Дисципліна: “Програмування складних алгоритмів”

Лабораторна робота №1. СКЛАДНІСТЬ АЛГОРИТМУ.

Мета роботи:

Метою лабораторної роботи є отримання практичних навичок визначати часову складність алгоритму.

Навчитися будувати алгоритми з мінімальною часовою складністю для вирішення поставлених задач.

Завдання до лабораторної роботи:

Використовуючи алгоритми, згідно варіанту завдання, скласти програму розрахунку заданих величин та провести аналіз ефективності реалізованих алгоритмів.

Лабораторна робота №1. СКЛАДНІСТЬ АЛГОРИТМУ.

Методичні вказівки

Лабораторна робота спирається на знання й уміння, отримані при вивченні наступних питань лекції:

– Поняття алгоритму, властивості.

– Поняття складності обчислення. Функція складності обчислень (за часом).

– Класи складності. Опис класів $\boldsymbol{P}$ і $\boldsymbol{NP}$. Приклади $\boldsymbol{NP}$-повних задач. Проблема перебору ($\boldsymbol{P} \stackrel{?}{=} \boldsymbol{NP}$). Застосування теорії $\boldsymbol{NP}$-повноти для аналізу складності завдань.

Лабораторна робота №1. СКЛАДНІСТЬ АЛГОРИТМУ.

Наведемо нижче декілька важливих визначень, які слід пам’ятати під час виконання лабораторної роботи.

Алгоритм – це скінченна послідовність команд, які треба виконати над вхідними даними для отримання результату.

Основними мірами обчислювальною складності алгоритмів є:

– часова складність, яка характеризує час, необхідний для виконання алгоритму на даній машині; цей час, як правило, визначається кількістю операцій, які потрібно виконати для реалізації алгоритму;

– ємнісна складність, яка характеризує пам’ять, необхідну для виконання алгоритму.

Часова та ємнісна складність тісно пов’язані між собою. Обидві є функціями від розміру вхідних даних.

Надалі обмежимося тільки аналізом часової складності.

Складність алгоритму описуванняється функцією $f(n)$, де n – розмір вхідних даних. Важливе теоретичне і практичне значення має класифікація алгоритмів, яка бере до увагу швидкість зростання цієї функції.

Лабораторна робота №1. СКЛАДНІСТЬ АЛГОРИТМУ.

Приклад 1. Розглянемо задачу пошуку найбільшого елементу в списку з $\boldsymbol{n}$ чисел. Для простоти припустимо, що цей список реалізований у вигляді масиву. Нижче приведений псевдокод алгоритму.

Алгоритм MaxElement ($\boldsymbol{A}\,[0, \ldots, n-1]$)

// Вхідні дані: масив дійсних чисел ($\boldsymbol{A}\,[0, \ldots, n-1]$)

// Вихідні дані: повертається значення найбільшого елементу масиву $\boldsymbol{A}$

$max \leftarrow A\,[0]$
$for\ i \leftarrow 1\ to\ n-1\ do$
    $if\ \ A\,[i] > max$
    $max \leftarrow A\,[i]$
$return\ max$

Позначимо через $\boldsymbol{C(n)}$ кількість виконуваних в алгоритмів операцій порівняння та спробуємо вивести формулу, що виражає їх залежність від розміру вхідних даних $\boldsymbol{n}$.

$$\boldsymbol{C(n) = \sum_{i=1}^{n-1} 1 = n-1 \in O(n)}$$

Лабораторна робота №1. СКЛАДНІСТЬ АЛГОРИТМУ.

Приклад 2. Розглянемо задачу знаходження максимального елемента в матриці.

Алгоритм MaxElement ($\boldsymbol{A}\,[0,\ldots,n-1][0,\ldots,n-1]$)

// Вхідні дані: масив дійсних чисел ($\boldsymbol{A}\,[0,\ldots,n-1][0,\ldots,n-1]$)

// Вихідні дані: повертається значення $\boldsymbol{max}$

$Max\ =\ A\,[0][0];$
$for\ (i\ =0; i<\ n; i++)$
$\quad for\ (j\ =0; j<\ n; j++)\ \{$
$\quad\quad if\ \ (Max\ <A\,[i][j])$
$Max\ =\ A\,[i][j];\ \}$

Позначимо через $\boldsymbol{C_m(n)}$ кількість виконуваних в алгоритмів операцій порівняння та спробуємо вивести формулу, що виражає їх залежність від розміру вхідних даних n^2.

$$C_m\ =\ \sum_{\boldsymbol{i=0}}^{\boldsymbol{n-1}}\sum_{\boldsymbol{j=0}}^{\boldsymbol{n-1}}\boldsymbol{1}\ =\ \boldsymbol{n*\ n\ =\ n^2\ =\ O(n^2)}$$

Лабораторна робота №1. СКЛАДНІСТЬ АЛГОРИТМУ.

Приклад 3. Розглянемо задачу перевірки єдиності елементів. Іншими словами, треба переконатися, що усі елементи масиву різні. Цю задачу можна вирішити за допомогою приведеного нижче нескладного алгоритму.

Алгоритм Elements ($\boldsymbol{A}\,[0, \ldots, n-1]$)

// Вхідні дані: масив дійсних чисел ($\boldsymbol{A}\,[0, \ldots, n-1]$)

// Вихідні дані: повертається значення ***true,*** якщо усі елементи масиву ***A*** різні, та ***false*** у іншому випадку

```
boolean l = true
for (i = 0; i < n − 2; i ++) {
    for (j = i + 1; j < n − 1; j ++) {
        if (A [i] = A [j])
            l = false; }}
return l;
```

Позначимо через $\boldsymbol{C_w}$ кількість виконуваних в алгоритмів операцій порівняння та спробуємо вивести формулу, що виражає їх залежність від розміру вхідних даних $\boldsymbol{n}$.

$$C_w = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 \approx \frac{1}{2}\, n^2 \in \theta(n^2)$$

Завдання та приклад виконання

Сформувати двовимірний масив цілих чисел, використовуючи датчик випадкових чисел. Використати оголошення масиву на ***n*** елементів (кількість елементів задавати з екрану). Провести оцінку часової складності алгоритму, використовуючи ***O***-символіку в найгіршому або/і в середньому. Порівняти час роботи та кількість ітерацій/кількість кроків роботи програми з різною розмірністю масиву (10x10, 50x50, 100x100, 500x500).

Оцінку часу роботи навести у вигляді графіка, або таблиці.

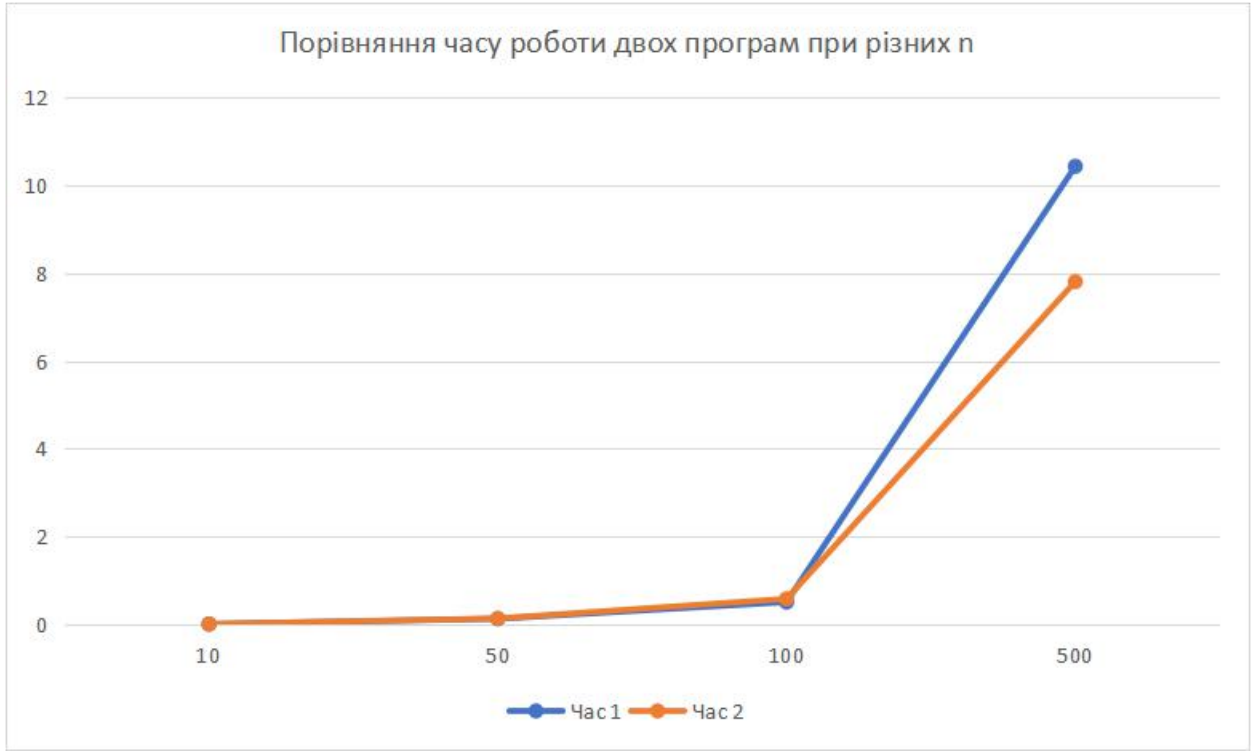
```
Enter a number of the task you want to see(1-2): 1
Choose array size(1-4):
1. 10x10
2. 50x50
3. 100x100
4. 500x500
1
Generating 10x10 array for task 1...
 -4  -9  -4  11  14   0   3  -5   1   4
  1  -1 -14   9 -13  -6  15   2   9  -4
 -1  15  -1 -15  -5  10   5   9   0 -18
 -6   0   5 -18   1 -10   3  -9 -18  -3
  7 -17   1 -18  -5   6   0   3  -5  -6
 -3  -7  -4 -10 -13   2   3   2   3   8
  9  10  -2   5   3   1   6   5  -1 -13
  3  10   4   1  12   3 -10 -16   7   4
 10  -8   1  -2  -4  -7   0 -18   5  -4
 -1  -2 -10  -3 -16   4   1   2   9  11

Redacted
Reversed lines: 1 2 3 4 5 7 8

  4   1  -5   3   0  14  11  -4  -9  -4
 -4   9   2  15  -6 -13   9 -14  -1   1
-18   0   9   5  10  -5 -15  -1  15  -1
 -3 -18  -9   3 -10   1 -18   5   0  -6
 -6  -5   3   0   6  -5 -18   1 -17   7
 -3  -7  -4 -10 -13   2   3   2   3   8
-13  -1   5   6   1   3   5  -2  10   9
  4   7 -16 -10   3  12   1   4  10   3
 10  -8   1  -2  -4  -7   0 -18   5  -4
 -1  -2 -10  -3 -16   4   1   2   9  11

Execution time is 11 ms
```

Варіант виконання завдання №1.



Розмір матриці	К-ть ітерацій (завдання №1)	Час виконання
10	20	533334 нс
50	350	381334 нс
100	1325	1076889 нс
500	31625	1737778 нс

Завдання №1.

1. В кожному рядку знайти максимальний елемент. Поміняти місцями рядки з найбільшим і найменшим максимальними елементами.

2. В кожному стовпчику знайти мінімальний елемент. Поміняти місцями стовпчики з найбільшим і найменшим мінімальними елементами.

3. Обнулити рядки, в яких від'ємних елементів більше ніж додатних.

4. Всі елементи стовпчиків, які знаходяться між мінімальним і максимальним елементами замінити на 3.

5. У кожному стовпчику підрахувати суму елементів між першим та останнім від'ємним елементом.

6. Рядки, у яких від'ємних елементів більше ніж додатних, перевернути у зворотному порядку.

7. В рядках, сума елементів яких менше ніж сума елементів 1-го стовпчика, кожен елемент збільшити на 5.

8. Попарно поміняти місцями стовпчики, які розміщені зліва від стовпчика з максимальним елементом.

9. Якщо у рядку сума елементів менша за суму елементів останнього стовпчика, перевернути цей рядок у зворотному порядку.

10. Поміняти місцями рядки з найменшою та найбільшою сумою.

Завдання №1.

11. Поміняти місцями стовпчики з найменшою та найбільшою сумою елементів, які більше 20.

12. Знайти стовпчик з найбільшою послідовністю елементів, впорядкованих за зростанням.

13. Знайти рядок з найбільшою послідовністю елементів, впорядкованих за спаданням.

14. Поміняти місцями стовпчики з найменшою та найбільшою послідовністю елементів.

15. Знайти рядок, у якому найменше нульових елементів. Поміняти його зі стовпчиком, у якого найбільше нульових елементів.

16. Знайти мінімальний елемент масиву. Поміняти місцями рядок та стовпчик, в яких знаходиться мінімальний елемент.

17. Знайти максимальний елемент масиву. Порівняти, кількість додатніх та від’ємних елементів,які знаходяться у правій та лівій прямокутній частині масиву, відносно стовпчика з максимальним елементом.

18. Знайти стовпчик з найбільшою послідовністю парних чисел.

19. В кожному рядку масиву всі від’ємні елементи перенести в ліву частину

Завдання №1.

20. В кожному рядку масиву всі від’ємні елементи перенести в ліву частину

21. В кожному стовпчику масиву всі парні елементи перенести наверх.

22. В кожному рядку масиву всі нульові елементи розмістити в лівій частині

23. Розвернути кожен стовпчик масиву у зворотному порядку.

24. Поставити на головну діагональ матриці мінімальний елемент стовпчика.

25. Поставити на побічну діагональ матриці максимальний елемент рядочка.

26. В кожному стовпчику знайти максимальний елемент. Поміняти місцями стовпчики з найбільшим і найменшим максимальними елементами.

27. Поміняти місцями рядки з найменшою та найбільшою послідовністю елементів, які дорівнюють 8.

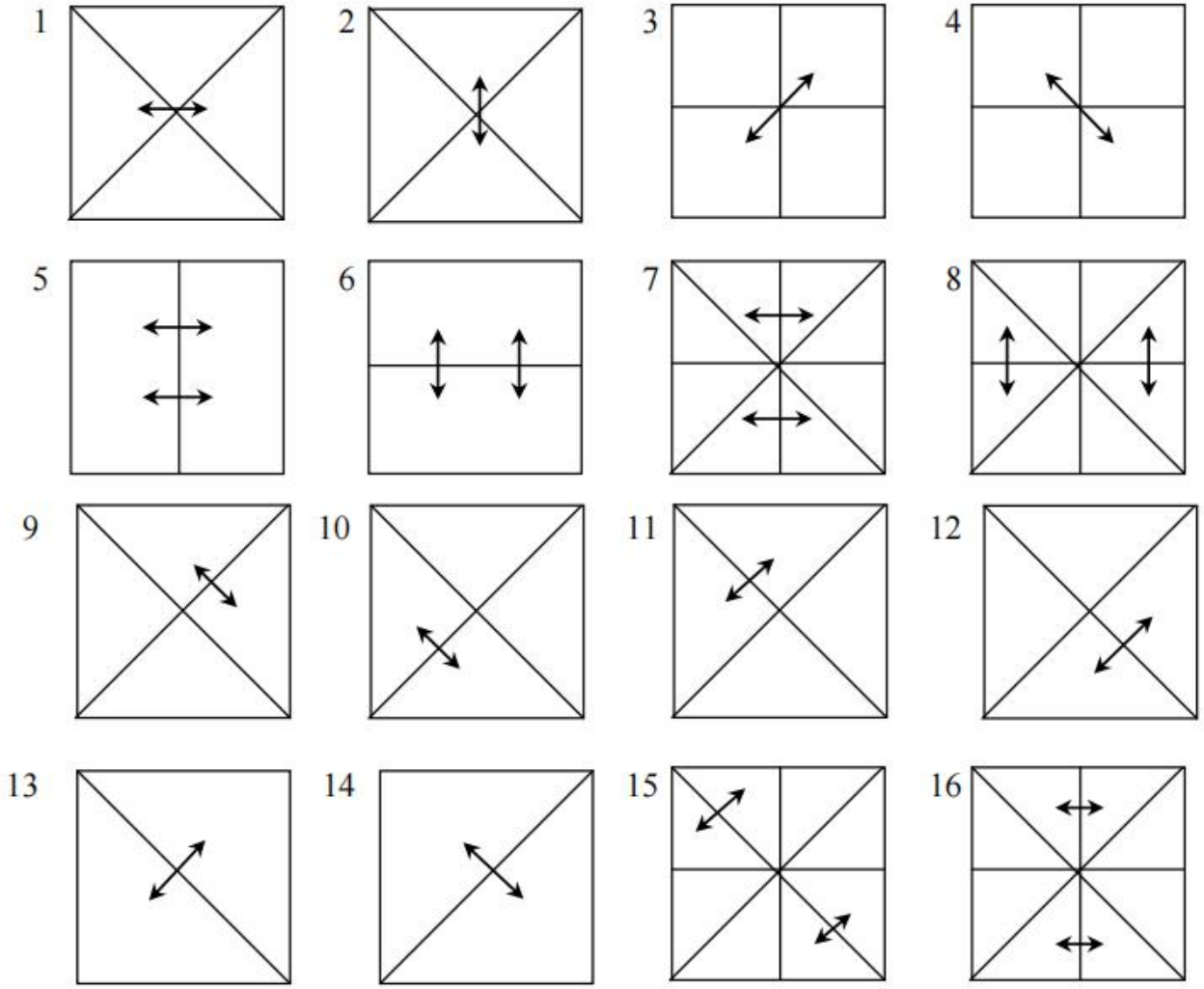
28. Знайти максимальний елемент масиву. Поміняти місцями рядок та стовпчик, в яких знаходиться максимальний елемент.

29. Поміняти місцями стовпчики з найменшою та найбільшою сумою.

30. Попарно поміняти місцями рядки, які розміщені зверху від рядка з максимальним елементом.

Завдання №2.

1-16. Провести обмін елементів матриці за вказаною схемою, любим обраним методом



Завдання №2.

17. Змістити вправо кожен рядок масиву елементи масиву на 5 елементів.

18. Видалити в кожному рядку масиву всі елементи, які знаходяться перед мінімальним елементом, змістивши рядок вліво.

19. Видалити в кожному стовпчику нульові елементи, змістивши на їх місце наступні.

20. В кожному стовпчику кожен третій елемент знищити, змістивши на його місце нижні елементи.

21. Провести циклічний зсув рядочків масиву на 3 рядки вгору.

22. Провести циклічний зсув стовпчиків масиву з ліва на право на 1 стовпчик. Останній стовпчик поставити на місце першого.

23. Провести циклічний зсув рядочків масиву на 2 рядки вниз.

24. Якщо в рядку від'ємних елементів більше ніж додатних, циклічно зсунути елементи рядка вліво на 1 елемент.

25. Видалити з масиву всі елементи, які знаходяться перед мінімальним елементом, змістивши масив вліво.

Завдання №2.

26. Провести циклічний зсув кожного рядочка масиву за таким правилом:

кожен рядочок з парним номером змістити на 2 елементи вліво,

а з непарним номером на 2 елементи вправо.

27. Циклічно змістити рядочки масиву, що стоять на парних місцях.

28. Кожен другий рядок, зсунути вниз на 3 елементи.

29. Кожен другий стовпчик змістити праворуч на 1 елемент.

30. Провести циклічний зсув рядочків масиву на 2 рядки вниз.

```
Enter the length of the matrix: 10
30  39  18  22  38  34  47  42  47  47
18  13  29  29  13  16  27  15  23  45
27  27  22  25  32  30  19  44  16  43
11  24  23  25  27  31  46  20  48  45
31  13  37  25  41  46  30  46  43  30
13  47  40  45  41  35  36  12  24  23
37  42  19  25  39  39  44  15  29  33
35  12  48  44  37  41  38  19  25  23
22  27  42  24  39  34  22  24  40  21
33  40  42  12  14  27  37  42  31  21

Changed matrix:
33  39  18  22  38  34  47  42  47  21
22  27  29  29  13  16  27  15  40  21
35  12  48  25  32  30  19  19  25  23
37  42  19  25  27  31  44  15  29  33
13  47  40  45  41  35  36  12  24  23
31  13  37  25  41  46  30  46  43  30
11  24  23  25  39  39  46  20  48  45
27  27  22  44  37  41  38  44  16  43
18  13  42  24  39  34  22  24  23  45
30  40  42  12  14  27  37  42  31  47
Iterations: 210
Time: 190
```

Дисципліна: “Програмування складних алгоритмів”

Лабораторна робота №1. СКЛАДНІСТЬ АЛГОРИТМУ.

В умовах дистанційного навчання для виконання лабораторних робіт пропонується використання On-line середовище Repl.it