МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ  
«ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

ЛУЦЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ»

Випускова циклова комісія Компʼютерних систем та інформаційних технологій

**Звіт**

**з практичної роботи №3**

з дисципліни “Алгоритми та структури даних”

Виконав: студент **2** курсу, групи **22-ICT**

Синиця Назар

Викладач: Володимир Остапюк

Термін здачі: 15-10-2024

Здано:

Луцьк – 2024

Зміст

1.[Теорія та дослідження 3](#_30j0zll)

[Опишіть принцип роботи алгоритму сортування бульбашкою. Як відбувається порівняння та обмін елементів? 3](#_1fob9te)

[Опишіть принцип роботи алгоритму сортування вибором. Чим він відрізняється від сортування бульбашкою? 3](#_3znysh7)

[Яка часова складність алгоритмів сортування бульбашкою та сортування вибором в найгіршому, середньому та найкращому випадках? 4](#_2et92p0)

[В яких випадках доцільно використовувати сортування бульбашкою, а в яких - сортування вибором? Які переваги та недоліки кожного алгоритму? 5](#_tyjcwt)

2.[Практичні завдання 6](#_3dy6vkm)

[Реалізуйте функцію bubble\_sort(array\_to\_sort), яка сортує список array\_to\_sort за допомогою алгоритму сортування бульбашкою. 6](#_1t3h5sf)

[Реалізуйте функцію selection\_sort(array\_to\_sort), яка сортує список array\_to\_sort за допомогою алгоритму сортування вибором. 6](#_4d34og8)

[Напишіть програму, яка зчитує список чисел з клавіатури та сортує його за допомогою обох алгоритмів. Виведіть відсортований список та кількість операцій порівняння та обміну, виконаних кожним алгоритмом. 7](#_2s8eyo1)

[\*[Додатково] Реалізуйте модифікацію алгоритму сортування бульбашкою, яка зупиняється, якщо під час проходу по списку не було зроблено жодного обміну. 8](#_17dp8vu)

[Висновок 10](#_3rdcrjn)

# 1.Теорія та дослідження

## Опишіть принцип роботи алгоритму сортування бульбашкою. Як відбувається порівняння та обмін елементів?

Порівняння сусідніх елементів: Алгоритм послідовно проходить по списку елементів, порівнюючи кожні два сусідні елементи далі якщо елементи стоять не в правильному порядку (наприклад, більший перед меншим), то їх міняють місцями. Цей процес повторюється кілька разів, доки весь список не буде відсортований.

Перший прохід: Алгоритм починає з першого елемента і порівнює його з другим. Якщо другий менший, вони міняються місцями. Потім порівнюються другий і третій елементи, і так далі, до кінця списку. В результаті першого проходу найбільший елемент опиняється в кінці списку.

Наступні проходи: Процес повторюється, але з кожним новим проходом алгоритм проходить на один елемент менше, оскільки найбільші елементи вже зайняли свої місця.

Хоча алгоритм сортування бульбашкою простий для розуміння, він має низьку ефективність для великих масивів даних. Кількість операцій порівняння і обміну зростає квадратично зі збільшенням розміру масиву.

## Опишіть принцип роботи алгоритму сортування вибором. Чим він відрізняється від сортування бульбашкою?

У цьому способі сортування на кожному кроці алгоритм шукає найменший елемент у невідсортованій частині масиву. Цей найменший елемент він ставить на початок масиву, і уже в наступних повторах не бере його до уваги. Цей процес повторюється, поки весь масив не буде відсортований.

На відміну від бульбашкового сортування, де порівнюються сусідні елементи, у сортуванні вибором шукається глобальний найменший елемент, і ставиться на 1 місце уже у відсортованому масиві. У сортувані бульбашкою може бути багато обмінів, навіть якщо масив вже майже відсортований. У сортуванні вибором менша кількість обмінів, оскільки елементи міняються місцями стільки раз, скільки елементів у масиві.

Яка часова складність алгоритмів сортування бульбашкою та сортування вибором в найгіршому, середньому та найкращому випадках?

Сортування бульбашкою:

Найгірший випадок: Коли масив відсортований у зворотному порядку. В такому випадку алгоритм виконує максимальну кількість порівнянь і перестановок, що призводить до квадратичної часової складності: O(n²).

Середній випадок: Для випадково переставленого масиву, як правило, також спостерігається квадратична часова складність: O(n²).

Найкращий випадок: Коли масив вже відсортований, алгоритм виконує лише один прохід, щоб це перевірити. Однак, для визначення, що масив відсортований, алгоритм все одно проходить по всьому масиву хоча б один раз. Тому часова складність у найкращому випадку також дорівнює O(n).

Сортування вибором

Найгірший, середній та найкращий випадок: Незалежно від початкового порядку елементів, алгоритм сортування вибором завжди виконує приблизно однакову кількість операцій. На кожному кроці він знаходить мінімальний елемент у невідсортованій частині масиву. Сума цієї арифметичної прогресії є величиною порядку *n²*. Тому часова складність сортування вибором у всіх випадках дорівнює O(n²).

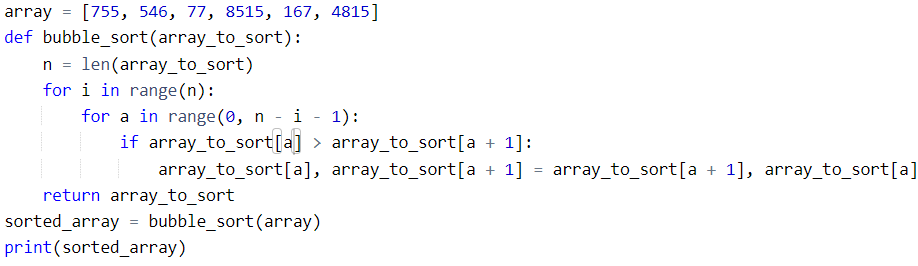
В яких випадках доцільно використовувати сортування бульбашкою, а в яких - сортування вибором? Які переваги та недоліки кожного алгоритму?

Переваги сортування бульбашкою: простий для розуміння та реалізації: Логіка алгоритму інтуїтивно зрозуміла, що робить його хорошим вибором для навчання. Недоліки: має квадратичну часову складність O(n²) навіть у найкращому випадку, що робить його непридатним для великих масивів та багато зайвих порівнянь. Використовувати можна для дуже маленьких масивів або ж у навчанні для вступу вивчення до більших і складніших алгоритмів.

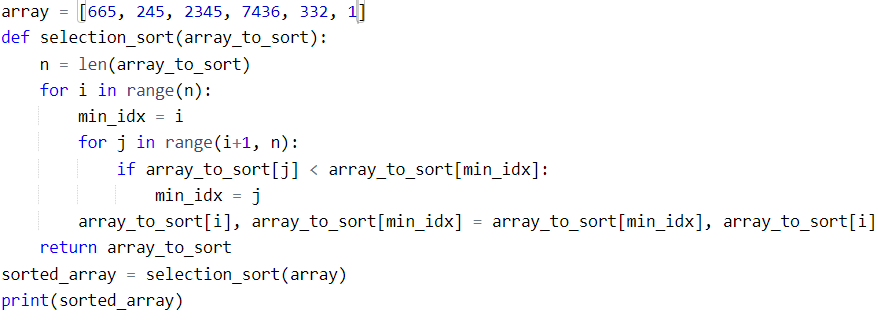
Переваги сортування вибором: він дуже легко зрозуміється. Та порівняно з бульбашковим сортуванням, виконується менша кількість обмінів елементів. Недоліки: як і бульбашкове сортування, має часову складність O(n²), що робить його неефективним для великих масивів. Використовувати можна у навчанні для вступу вивчення до більших і складніших алгоритмів.

# 2.Практичні завдання

## Реалізуйте функцію bubble\_sort(array\_to\_sort), яка сортує список array\_to\_sort за допомогою алгоритму сортування бульбашкою.



Реалізуйте функцію selection\_sort(array\_to\_sort), яка сортує список array\_to\_sort за допомогою алгоритму сортування вибором.



Цикл for i in range(n) проходить через кожний елемент списку;

min\_idx = i встановлює індекс і як індекс найменшого числа;

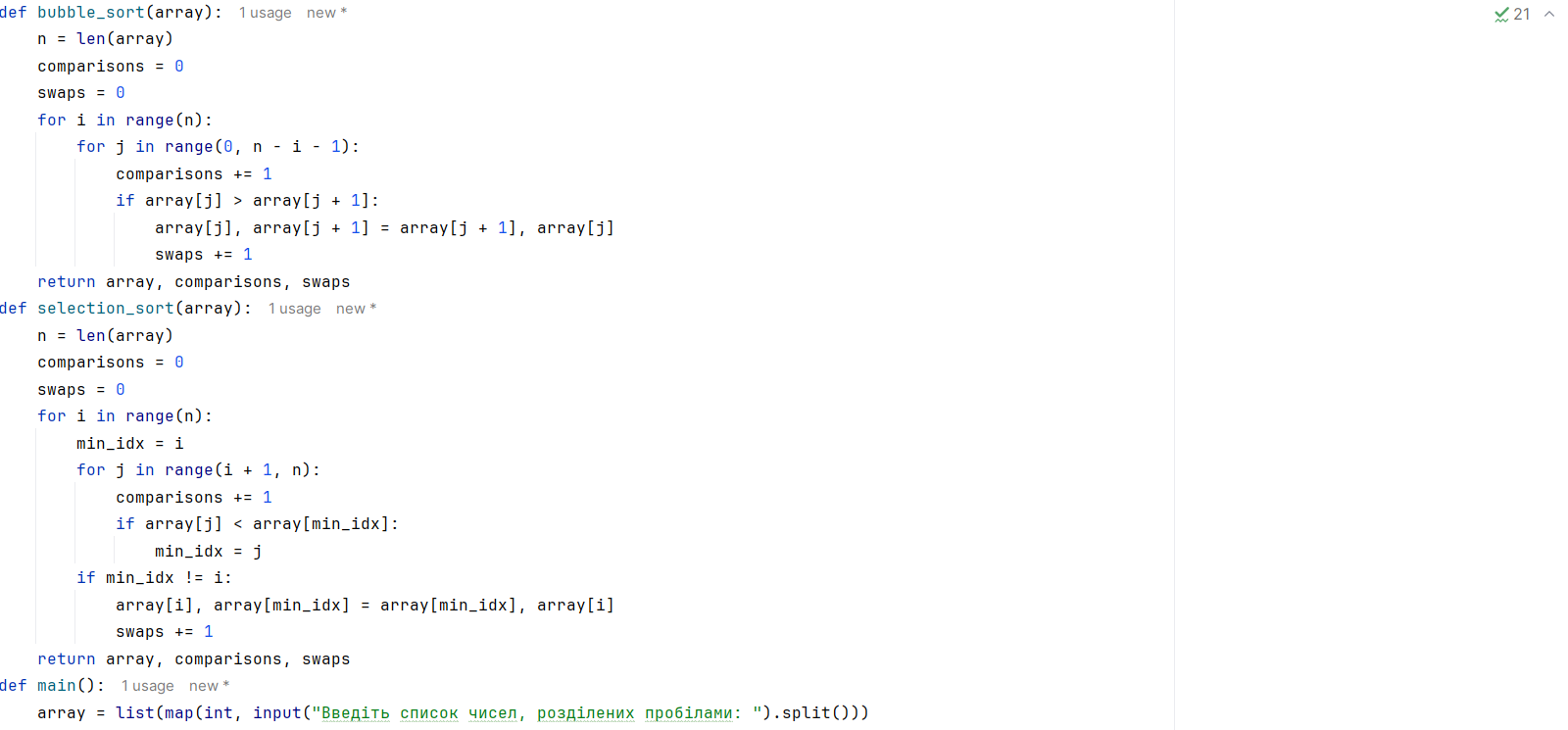
Цикл for j in range(i + 1, n) проходить через елементи після поточного;

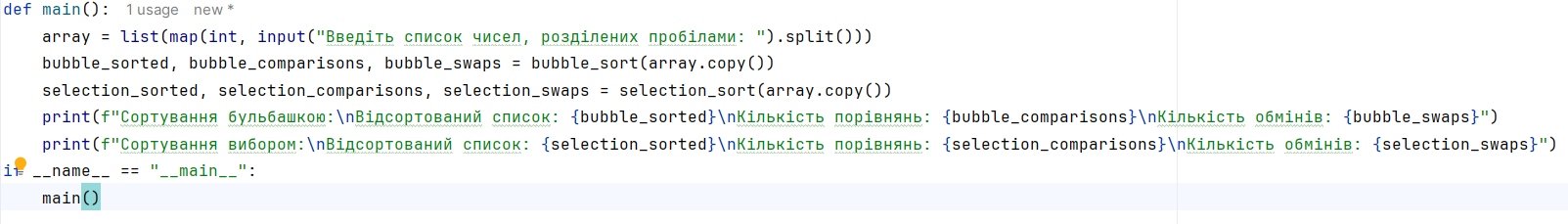
if array\_to\_sort[j] < array\_to\_sort[min\_idx] перевіряє, чи елемент за індексом [j] менший, ніж елемент [min\_idx];

Якщо так, то min\_idx = j оновлює індекс найменшого елемента.

Цикл array\_to\_sort[i], array\_to\_sort[min\_idx] = array\_to\_sort[min\_idx], array\_to\_sort[i] міняє місцями елемент за поточним індексом [j] та найменший елемент, знайдений у внутрішньому циклі.

## Напишіть програму, яка зчитує список чисел з клавіатури та сортує його за допомогою обох алгоритмів. Виведіть відсортований список та кількість операцій порівняння та обміну, виконаних кожним алгоритмом.





Функція bubble\_sort:

Вхідний параметр: array.

Ініціалізація: comparisons і swaps для підрахунку кількості порівнянь і обмінів.

Зовнішній цикл for i in range(n): визначає кількість проходів.

Внутрішній цикл for j in range(0, n - i - 1): для кожного проходу.

comparisons += 1: рахує порівняння.

if array[j] > array[j + 1]: перевірка і обмін елементів.

return: відсортований масив, кількість порівнянь і обмінів.

Функція selection\_sort:

Вхідний параметр: array.

Ініціалізація: comparisons і swaps.

Зовнішній цикл for i in range(n): проходить через кожен елемент.

Внутрішній цикл for j in range(i + 1, n): для пошуку найменшого елемента.

comparisons += 1: рахує порівняння.

if array[j] < array[min\_idx]: знаходить найменший елемент.

Обмін елементами: якщо min\_idx змінився.

Повертає: відсортований масив, кількість порівнянь і обмінів.

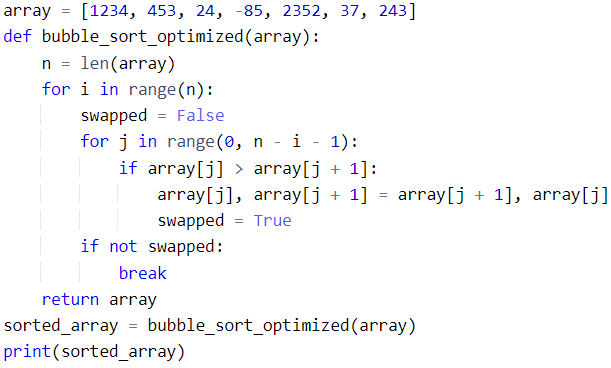
Функція main:

Зчитує список чисел з клавіатури.

Викликає функції сортування, передаючи копії списку.

Виводить відсортовані списки і кількість операцій для обох алгоритмів.

## \*[Додатково] Реалізуйте модифікацію алгоритму сортування бульбашкою, яка зупиняється, якщо під час проходу по списку не було зроблено жодного обміну.

for i in range(n): повторює цикл від початку до кінця масиву.

swapped = False ініціалізує чи відбулися якісь обміни у внутрішньому циклі.

for j in range(0, n - i - 1): повторює цикл від початку до кінця

if array[j] > array[j + 1]: порівнює сусідні елементи.

array[j], array[j + 1] = array[j + 1], array[j]: якщо поточний елемент більший за наступний, вони міняються місцями

swapped = True вказує на те, що відбувся обмін.

if not swapped: перевіряє, чи не відбулися обміни у внутрішньому циклі.

sorted\_array = bubble\_sort\_optimized(array) викликає функцію з даним масивом. print(sorted\_array) друкує відсортований масив.

# Висновок

На цій практичній роботі я знову дізнався про нових 2 види сортування: сортування бульбашкою та вибором.

Сортування бульбашкою працює шляхом обміну елементів між собою місцями, якщо наступний елемент менший за цей, від якого розпочалося сортування масиву.

Сортування вибором одразу ж у всьому масиві шукає найменший елемент, та ставить його на початок уже відсортованого масиву. Далі алгоритм знову шукає найменший елемент без попереднього( який уже відсортовано ). І так знову і знову, допоки масив не буде відсортований.

Під час роботи з практичною стало зрозуміло, що обидва алгоритми сортування бульбашкою та вибором мають квадратичну часову складність. Під час сортування бульбашкою виконується дуже багато обмінів місцями. Сортування вибором, як правило, виконує менше обмінів, але кількість порівнянь приблизно така ж. Для невеликих масивів різниця в ефективності може бути незначною, але для великих масивів сортування вибором працює дещо швидше.