки програмного забезпечення, що вимагає застосовування вивчених структур даних та алгоритмів.

Завданнями курсового проекту є:

– набуття навичок у розробці програм, які включають нетривіальні алгоритми;

– розробка алгоритмічного забезпечення;

– розробка програмного забезпечення.

Це включає:

– вибір платформи (операційної системи) для програмного забезпечення;

– вибір засобів розробки прикладного програмного забезпечення;

– розробку прикладного програмного забезпечення;

– налагоджування та тестування прикладного програмного забезпечення;

– дослідну експлуатацію прикладного програмного забезпечення на контрольних даних і аналіз отриманих результатів.

Результати виконання курсового проекту надаються у вигляді розробленого програмного забезпечення, пояснювальної записки до курсового проекту та презентаційних матеріалів.

3.2. Завдання для курсового проектування

Нижче наведено типові завдання для курсового проектування.

Варіант 1. Тема: «Реалізація та аналіз якості генераторів випадкових чисел»

Вхідні дані: довжина послідовності, яку необхідно згенерувати, N і початкові параметри, якщо вони потрібні генератору; $N = 100$ для тестів і $N = 10\,000$ для демонстрації.

Вимоги до інтерфейсу: користувач повинен мати можливість переглянути всю згенеровану послідовність, поспостерігати її графічні інтерпретації:

– гістограму розподілу елементів послідовності;

– розподіл на площині (елементи попарно обробляються як координати точок (x, y));

– автокореляцію (користувач задає зсув для копії послідовності).

Також необхідно виконати перевірку згідно з будь-яким (на вибір студента) чотирма тестами з пакета NIST.

Алгоритми, що використовуються: лінійний конгруентний метод чи метод Фібоначчі із затримуванням, а також: генератор із стандартної бібліотеки мови програмування, читання з файлів $/dev/random$, $/dev/urandom$ та довільного іншого файлу.

Варіант 2. Тема: «Візуалізація базових алгоритмів пошуку на графах»

Вхідні дані: граф з кількістю вершин N, заданий списком своїх M ребер; $N, M = 10$ для тестів та $N = 256,\ M = 2048$ для демонстрації.

Джерело даних: обраний довільно відеофайл, дані якого обробляються, як послідовність пар однобайтових чисел – номерів вершин, поєднаних ребром.

Вимоги до інтерфейсу: користувач повинен мати можливість переглянути граф на площині та переміщувати його вершини за допомогою миші для кращого вигляду; результати переміщення, так само, як і сам граф, можна зберегти у файлі у форматі, обраному студентом (можна базуватися на *plain text*, *XML, JSON*); користувач повинен мати можливість обрати вершину та запустити процес пошуку з неї, поряд з кожною вершиною повинні виводитися її статус (біла, сіра, чорна) та час входу та виходу.

Алгоритми, що використовуються: пошук у глибину та пошук у ширину.

Варіант 3. Тема: «Розробка прикладного застосування для архівації даних на базі кодів Хаффмана»

Вхідні дані: файл, що обирає користувач, довжина блоку.

Вимоги до інтерфейсу: користувач має можливість виконати стискання даних та розпакувати архів, переглянути статистику густоти кожного символу, додатково вказати довжину блоків, на які розділяється файл при стисканні;

програма повинна мати також командний інтерфейс, який є подібним до архіваторів *gzip*, *bzip2* або *lzma*.

Алгоритми, що використовуються: алгоритм Хаффмана.

Варіант 4. Тема: «Побудова кістяка для графа транспортних маршрутів»

Вхідні дані: граф з кількістю вершин N, заданий списком своїх M ребер; $N, M = 10$ для тестів та $N = 256,\ M = 2048$ для демонстрації.

Джерело даних: обраний довільно відеофайл, дані якого обробляються як послідовність пар однобайтових чисел – номерів вершин, поєднаних ребром.

Вимоги до інтерфейсу: користувач повинен мати можливість переглянути граф на площині та переміщувати його вершини за допомогою миші для кращого вигляду; результати переміщення, так само, як і сам граф, можна зберегти у файлі у форматі, обраному студентом (можна базуватися на *plain text, XML, JSON*); користувач повинен мати можливість переглянути процес побудови кістяка крок за кроком та отримати результат.

Алгоритми, що використовуються: алгоритм Прима чи алгоритм Крускала.

Варіант 5. Тема: «Дослідження ефективності квадратичних та оптимальних алгоритмів сортування»

Вхідні дані: масив з N цілих чисел, $N = 10$ для тестів і $N = 10\,000\,000$ для демонстрації.

Джерело даних: обраний довільно відеофайл, дані якого обробляються як послідовність 32-бітних чисел у форматі Big Endian.

Вимоги до інтерфейсу: користувач повинен мати можливість переглянути кожний крок виконання будь-якого алгоритму сортування та отримати статистичні відомості: скільки перестановок елементів довелося виконати, скільки мілісекунд знадобилося для сортування тощо.

Алгоритми, що використовуються: сортування вставками, бульбашкою, швидке, злиттям, купою (один квадратичний та один оптимальний).

Варіант 6. Тема: «Побудова опуклої оболонки на площині»

Вхідні дані: набір з N точок, заданий своїми цілочисловими координатами; $N = 20$ для тестів та $N = 256$ для демонстрації.

Джерело даних: обраний довільно відеофайл, дані якого обробляються як послідовність пар однобайтових чисел – координат вершин.

Вимоги до інтерфейсу: користувач повинен мати можливість переглянути точки на площині, переміщувати їх за допомогою миші; результати переміщення можна зберегти у файлі у форматі, обраному студентом (можна базуватися на *plain text, XML, JSON*); користувач повинен мати можливість переглянути крок за кроком процес побудови; користувач повинен мати можливість вказати за допомогою миші довільну точку на площині, щоб програма повідомила, чи належить вона до багатокутника, що відповідає оболонці.

Алгоритми, що використовуються: алгоритм Джарвіса чи алгоритм Грехема.

Варіант 7. Тема: «Пошук найкоротшого шляху в двовимірному лабіринті»

Вхідні дані: набір з N стін, заданий двома парами цілочислових координат своїх вершин; $N = 20$ для тестів та $N = 50$ для демонстрації.

Джерело даних: обраний довільно відеофайл, дані якого обробляються як послідовність однобайтових чисел – координат вершин.

Вимоги до інтерфейсу: користувач повинен мати можливість переглянути стіни на площині, переміщувати їх за допомогою миші; користувач має можливість обрати дві точки, щоб програма побудувала між ними найкоротший шлях, що не перетинає жодної стіни, або повідомила, що такого шляху не існує; має бути можливість переглянути етапи побудови маршруту.

Алгоритми, що використовуються: алгоритм Дейкстри чи алгоритм Флойда-Воршолла.

Варіант 8. Тема: «Визначення порядку збирання програмного комплексу з урахуванням залежностей між компонентами»

Вхідні дані: направлений ациклічний граф (DAG).

Джерело даних: відомості про залежність програм на комп’ютері користувача, наприклад, з файлу $/var/lib/dpkg/available$.

Вимоги до інтерфейсу: користувач повинен мати можливість переглянути граф на площині та переміщувати його вершини за допомогою миші для кращого вигляду; користувач повинен мати можливість переглянути процес побудови порядку збирання програм та зберегти результати у файлі.

Алгоритми, що використовуються: топологічне сортування.

Варіант 9. Тема: «Реалізація шифрованого обміну повідомленнями»

Вхідні дані: текстовий файл, що обирає користувач, параметри алгоритму.

Вимоги до інтерфейсу: користувач має можливість запустити дві копії програми на різних комп'ютерах, щоб вони обмінювалися даними через мережу, або зберегти зашифроване повідомлення у файлі з можливістю розшифрувати його.

Алгоритми, що використовуються: RSA або шифр Віжинера.

Варіант 10. Тема: «Наближене розв'язання задачі комівояжера»

Вхідні дані: N міст на площині, заданих своїми цілочисловими координатами; $N = 20$ для тестів та $N = 256$ для демонстрації.

Джерело даних: обраний довільно відеофайл, дані якого обробляються як послідовність однобайтових чисел – координат вершин.

Вимоги до інтерфейсу: користувач повинен мати можливість переглянути місця на площині, переміщувати їх за допомогою миші; користувач має можливість переглянути етапи побудови маршруту комівояжера та результат із відображенням довжини маршруту та його послідовності на площині.

Алгоритми, що використовуються: наближене розв'язання задачі комівояжера.

Варіант 11. Тема: «Кластеризація живих організмів на підставі ступеня близькості їхньої ДНК»

Вхідні дані: N зразків ДНК довжиною M нуклеотидів різноманітних живих організмів; $N,\ M = 20$ для тестів та $N,\ M = 256$ для демонстрації.

Джерело даних: обраний довільно відеофайл, дані якого обробляються як послідовність пар бітів, кожна з яких кодує один нуклеотид (аденін, тимін, гуанін, цитозин).

Вимоги до інтерфейсу: користувач повинен мати можливість обрати пару зразків ДНК та отримати їхню найбільшу спільну підпослідовність, а також побудувати дендрограму (кладограму) для всіх розглянутих організмів.

Алгоритми, що використовуються: пошук найбільшої спільної підпослідовності, агломеративний чи дивізимний алгоритм ієрархічної кластеризації.

3.2. Основні етапи і календарний план виконання курсового проекту

Виконання курсового проекту розбивається на такі етапи:

– постановка задачі дослідження;

– побудова математичної моделі (за необхідності);

– вивчення теоретичних основ алгоритмів та поглиблене вивчення структур даних, які будуть використовуватись;

– розробка та тестування алгоритмічного та програмного забезпечення;

– розв’язання задачі та аналіз результатів;

– оформлення пояснювальної записки та презентації;

– захист курсового проекту.

Усі етапи необхідно виконувати в терміни, вказані в графіку виконання (табл. 3.1).

Таблиця 3.1 – Календарний план виконання курсового проекту

№ з/п	Найменування етапу	Терміни виконання
1	Вивчення теоретичних основ використаних методів. Постановка задачі дослідження.	25 лютого
2	Алгоритмічне забезпечення розв’язання задачі дослідження	15 березня
3	Розробка структури програмного забезпечення	30 березня
4	Розробка програмного забезпечення. Програмна реалізація методів, які використовуються	
5	Розв’язання задачі та аналіз отриманих результатів	10 квітня
6	Оформлення пояснювальної записки	17 квітня
7	Підготовка презентації курсового проекту	22 квітня
8	Захист курсового проекту	15 травня

Дослідження підрозділяється на три етапи.

На підготовчому етапі дослідження кожен студент повинен:

– ознайомитись зі змістовною постановкою отриманої задачі;

– вивчити алгоритми та структури даних, необхідні, щоб виконати завдання;

– розробити алгоритмічне та програмне забезпечення;

– побудувати набори тестових даних.

На робочому етапі дослідження студент повинен:

– для отриманих тестових даних на основі розробленого програмного забезпечення розв'язати поставлену задачу, щоб перевірити правильність побудованих алгоритмів;

– розв'язати поставлену задачу для даних великої розмірності, щоб визначити межі використання програмного забезпечення.

На завершальному етапі дослідження студент повинен:

– інтерпретувати отримані результати в термінах предметної галузі змістовної постановки задачі;

– зробити висновки за результатами виконання роботи.

3.3. Вимоги до програмного забезпечення, що розробляється

Вибір засобів розробки прикладного програмного забезпечення необхідно виконувати із урахуванням таких чинників:

– наявність досвіду роботи з цим програмним продуктом;

– доступність цього програмного продукту, зокрема, наявність ліцензій;

– ефективність його використання при розробці прикладного програмного забезпечення;

– швидке візуальне проектування компонентів прикладного програмного забезпечення;

– зручність налагодження;

– можливість отримання консультацій.

Застосунок, що розробляється, повинен реалізовувати такі функції:

– введення, редагування і збереження вихідних даних;

– розв'язання основної задачі;

– перевірка отриманого розв'язку;

– перегляд пошуку розв'язку крок за кроком;

– збереження звіту про виконані дослідження;

– отримання довідки.

Розглянемо їх детальніше.

Введення, редагування і збереження початкових даних

Необхідно забезпечити введення початкових даних вручну або з файлу, редагування даних та їх збереження у файл. Формат файлу довільний. Рекомендовано використовувати текстовий формат (plain text, XML, JSON).

Багато варіантів завдань потребують читання даних із великого бінарного файлу. Збереження у бінарному форматі можна не реалізовувати, але збереження у базовому форматі (текстовому) потрібно підтримувати.

Розв'язання основної задачі

Необхідно забезпечити розв'язання задачі заданими алгоритмами. У разі несумісних умов або некоректного формату вхідних даних застосунок повинен повідомити про це користувача. Програма повинна функціонувати так, що для будь-якого входу вона або видає розв'язок, або повідомляє про помилку.

Крім того, застосунок повинен надавати можливість перегляду процесу пошуку розв'язку крок за кроком, навіть якщо задача має несумісні умови, що дозволяють виконати кілька кроків алгоритму.

Перевірка отриманого розв'язку

Багато варіантів курсового проекту допускають перевірку отриманого розв'язку.

Перевірка може бути виконана користувачем за допомоги графічної інтерпретації, наприклад, елементи відсортованої послідовності можна зобразити у вигляді гістограми.

Такі завдання, як генерація випадкових послідовностей, включають формальні тести якості отриманого результату.

Збереження звіту про виконані дослідження

Програма повинна забезпечувати збереження звітів про отриманий розв'язок. Рекомендуються формати plain text, HTML, LaTeX (із компіляцією до PDF) або PDF.

До звіту про отриманий розв'язок необхідно внести:

– умову задачі;

– кроки алгоритму;

– розв’язок;

– графічну або табличну інтерпретацію результату.

Отримання довідки

Користувач повинен мати можливість ознайомитися з інформацією, достатньою для використання можливостей прикладного програмного забезпечення і режимами їх реалізації. Формат довідки довільний. Рекомендовано застосовувати HTML або PDF. Можливо написання довідки у форматах reStuctured Text або LaTeX із наступною компіляцією до HTML або PDF.

3.4. Вимоги до змісту пояснювальної записки до курсового проекту

Розглянемо зразкову структуру записки до курсового проекту. Записка складається з вступу, чотирьох розділів, висновків і додатків. У додатки можна помістити ілюстративний матеріал, таблиці, формули. Мінімальний обсяг записки – 40 сторінок.

До записки до курсового проекту необхідно додавати носій, що містить повну інформацію за матеріалами, розробленими в результаті виконання курсового проекту:

– вихідні коди розробленого програмного забезпечення;

– виконувані файли прикладного програмного забезпечення, наприклад, форматів ELF або JAR, додавати Portable Executable (.exe) не рекомендується;

– повний текст записки.

На рис. 3.1 показано зміст пояснювальної записки.

Варіант, що наводиться, є рекомендаційним. Він відображає той необхідний обсяг інформації, який має бути наведений в записці. При виконанні курсового проекту склад і зміст розділів записки можуть бути змінені студентом за узгодженням з керівником.

Дамо коротку характеристику змісту розділів і підрозділів записки.

У ***вступі*** розглядаються цілі, які мають бути досягнуті в результаті виконання курсового проекту, обґрунтовується актуальність теми.

Основна мета ***першого розділу «Теоретичні основи розробки та аналізу алгоритмів. Постановка задачі»*** – показати, що студент ознайомився з ос-

новами теорії алгоритмів та необхідними структурами даних, і сформулювати основні вимоги до програмного забезпечення, що розробляється.

Вступ
1. Теоретичні основи розробки та аналізу алгоритмів. Постановка задачі
   1.1. Математичні основи аналізу алгоритмів
   1.2. Структури даних і класифікація основних алгоритмів
   1.3. Постановка задачі
2. Теоретичні основи розробки програмного забезпечення
   2.1. Структури даних і класифікація основних алгоритмів заданого класу
   2.2. Опис використаних алгоритмів
3. Опис розробленого програмного забезпечення
   3.1. Визначення варіантів використання програмного забезпечення
   3.2. Обґрунтування вибору операційної системи і засобів розробки прикладного програмного забезпечення
   3.3. Структура застосунку
4. Використання розробленого програмного забезпечення
   4.1. Установлення програмного забезпечення
   4.2. Інструкція користувачеві
   4.3. Аналіз результатів
Висновки
Список джерел інформації

Рисунок 3.1 – Зміст пояснювальної записки

У ***підрозділі «1.1 Математичні основи аналізу алгоритмів»*** необхідно дати визначення таким основним поняттям теорії алгоритмів, як складність алгоритму, алгоритми «розділяй-та-володарюй», динамічне програмування, амортизаційний аналіз та ін.

У ***підрозділі «1.2 Структури даних і класифікація основних алгоритмів»*** треба дати характеристику структур даних і класифікацію основних алгоритмів: алгоритми на графах, динамічне програмування, геометричні алгоритми тощо.

У ***підрозділі «1.3 Постановка задачі»*** необхідно:

– навести математичну модель задачі;

– сформулювати якісну постановку задачі;

– описати основні цілі, які потрібно досягти в результаті виконання курсового проекту.

Другий розділ «Теоретичні основи розробки програмного забезпечення» присвячується опису математичного забезпечення розв'язання задачі.

У ***підрозділі «2.1 Структури даних і класифікація основних алгоритмів заданого класу***» слід охарактеризувати заданий клас алгоритмів та структури даних, наприклад, описати алгоритми на графах та способи подання графа в пам'яті.

У ***підрозділі «2.2 Опис використаних алгоритмів***» необхідно описати конкретні алгоритми, які буде використано в проекті. Слід навести схеми алгоритмів у вигляді добре структурованого псевдокоду або справжнього коду. Показати, як визначається складність описаних алгоритмів та навести приклади роботи алгоритму на простих даних.

Основна мета ***третього розділу «Опис розробленого програмного забезпечення»*** – описати результати, отримані в процесі розробки інформаційного і програмного забезпечення для розв'язання прикладних задач.

У ***підрозділі «3.1 Визначення варіантів використання програмного забезпечення»*** необхідно визначити основні типи користувачів, що працюють з прикладним програмним забезпеченням, сформулювати і описати варіанти використання. Необхідно навести діаграму варіантів використання.

У ***підрозділі «3.2 Обґрунтування вибору операційної системи і засобів розробки прикладного програмного забезпечення»*** мають бути розглянуті можливості 2-3 операційних систем та 2-3 засобів розробки прикладного програмного забезпечення. Мають бути наведені основні переваги і недоліки даних програмних продуктів.

Підрозділ повинен закінчуватися обґрунтуванням вибору операційної системи та засобів розробки прикладного програмного забезпечення.

У ***підрозділі «3.3 Структура застосунку»*** мають бути описані основні елементи прикладного програмного забезпечення і взаємозв'язку між ними. Необхідно надати UML-діаграму компонентів і діаграму класів та описати їх словами.

Діаграма компонентів показує різні компоненти системи і зв'язки між ними. Компонент є фізичним модулем програмного коду. Компонент часто вважають синонімом пакета, але ці поняття можуть відрізнятися, оскільки компоненти є фізичним об'єднанням програмного коду. Хоча окремий клас може бути поданий в цілій сукупності компонентів, цей клас має бути визначений тільки в одному пакеті. Наприклад, клас String в мові Java є частиною

пакета java.lang, але він може міститися у ряді компонентів.

Залежності між компонентами показують, як зміни одного компонента можуть вплинути на зміни інших компонентів. Існує досить обмежена кількість видів залежностей, які можна використовувати, включаючи залежності типу «зв'язок» і «компіляція».

Діаграма класів описує типи об’єктів системи і різного роду статичні стосунки, які існують між ними. Є два основні види статичних відносин :

– асоціації (наприклад, клієнт може взяти напрокат ряд відеокасет);

– підтипи (медсестра є різновидом особи).

На діаграмах класів зображуються також атрибути класів, операції класів та обмеження, які накладаються на зв’язки між об’єктами.

Основна мета ***четвертого розділу*** «***Використання розробленого програмного забезпечення***» – описати результати, отримані в процесі використання розробленого інформаційного і програмного забезпечення для розв’язання прикладних задач. У цьому розділі описується послідовність роботи користувача з прикладним програмним забезпеченням, а також описуються і аналізуються результати, отримані при розв’язанні практичних задач.

У ***підрозділі «4.1 Установлення програмного забезпечення***» описується процедура установлення розробленого прикладного програмного забезпечення на комп’ютер користувача. Процедуру установлення рекомендується описати у вигляді послідовності кроків, що виконуються користувачем у процесі установки. Рекомендується також навести основні вимоги до апаратного і програмного забезпечення: тип операційної системи, наявність спеціального програмного забезпечення, тип процесора, мінімальний обсяг оперативної пам’яті, наявність вільного місця на жорсткому диску і т.д.

Якщо обрано операційну систему GNU/Linux, рекомендовано виконати пакування програмного забезпечення (rpm, deb, або ebuild) – це значно спрощує установку залежностей програми та її видалення із системи.

Необхідно описати послідовність дій при першому запуску прикладного програмного забезпечення (тобто як запустити програму і як з неї вийти).

У ***підрозділі «4.2 Інструкція користувачеві»*** наводиться опис роботи користувача з розробленим прикладним програмним забезпеченням.

Цей опис можна розглядати як фрагмент посібника користувача, що входить в комплект документації на програмний продукт. Опис має бути до-

сить детальним, дозволяти працювати з прикладним програмним забезпеченням користувачеві, що не має спеціальної підготовки. Опис повинен ілюструватися прикладами екранних форм, які розташовуються в тексті у вигляді рисунків.

У ***підрозділі «4.3 Аналіз результатів»*** наводиться опис результатів, отриманих при використанні прикладного програмного забезпечення. Потрібно навести інтерпретацію фрагмента або всіх вхідних даних великої розмірності у вигляді таблиці чи рисунка. Можна додати відповідні скріншоти екранних форм. Потрібно показати результат та проаналізувати його.

У ***висновках*** підбиваються підсумки виконання курсового проекту, коротко перераховуються отримані результати, робляться висновки про працездатність розробленого застосування і доцільність його використання для вирішення практичних завдань.

У ***розділі*** «***Список джерел інформації»*** наводиться перелік цитованих, розглянутих, згаданих і використаних джерел інформації. Джерелами інформації є книги, статті, нормативно-технічні документи, дисертації і т.д. Джерела інформації записують, не згруповуючи їх на ті, що мають та не мають посилань у тексті.

3.5. Захист курсового проекту і критерії оцінювання

До захисту курсового проекту допускаються студенти, що виконали курсовий проект в повному обсязі, про що свідчить записка до курсового проекту, підписана керівником.

Головні висновки про відповідність змісту курсового проекту завданню, міри самостійності виконання проекту і т.д. мають бути вказані у відгукові керівника.

Студент повинен представити презентаційні матеріали та прикладне програмне забезпечення, розроблене в процесі виконання курсового проекту.

Захист курсового проекту проходить з обов’язковим використанням комп’ютерної техніки. При підготовці до захисту студент повинен завчасно встановити на наданому йому комп’ютері презентаційні матеріали та прикладне програмне забезпечення. Захист курсового проекту починається з доповіді студента, після закінчення якої він повинен відповісти на поставлені йому запитання за темою проекту.

Захист курсового проекту є публічним, тобто на захисті можуть бути

присутніми усі, хто бажає, і ставити будь-які запитання за темою курсового проекту.

Матеріали, що використовуються як презентаційні, мають бути розміщені у тексті записки до курсового проекту у вигляді рисунків, таблиць, схем і т.п. Якщо презентаційні матеріали в тексті записки відсутні (наприклад, якщо в цих матеріалах надана інформація з різних розділів записки до курсового проекту), то вони мають бути наведені в додатках до записки.

Зразковий перелік презентаційних матеріалів такий:

1) постановка задачі дослідження;
2) класифікація основних алгоритмів;
3) опис класу поданих алгоритмів;
4) схема роботи реалізованих алгоритмів;
5) діаграма варіантів використання;
6) вимоги до програмного забезпечення;
7) середовище розробки;
8) введення даних;
9) результати: графічна або таблична інтерпретація;
10) результати: перевірка розв’язку;
11) висновки.

Презентаційні матеріали можуть бути виконані в паперовому або електронному вигляді. Паперові презентаційні матеріали виконуються на папері формату А4 в машинописному вигляді (тобто мають бути надруковані на принтері). Усі написи і рисунки мають бути чіткими, добре читатися. Рукописний варіант презентаційних матеріалів не допускається. Електронні презентаційні матеріали виконуються з використанням відповідного програмного забезпечення (LibreOffice Impress, Microsoft Power Point і т.п.) і демонструються за допомогою комп’ютера.

Вимоги до доповіді. Мета доповіді – викласти цілі курсового проекту, виділити і охарактеризувати основні етапи його виконання та отримані результати. Час доповіді – до 5 хвилин. Під час доповіді студент повинен користуватися презентаційними матеріалами. Після завершення доповіді студент переходить до демонстрації розробленого програмного забезпечення.

Основне завдання демонстрації програмного забезпечення – показати працездатність розробленого програмного забезпечення, його основні функціональні можливості, зручність роботи користувача і т.д.

Під час демонстрації студент повинен показати, як працювати з програмним забезпеченням у режимі:

– введення та модифікації початкових даних;

– побудови розв’язку задачі;

– демонстрації та перевірки розв’язку.

На оцінку курсового проекту впливають такі фактори:

1) помилки, збої тощо у роботі прикладного програмного забезпечення, виявлені при його демонстрації в процесі захисту курсового проекту;

2) неякісна підготовка доповіді студентом;

3) неякісні презентаційні матеріали, які не повною мірою відображають особливості предметної галузі, результати, що отримані при виконанні курсового проекту тощо.

При оцінюванні роботи також враховується відгук наукового керівника, в якому повинні бути відбиті такі дані:

– ступінь самостійності виконання роботи студентом;

– основні результати, що отримані при виконанні роботи;

– оцінка роботи за національною шкалою і шкалою ECTS.

ВИСНОВКИ

Навчальний посібник є наслідком узагальнення більше ніж десятирічного досвіду викладання дисциплін «Теорія алгоритмів» і «Алгоритми і структури даних» для студентів спеціальностей «Комп'ютерні науки» та «Інженерія програмного забезпечення» на кафедрі програмної інженерії та інформаційних технологій управління Національного технічного університету «Харківський політехнічний інститут». Структури даних і алгоритми розглядаються у великій кількості літературних джерел, частина яких наведена в бібліографічному вказівникові, всі вони тою чи іншою мірою були використані при підготовці даного навчального посібника.

Особливістю даного практикуму є те, що він є цілісною системою, що об'єднує в собі практичні, лабораторні заняття, курсове проектування, завдання для практичної самостійної роботи студентів, що визначаються основними компетентностями бакалавра комп'ютерних наук і програмної інженерії.

Головним підсумком навчального посібника є надія авторів на те, що викладачі, аспіранти, студенти вищого навчального закладу, які використають описаний авторський педагогічний підхід, досягнуть суттєвих успіхів у формуванні практичних навичок реалізації основних алгоритмів і структур даних, які найчастіше використовуються при розробці програмного забезпечення.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Основна

1. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн ; пер. с англ. под ред. А. Шеня. – 3-е изд. – М. : ООО «И. Д. Вильямс», 2013. – 1328 с.

2. Ахо, Альфред, В., Структуры данных и алгоритмы / Ахо, Альфред, В., Хопкрофт, Джонс, Ульман, Джефри, Д. : пер. с англ. – М. : Вильямс, 2003. – 384 с.

3. Препарата Ф., Шеймос М. Вычислительная геометрия: Введение / Ф. Препарата, М. Шеймос : пер. с англ. – М. : Мир, 1989. – 478 с.

4. Кнут, Дональд, Эрвин. Искусство программирования. Том 1. Основные алгоритмы, 3-е изд. / Кнут, Дональд, Эрвин : пер. с англ. – М. : ООО «И. Д. Вильямс», 2015. – 720 с.

5. Кнут, Дональд, Эрвин. Искусство программирования. Том 3. Сортировка и поиск, 2-е изд. / Кнут, Дональд, Эрвин : пер. с англ. – М. : ООО «И. Д. Вильямс», 2007. – 832 с.

6. Седжвик, Р. Фундаментальные алгоритмы на C++. Анализ. Структуры данных. Сортировка. Поиск : Части 1–4 / Роберт Седжвик : пер. с англ. – Киев : ДиаСофт, 2001. – 688 с.

7. Вирт, Никлаус. Алгоритмы и структуры даннях. Новая версия для Оберона+CD / Вирт, Никлаус ; пер. с англ. Ткачев Ф.В. – М. : ДПК Пресс, 2010. – 272 с.

8. Гагарина Л.Г. Алгоритмы и структуры данных: учеб. пособ. / Л.Г. Гагарина, В.Д. Колдаев. – М. : Финансы и статистика; ИРФРА-М, 2009. – 304 с.

9. Игошин В.И. Математическая логика и теория алгоритмов : учеб. пособ. для студ. высш. учеб. заведен. / В.И. Игошин. – 2-е изд., стер. – М. : Изд. центр «Академия», 2008. – 448 с.

10. Верещагин Н.К. Лекции по математической логике и теории алгоритмов. Часть 3. Вычислимые функции / Верещагин Н.К., Шень А.В. – 4-е изд., испр. – М. : МЦНМО, 2012. – 160 с.

11. Котов В.М. Алгоритмы и структуры данных : учеб. пособ. / В.М. Котов, Е.П. Соболевская, А.А. Толстиков. – Минск : БГУ, 2011. – 267 с.

Додаткова

12. Евстигнеев В.А. Применение теории графов в программировании / В.А. Евстигнеев. – М. : Наука, 1985. – 352 с.

13. O'Rourke J. Computational geometry in C. – Cambridge University Press. – 1994. – 376 p.

14. Jarvis A. On the identification of the convex hull of a finite set of points in the plane. // Information Processing Letters. – 1973. – Vol. 2. – pp.18–21.

15. Graham R.L. An efficient algorithm for determining the convex hull of a finite planar set // Information Processing Letters. – 1972. – Vol. 1. – pp. 132–133.

16. Иванов М.А., Теория, применение и оценка качества генераторов псевдослучайных последовательностей / М.А. Иванов, И.В. Чугунков. – М. : КУДИЦ-ОБРАЗ, 2003 – 240 с.

17. Коллинз У.Дж. Структуры данных и стандартная библиотека шаблонов / У.Дж. Коллинз : пер. с англ. – М. : Бином, 2004. – 624 с.

18. Курносов М.Г. Введение в структуры и алгоритмы обработки данных : учеб. пособ. / М.Г. Курносов. – Новосибирск : Автограф, 2015. – 179 с.

19. Мейн М. Структуры данных и другие объекты в C++ / М. Мейн, У. Савитч : пер. с англ. – 2-е изд. – М. : Вильямс, 2002. – 832 с.

20. Алексеев В.Е. Графы и алгоритмы. Структуры данных. Модели вычислений : учебник для вузов / В.Е. Алексеев, В.А. Таланов. – М. : Интернет-Университет Информационных Технологий, М. : БИНом, Лаборатория знаний, 2006. – 319 с.

21. Бабенко М.А. Введение в теорию алгоритмов и структур данных / М.А. Бабенко, М.В. Левин. – М. : ФМОП, МЦНМО, 2012. – 144 с.

22. Гудман С. Введение в разработку и анализ алгоритмов. / С. Гудман, С. Хидетниеми ; пер. с англ. под ред. В.В. Мартынюка – М. : Мир, 1981. – 368 с.

23. Трамбле Ж. Введение в структуры данных / Ж. Трамбле, П. Соренсон : пер. с англ. – М. : Машиностроение, 1984. – 784 с.

Навчальне видання

СТРАТІЄНКО Наталія Костянтинівна
ГОДЛЕВСЬКИЙ Михайло Дмитрович
БОРОДІНА Інна Олександрівна

АЛГОРИТМИ І СТРУКТУРИ ДАНИХ: ПРАКТИКУМ

Навчальний посібник
для студентів спеціальностей
«Комп'ютерні науки» та «Інженерія програмного забезпечення»

Відповідальний за випуск проф. Годлевський М. Д.
Роботу до видання рекомендував проф. Горілий О. В.

Редактор О. С. Самініна

План 2017 р., поз. 90.

Підписано до друку 11.10.2017 р. Формат 60x84 1/16. Папір офісний.
Riso-друк. Гарнітура Таймс. Ум. друк. арк. 13,1. Наклад 50 пр.
Зам № 175. Ціна договірна

Видавець і виготовлювач
Видавничий центр НТУ «ХПІ»,
вул. Кирпичова, 2, м. Харків, 61002

Свідоцтво суб'єкта видавничої справи ДК № 5478 від 21.08.2017 р.