

Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Навчально-науковий інститут комп’ютерних систем
Кафедра інформаційних систем

Лабораторна робота № 1
з дисципліни: «Теорія алгоритмів»
Тема: «Квадратичні алгоритми сортування»

Варіант № 6

Виконав:

Студент групи АІ-243

Гаврилов О.В.

Перевірили:

Смік С. Ю.

Арсірій О.О.

Одеса 2025

Мета роботи: Метою виконання лабораторної роботи є набуття практичних навичок із проектування, реалізації, тестування та аналізу алгоритмів сортування вибором та вставкою. За рахунок виконання трасування та тестування студент має практично довести квадратичну часову складність та стійкість вивчених алгоритмів, а також порівняти ці алгоритми за кількістю базових операцій.

Завдання:

1.1 Спроектувати, реалізувати, протестувати та проаналізувати алгоритм сортування вибором.

1.2 Спроектувати, реалізувати, протестувати та проаналізувати алгоритм сортування вставками.

1.3 Порівняти обчислювальну складність алгоритмів сортування вибором та вставками за базовими операціями.

Номер варіанту: 6;

Вхідні дані (послідовність): 58, 5, 50, 99, 61, 32, 27, 45, 75.

Результати виконання завдання:

1. Спроектувати, реалізувати, протестувати та проаналізувати алгоритм сортування вибором.

Лістинг 1.1 – Псевдокод алгоритму сортування вибором.

Допоміжна змінна масиву ($A[0], \dots, A[n-1]$)

// Вихідні дані: масив $A [0, \dots, n-1]$ елементів, що впорядковуються у зростаючому порядку

// Початковий масив: $A = [58, 5, 50, 99, 61, 32, 27, 45, 75]$

```
for i ← 0 to n-2 do // зовнішній цикл
{
    min ← i

    for j ← i+1 to n-1 do // внутрішній цикл
        if A[j] < A[min]
            min ← j
    B ← A[i] // Обмін A[i] та A[min]
    A[i] ← A[min]
    A[min] ← B
}
```

Лістинг 1.2 – Python-код алгоритму сортування вибором.

```
def selection_sort(arr):
    n = len(arr)
    # Ініціалізація лічильників
    comparisons = 0
    assignments = 0
```

```

# # Зовнішній цикл ітерується по всьому списку
# # від 0 до n-2 проходить
for i in range(n - 1):

    # # Припускаємо, що поточний елемент є мінімальним
    min_index = i
    assignments += 1 # Присвоєння змінній min_index

    # # Внутрішній цикл шукає найменший елемент в решті списку
    for j in range(i + 1, n):
        comparisons += 1 # Операція порівняння arr[j] < arr[min_index]
        if arr[j] < arr[min_index]:
            min_index = j
            assignments += 1 # Присвоєння змінній min_index

    # # Обмін елементів, якщо знайдено новий мінімальний
    comparisons += 1 # Операція порівняння min_index != i
(xоча в чистому коді може бути неявним, тут рахуємо для логіки)
    if min_index != i:
        # # Обмін елементів
        arr[i], arr[min_index] = arr[min_index], arr[i]
        assignments += 3 # Три присвоєння при обміні (темп,
arr[i], arr[min_index])

return arr, comparisons, assignments

my_list = [58, 5, 50, 99, 61, 32, 27, 45, 75]

# Виконання сортування та отримання лічильників
# Використання .copy(), щоб не змінювати оригінал
sorted_list, comps, assigs = selection_sort(my_list.copy())

print(f"Оригінальний список: {my_list}")
print(f"Відсортований список: {sorted_list}")
print(f"Кількість порівнянь: {comps}")
print(f"Кількість присвоєнь: {assigs}")

```

Результати виконання Python-коду показано далі:

- Оригінальний список: [58, 5, 50, 99, 61, 32, 27, 45, 75]
- Відсортований список: [5, 27, 32, 45, 50, 58, 61, 75, 99]
- Кількість порівнянь: 44
- Кількість присвоєнь: 40

Таблиця 1 Результати чисельного моделювання та трасування алгоритму сортування вибором.

Чисельне моделювання псевдокоду Лістінг 1.1	Трасування Python-коду Лістінг 1.2
<p>Вхідні дані: $A = [58_0, 5_1, 50_2, 99_3, 61_4, 32_5, 27_6, 45_7, 75_8]$ $n = 9$</p> <p>(for i)</p> <ol style="list-style-type: none"> 1. $i=0; min=0; j=0+1=1; A[1]<A[0] (5<58) min=1$ 1. $j=1 A[1]<A[0] (5<58) min=1$ 2. $j=2 A[2]<A[1] (50<5) min=1$ 3. $j=3 A[3]<A[1] (99<5) min=1$ 4. $j=4 A[4]<A[1] (61<5) min=1$ 5. $j=5 A[5]<A[1] (32<5) min=1$ 6. $j=6 A[6]<A[1] (27<5) min=1$ 7. $j=7 A[7]<A[1] (45<5) min=1$ 8. $j=8 A[8]<A[1] (75<5) min=1$ <p>(end for j)</p> <p>b=A[0] A[0]=A[1] A[1]=B A[0]=5 A[1]=58</p> <p>$A=[5_0, 58_1, 50_2, 99_3, 61_4, 32_5, 27_6, 45_7, 75_8]$</p> <p>2. $i=1 min=1; j=1+1=2; A[2]<A[1] (50<58) \rightarrow min=2$</p> <ol style="list-style-type: none"> 1. $j=2 A[2]<A[1] (50<58) min=2$ 2. $j=3 A[3]<A[2] (99<50) min=2$ 3. $j=4 A[4]<A[2] (61<50) min=2$ 4. $j=5 A[5]<A[2] (32<50) min=5$ 5. $j=6 A[6]<A[5] (27<32) min=6$ 6. $j=7 A[7]<A[6] (45<27) min=6$ 7. $j=8 A[8]<A[6] (75<27) min=6$ <p>(end for j)</p> <p>b=A[1] A[1]=A[6] A[6]=B A[1]=27 A[6]=58</p> <p>$A=[5_0, 27_1, 50_2, 99_3, 61_4, 32_5, 58_6, 45_7, 75_8]$</p> <p>3. $i=2; min=2; j=2+1=3; A[3]<A[2] (99<50) \rightarrow min=2$</p> <ol style="list-style-type: none"> 1. $j=3 A[3]<A[2] (99<50) min=2$ 2. $j=4 A[4]<A[2] (61<50) min=2$ 3. $j=5 A[5]<A[2] (32<50) min=5$ 4. $j=6 A[6]<A[5] (58<32) min=5$ 5. $j=7 A[7]<A[5] (45<32) min=5$ 6. $j=8 A[8]<A[5] (75<32) min=5$ <p>(end for j)</p>	<p>Початковий масив: $[58, 5, 50, 99, 61, 32, 27, 45, 75]$</p> <hr/> <p>Ітерація 0: ($i=0$)</p> <p>Поточний елемент (для обміну): $arr[0] = 58$ Шукаємо мінімальний елемент у частині: $[58, 5, 50, 99, 61, 32, 27, 45, 75]$</p> <p>Знайдено мінімальний елемент: $arr[1] = 5$</p> <p>Обмін $arr[0]$ (58) і $arr[1]$ (5)</p> <p>Масив після ітерації 0: $[5, 58, 50, 99, 61, 32, 27, 45, 75]$</p> <hr/> <p>Ітерація 1: ($i=1$)</p> <p>Поточний елемент (для обміну): $arr[1] = 58$ Шукаємо мінімальний елемент у частині: $[58, 50, 99, 61, 32, 27, 45, 75]$</p> <p>Знайдено мінімальний елемент: $arr[6] = 27$</p> <p>Обмін $arr[1]$ (58) і $arr[6]$ (27)</p> <p>Масив після ітерації 1: $[5, 27, 50, 99, 61, 32, 58, 45, 75]$</p> <hr/> <p>Ітерація 2: ($i=2$)</p> <p>Поточний елемент (для обміну): $arr[2] = 50$ Шукаємо мінімальний елемент у частині: $[50, 99, 61, 32, 58, 45, 75]$</p> <p>Знайдено мінімальний елемент: $arr[5] = 32$</p> <p>Обмін $arr[2]$ (50) і $arr[5]$ (32)</p> <p>Масив після ітерації 2: $[5, 27, 32, 99, 61, 50, 58, 45, 75]$</p> <hr/> <p>Ітерація 3: ($i=3$)</p> <p>Поточний елемент (для обміну): $arr[3] = 99$ Шукаємо мінімальний елемент у частині: $[99, 61, 50, 58, 45, 75]$</p> <p>Знайдено мінімальний елемент: $arr[7] = 45$</p> <p>Обмін $arr[3]$ (99) і $arr[7]$ (45) Масив після ітерації 3: $[5, 27, 32, 45, 61, 50, 58, 99, 75]$</p> <hr/> <p>Ітерація 4: ($i=4$)</p> <p>Поточний елемент (для обміну): $arr[4] = 61$ Шукаємо мінімальний елемент у частині: $[61, 50, 58, 99, 75]$</p>

b=A[2] A[2]=A[5] A[5]=B A[2]=32 A[5]=50

A=[5₀, 27₁, 32₂, 99₃, 61₄, 50₅, 58₆, 45₇, 75₈]

4. i=3; min=3; j=3+1=4; A[4]<A[3] (61<99) → min=4

1. j=4 A[4]<A[3] (61<99) min=4

2. j=5 A[5]<A[4] (50<61) min=5

3. j=6 A[6]<A[5] (58<50) min=5

4. j=7 A[7]<A[5] (45<50) min=7

5. j=8 A[8]<A[7] (75<45) min=7

(end for j)

b=A[3] A[3]=A[7] A[7]=B A[3]=45 A[7]=99

A=[5_0, 27_1, 32_2, 45_3, 61_4, 50_5, 58_6, 99_7, 75_8]

5. i=4; min=4; j=4+1=5; A[5]<A[4] (50<61) → min=5

1. j=5 A[5]<A[4] (50<61) min=5

2. j=6 A[6]<A[5] (58<50) min=5

3. j=7 A[7]<A[5] (99<50) min=5

4. j=8 A[8]<A[5] (75<50) min=5

(end for j)

b=A[4] A[4]=A[5] A[5]=B A[4]=50 A[5]=61

A=[5_0, 27_1, 32_2, 45_3, 50_4, 61_5, 58_6, 99_7, 75_8]

6. i=5; min=5; j=5+1=6; A[6]<A[5] (58<61) → min=6

1. j=6 A[6]<A[5] (58<61) min=6

2. j=7 A[7]<A[6] (99<58) min=6

3. j=8 A[8]<A[6] (75<58) min=6

(end for j)

b=A[5] A[5]=A[6] A[6]=B A[5]=58 A[6]=61

A=[5_0, 27_1, 32_2, 45_3, 50_4, 58_5, 61_6, 99_7, 75_8]

7. i=6; min=6; j=6+1=7; A[7]<A[6] (99<61) → min=6

1. j=7 A[7]<A[6] (99<61) min=6

2. j=8 A[8]<A[6] (75<61) min=6

(end for j)

b=A[6] A[6]=A[6] A[6]=B A[6]=61 A[6]=61

A=[5₀, 27₁, 32₂, 45₃, 50₄, 58₅, 61₆, 99₇, 75₈]

8. i=7 min=7; j=7+1=8;

Знайдено мінімальний елемент: arr[5] = 50 Обмін arr[4] (61) і arr[5] (50)

Масив після ітерації 4: [5, 27, 32, 45, 50, 61, 58, 99, 75]

Ітерація 5: (i=5)

Поточний елемент (для обміну): arr[5] = 61 Шукаємо мінімальний елемент у частині:

[61, 58, 99, 75]

Знайдено мінімальний елемент: arr[6] = 58

Обмін arr[5] (61) і arr[6] (58)

Масив після ітерації 5: [5, 27, 32, 45, 50, 58, 61, 99, 75]

Ітерація 6: (i=6)

Поточний елемент (для обміну): arr[6] = 61 Шукаємо мінімальний елемент у частині:

[61, 99, 75]

Знайдено мінімальний елемент: arr[6] = 61

Обмін не відбувається (min_index = 6)

Масив після ітерації 6: [5, 27, 32, 45, 50, 58, 61, 99, 75]

Ітерація 7: (i=7)

Поточний елемент (для обміну): arr[7] = 99 Шукаємо мінімальний елемент у частині:

[99, 75]

Знайдено мінімальний елемент: arr[8] = 75

Обмін arr[7] (99) і arr[8] (75)

Масив після ітерації 7: [5, 27, 32, 45, 50, 58, 61, 75, 99]

Сортування завершено.

Фінальний відсортований масив: \$[5, 27, 32, 45, 50, 58, 61, 75, 99]\$

Загальна кількість порівнянь: 44

Загальна кількість присвоєнь: 40

```

1.           j=8 A[8]<A[7] (75<99) min=8
(end for j)

b=A[7] A[7]=A[8] A[8]=B A[7]=75 A[8]=99
A=[50, 271, 322, 453, 504, 585, 616, 757, 998]

(end for i)

```

2. Спроектувати, реалізувати, протестувати та проаналізувати алгоритм сортування вставками.

Лістинг 1.3 - Псевдокод алгоритму сортування вставками.

```

Insertion_Sort
for j = 2 to A.length do
    key = A[j]
    i = j - 1
    while (int i > 0 and A[i] > key) do
        A[i + 1] = A[58, 5, 50, 99, 61, 32, 27, 45, 75].i]
        i = i - 1
    end while
    A[i+1] = key
end

```

Лістинг 1.4 - Python-код алгоритму сортування вставками

```

def insertion_sort(arr):
    n = len(arr)
    comparisons = 0
    assignments = 0

    # Цикл ітерується від другого елемента до кінця
    # i - це індекс елемента, який потрібно вставити
    for i in range(1, n):
        key
        # Зберігаємо поточний елемент для вставки
        key = arr[i]
        assignments += 1

        # j - індекс попереднього елемента
        j = i - 1
        assignments += 1

        # Пересуваємо елементи, що більші за key,
        # вправо, щоб звільнити місце для вставки
        while j >= 0 and arr[j] > key:

            # Порівняння в умові while (arr[j] > key)
            comparisons += 1

            arr[j + 1] = arr[j]
            assignments += 1 # Присвоєння елементу масиву

        j = j - 1

```

```

assignments += 1 # Присвоєння змінній j

# Додаткове порівняння, коли умова while стає false
# Якщо цикл зупинився через arr[j] <= key
if j >= 0:
    comparisons += 1

    # Вставляємо key на його правильне місце
    arr[j + 1] = key
    assignments += 1

return arr, comparisons, assignments

```

`my_list = [58, 5, 50, 99, 61, 32, 27, 45, 75]`

`# Використання .copy(), щоб не змінювати оригінал`

`sorted_list, comps, assigs = insertion_sort(my_list.copy())`

`print(f"Оригінальний список: {my_list}")`

`print(f"Відсортований список: {sorted_list}")`

`print(f"Кількість порівнянь: {comps}")`

`print(f"Кількість присвоєнь: {assigs}")`

Результати виконання Python-коду показано далі:

- Оригінальний список: [58, 5, 50, 99, 61, 32, 27, 45, 75]
- Відсортований список: [5, 27, 32, 45, 50, 58, 61, 75, 99]
- Кількість порівнянь: 24
- Кількість присвоєнь: 58

Таблиця 2 Результати чисельного моделювання та трасування алгоритму сортування вставками.

Чисельне моделювання псевдокоду Лістінг 1.3	Трасування Python-коду Лістінг 2 1.4
<p>Вхідні дані: A = [58₀, 5₁, 50₂, 99₃, 61₄, 32₅, 27₆, 45₇, 75₈] n = 9</p> <p>A.length=9 (for j)</p> <p>j = 2; key = A[2] = 5; i = 2 - 1 = 1; i > 0 (True) A[1] > key (58 > 5 - True) → A[2] = A[1] = 58;</p> <p>A=[58₁, 58₂, 50₃, 99₄, 61₅, 32₆, 27₇, 45₈, 75₉]</p> <p>i = 1 - 1 = 0;</p> <p>i > 0 (False)</p> <p>A[1] = key = 5</p> <p>A=[5₁, 58₂, 50₃, 99₄, 61₅, 32₆, 27₇, 45₈, 75₉]</p> <p>j = 3; key = A[3] = 50; i = 3 - 1 = 2;</p> <p>i > 0 (True) A[2] > key (58 > 50 - True) → A[3] = A[2] = 58;</p> <p>A=[5₁, 58₂, 58₃, 99₄, 61₅, 32₆, 27₇, 45₈, 75₉]</p>	<p>Початковий масив: [58, 5, 50, 99, 61, 32, 27, 45, 75]</p> <p>-----</p> <p>Ітерація 1: (i=1)</p> <p>Елемент для вставки (key): 5</p> <p>Відсортована частина: [58]</p> <p>Порівняння: 58 > 5. True. Зсуваємо 58 вправо.</p> <p>Порівняння:</p> <p>Досягнуто початку масиву. Цикл завершено. Вставка 5 на позицію 0.</p> <p>Масив після ітерації 1:</p> <p>[5, 58, 50, 99, 61, 32, 27, 45, 75]</p> <p>-----</p> <p>Ітерація 2: (i=2)</p> <p>Елемент для вставки (key): 50</p> <p>Відсортована частина: [5, 58]</p>

i = 2 - 1 = 1;	Порівняння: $58 > 50$.
i > 0 (True) A[1] > key ($5 > 50$ - False)	True. Зсуваємо 58 вправо. Порівняння: $5 > 50$.
A[2] = key = 50	False. Цикл завершено.
A=[5 ₁ , 50 ₂ , 58 ₃ , 99 ₄ , 61 ₅ , 32 ₆ , 27 ₇ , 45 ₈ , 75 ₉]	Вставка 50 на позицію 1.
 	Масив після ітерації 2:
j = 4; key = A[4] = 99; i = 4 - 1 = 3;	[5, 50, 58, 99, 61, 32, 27, 45, 75]
i > 0 (True) A[3] > key ($58 > 99$ — False)	-----
A[4] = key = 99	Ітерація 3: (i=3)
A=[5 ₁ , 50 ₂ , 58 ₃ , 99 ₄ , 61 ₅ , 32 ₆ , 27 ₇ , 45 ₈ , 75 ₉]	Елемент для вставки (key): 99
 	Відсортована частина: [5, 50, 58]
j = 5; key = A[5] = 61; i = 5 - 1 = 4;	Порівняння: $58 > 99$. False.
i > 0 (True) A[4] > key ($99 > 61$ - True) → A[5] = A[4] = 99;	Цикл завершено. Вставка 99 на позицію 3.
A=[5 ₁ , 50 ₂ , 58 ₃ , 99 ₄ , 99 ₅ , 32 ₆ , 27 ₇ , 45 ₈ , 75 ₉]	Масив після ітерації 3: [5, 50, 58, 99, 61, 32, 27, 45, 75]
i = 4 - 1 = 3;	-----
i > 0 (True) A[3] > key ($58 > 61$ - False)	Ітерація 4: (i=4)
A[4] = key = 61	Елемент для вставки (key): 61
A=[5 ₁ , 50 ₂ , 58 ₃ , 61 ₄ , 99 ₅ , 32 ₆ , 27 ₇ , 45 ₈ , 75 ₉]	Відсортована частина: [5, 50, 58, 99]
 	Порівняння: $99 > 61$. True. Зсуваємо 99 вправо.
j = 6; key = A[6] = 32; i = 6 - 1 = 5;	Порівняння: $58 > 61$. False.
i > 0 (True) A[5] > key ($99 > 32$ - True) → A[6] = A[5] = 99;	Цикл завершено. Вставка 61 на позицію 3.
A=[5 ₁ , 50 ₂ , 58 ₃ , 61 ₄ , 99 ₅ , 99 ₆ , 27 ₇ , 45 ₈ , 75 ₉]	Масив після ітерації 4: [5, 50, 58, 61, 99, 32, 27, 45, 75]\$
i = 5 - 1 = 4;	-----
i > 0 (True) A[4] > key ($61 > 32$ - True) → A[5] = A[4] = 61;	Ітерація 5: (i=5)
A=[5 ₁ , 50 ₂ , 58 ₃ , 61 ₄ , 61 ₅ , 99 ₆ , 27 ₇ , 45 ₈ , 75 ₉]	Елемент для вставки (key): 32
i = 4 - 1 = 3; i > 0 (True) A[3] > key ($58 > 32$ - True) →	Відсортована частина: [5, 50, 58, 61, 99] Порівняння: $99 > 32$. True. Зсуваємо 99 вправо. Порівняння: $61 > 32$. True. Зсуваємо 61 вправо. Порівняння: $58 > 32$. True. Зсуваємо 58 вправо. Порівняння: $50 > 32$. True. Зсуваємо 50 вправо. Порівняння: $5 > 32$. False.
A[4] = A[3] = 58;	Цикл завершено.
A=[5 ₁ , 50 ₂ , 58 ₃ , 58 ₄ , 61 ₅ , 99 ₆ , 27 ₇ , 45 ₈ , 75 ₉]	Вставка 32 на позицію 1. Масив після ітерації 5: [5, 32, 50,
i = 3 - 1 = 2;	58, 61, 99, 27, 45, 75]
i > 0 (True) A[2] > key ($50 > 32$ - True) → A[3] = A[2] = 50;	-----
A=[5 ₁ , 50 ₂ , 50 ₃ , 58 ₄ , 61 ₅ , 99 ₆ , 27 ₇ , 45 ₈ , 75 ₉]	Ітерація 6: (i=6)
i = 2 - 1 = 1;	Елемент для вставки (key): 27
i > 0 (True) A[1] > key ($5 > 32$ - False)	Відсортована частина: [5, 32, 50, 58, 61, 99] Порівняння: $99 > 27$. True. Зсуваємо 99 вправо. Порівняння: $61 > 27$. True. Зсуваємо 61 вправо. Порівняння: $58 > 27$. True. Зсуваємо 58 вправо. Порівняння: $50 > 27$. True. Зсуваємо 50 вправо. Порівняння: $32 > 27$. True. Зсуваємо 32 вправо. Порівняння: $5 > 27$. False.
A[2] = key = 32;	Цикл завершено. Вставка 27 на позицію 1.
A=[5 ₁ , 32 ₂ , 50 ₃ , 58 ₄ , 61 ₅ , 99 ₆ , 27 ₇ , 45 ₈ , 75 ₉]	Масив після ітерації 6: [5, 27, 32, 50, 58, 61, 99, 45, 75]

j = 7; key = A[7] = 27; i = 7 - 1 = 6;	Ітерація 7: (i=7)
i > 0 (True) A[6] > key ($99 > 27$ - True) → A[7] = A[6] = 99;	Елемент для вставки (key): 45
A=[5 ₁ , 32 ₂ , 50 ₃ , 58 ₄ , 61 ₅ , 99 ₆ , 99 ₇ , 45 ₈ , 75 ₉]	Відсортована частина: [5, 27, 32, 50, 58, 61, 99]
	Порівняння: $99 > 45$. True. Зсуваємо 99 вправо.

i = 6 - 1 = 5;
 i > 0 (True) A[5] > key (61 > 27 - True) → A[6] = A[5] = 61;
 A=[5₁, 32₂, 50₃, 58₄, 61₅, 61₆, 99₇, 45₈, 75₉]
 i = 5 - 1 = 4;
 i > 0 (True) A[4] > key (58 > 27 - True) → A[5] = A[4] = 58;
 A=[5₁, 32₂, 50₃, 58₄, 58₅, 61₆, 99₇, 45₈, 75₉]
 i = 4 - 1 = 3;
 i > 0 (True) A[3] > key (50 > 27 - True) → A[4] = A[3] = 50;
 A=[5₁, 32₂, 50₃, 50₄, 58₅, 61₆, 99₇, 45₈, 75₉]
 i = 3 - 1 = 2;
 i > 0 (True) A[2] > key (32 > 27 - True) → A[3] = A[2] = 32;
 A=[5₁, 32₂, 32₃, 50₄, 58₅, 61₆, 99₇, 45₈, 75₉]
 i = 2 - 1 = 1;
 i > 0 (True) A[1] > key (5 > 27 - False)
 A[2] = key = 27;
 A=[5₁, 27₂, 32₃, 50₄, 58₅, 61₆, 99₇, 45₈, 75₉]

j = 8; key = A[8] = 45; i = 8 - 1 = 7;
 i > 0 (True) A[7] > key (99 > 45 - True) → A[8] = A[7] = 99;
 A=[5₁, 27₂, 32₃, 50₄, 58₅, 61₆, 99₇, 99₈, 75₉]
 i = 7 - 1 = 6;
 i > 0 (True) A[6] > key (61 > 45 - True) → A[7] = A[6] = 61;
 A=[5₁, 27₂, 32₃, 50₄, 58₅, 61₆, 61₇, 99₈, 75₉]
 i = 6 - 1 = 5;
 i > 0 (True) A[5] > key (58 > 45 - True) → A[6] = A[5] = 58;
 A=[5₁, 27₂, 32₃, 50₄, 58₅, 58₆, 61₇, 99₈, 75₉]
 i = 5 - 1 = 4;
 i > 0 (True) A[4] > key (50 > 45 - True) → A[5] = A[4] = 50;
 A=[5₁, 27₂, 32₃, 50₄, 50₅, 58₆, 61₇, 99₈, 75₉]
 i = 4 - 1 = 3;
 i > 0 (True) A[3] > key (32 > 45 - False)
 A[4] = key = 45;
 A=[5₁, 27₂, 32₃, 45₄, 50₅, 58₆, 61₇, 99₈, 75₉]

j = 9; key = A[9] = 75; i = 9 - 1 = 8;
 i > 0 (True) A[8] > key (99 > 75 - True) → A[9] = A[8] = 99;
 A=[5₁, 27₂, 32₃, 45₄, 50₅, 58₆, 61₇, 99₈, 99₉]

Порівняння: 61 > 45. True. Зсуваємо 61 вправо.
 Порівняння: 58 > 45. True. Зсуваємо 58 вправо.
 Порівняння: 50 > 45. True. Зсуваємо 50 вправо.
 Порівняння: 32 > 45. False.

Цикл завершено. Вставка 45 на позицію 3.

Масив після ітерації 7: [5, 27, 32, 45, 50, 58, 61, 99, 75]

Ітерація 8: (i=8)

Елемент для вставки (key): 75

Відсортована частина: [5, 27, 32, 45, 50, 58, 61, 99]
 Порівняння: 99 > 75. True. Зсуваємо 99 вправо.
 Порівняння: 61 > 75. False.

Цикл завершено. Вставка 75 на позицію 7.

Масив після ітерації 8:

[5, 27, 32, 45, 50, 58, 61, 75, 99]

Сортування завершено.

Фінальний відсортований масив: \$[5, 27, 32, 45, 50, 58, 61, 75, 99]\$

Загальна кількість порівнянь: 24

Загальна кількість присвоєнь: 58

i = 8 - 1 = 7; i > 0 (True) A[7] > key (61 > 75 - False) A[8] = key = 75; A=[5 ₁ , 27 ₂ , 32 ₃ , 45 ₄ , 50 ₅ , 58 ₆ , 61 ₇ , 75 ₈ , 99 ₉] (End for j)	
--	--

3. Порівняти обчислювальну складність алгоритмів сортування вибором та вставками за базовими операціями.

Таблиця 3. Порівняння обчислювальної складності алгоритмів сортування вибором та вставками.

Назва алгоритму	Теоретичні відомості		Результати експерименту n=9	
	порівняння	присвоювання	порівняння	присвоювання
Selection Sort	$O(n^2)$,	$O(n)$	44	40
Insertion Sort	$O(n)$ (Best) $O(n^2)$ (Worst, Average)	$O(n)$ (Best) $O(n^2)$ (Worst, Average)	24	58

4. Яким чином зміняться наведені в п. 3 результати для відсортованих послідовностей в прямому та зворотному порядку.

Таблиця 4. для випадку відсортованої послідовності в прямому порядку:

Назва алгоритму	Теоретичні відомості		Результати експерименту n=9	
	порівняння	присвоювання	порівняння	присвоювання
Selection Sort	$O(n^2)$,	$O(n)$	Без змін (залишиться 44)	Без змін/Мінімум (залишиться 40)
Insertion Sort	$O(n)$ (Best)	$O(n)$ (Best)	Значне зростання (з 24 до 36)	Значне зростання (з 58 до $O(N^2)$)

Висновки: Виконання лабораторної роботи №1 дало можливість глибоко зрозуміти та практично закріпiti принципи роботи класичних алгоритмів сортування: **сортування вибором (Selection Sort)** та **сортування вставками (Insertion Sort)**. У процесі роботи було успішно спроектовано, реалізовано та протестовано обидва алгоритми на мові Python, використовуючи заданий вхідний масив A=[58, 5, 50, 99, 61, 32, 27, 45, 75] (N=9). Результатом роботи стало успішне впорядкування масиву у зростаючому порядку: [5, 27, 32, 45, 50, 58, 61, 75, 99].

Особливо цінним етапом став детальний аналіз ефективності алгоритмів через чисельне моделювання (трасування) та підрахунок ключових операцій. На основі

даних **Таблиці 3** для $N=9$ було встановлено, що **сортування вставками (Insertion Sort)** виявилося значно економнішим за порівняннями **(24)** порівняно з сортуванням вибором **(44)**. Однак, Selection Sort продемонстрував меншу кількість присвоювань **(40)** порівняно з Insertion Sort **(58)**, підтверджуючи свою теоретичну перевагу у мінімізації записів. Таким чином, мета роботи — опанування методів сортування, їх реалізації та аналізу — була повністю досягнута, що сприяло розвитку навичок програмування та аналітичного мислення при оцінці складності алгоритмів.

Посилання на GitHub:

<https://github.com/AlexKim71/Theory-of-Algorithms/tree/main>