Міністерство освіти науки україни

Кременчуцький національний університет

імені михайла остроградського

Навчально-науковий інститут електричної інженерії   
та інформаційних технологій

Кафедра автоматизації та інформаційних систем

НаВчальна дисципліна  
«**Дослідження алгоритмів сортування**»

Звіт

З лабораторної роботи №3

(Робота з Anaconda, OpenCV та Jupyter Notebook)

Виконав:

студент групи КН-24-1

Левченко Д. В.

Кременчук 2024

#### Кроки виконання завдання:

# Вступ

У даній роботі досліджуються два алгоритми сортування:

**Сортування вставками**

**Сортування бульбашкою**

Мета роботи — порівняти їхню асимптотичну складність і виміряти час виконання на різних обсягах даних.

# Теоретичний аналіз

## Сортування вставками

Алгоритм проходить по масиву та вставляє елементи у потрібне місце.

Найгірший випадок: масив відсортований у зворотному порядку. **Складність:**

Найкращий випадок: масив уже відсортований. **Складність:** .

## Сортування бульбашкою

На кожній ітерації найбільший елемент "спливає" у кінець масиву.

Найгірший і середній випадок:

Найкращий випадок (якщо масив уже відсортований): .

# Реалізація алгоритмів

## Сортування бульбашкою

Цей алгоритм багаторазово проходить по масиву, порівнюючи сусідні елементи та змінюючи їх місцями, якщо вони розташовані в неправильному порядку.

## Сортування вставками

Алгоритм вставляє кожен новий елемент у відсортовану частину масиву, зсуваючи інші елементи за необхідності.

# Вимірювання часу виконання

Для оцінки продуктивності був використаний модуль time, що вимірює швидкість роботи алгоритмів на масивах різних розмірів:

Замірювання проводились для кожного алгоритму окремо, результати представлені у вигляді графіків.

# Побудова графіків

Були побудовані графіки залежності часу виконання від розміру масиву.

Графік показує, що сортування вставками швидше за бульбашкове сортування на малих обсягах даних, але обидва алгоритми демонструють квадратичну складність .

# Висновки

Обидва алгоритми працюють **O(n²)** у найгіршому випадку.

Сортування вставками швидше на невеликих обсягах даних.

Бульбашкове сортування повільніше й рідко використовується на практиці.

# Експорт звіту

Для збереження звіту у форматі .html використовувалася команда:

jupyter nbconvert lab\_2\_StudentLastName.ipynb --to html

# Відповіді на контрольні запитання

1. **Що таке O-нотація і чим вона відрізняється від Ω-нотації?**

* **O-нотація (Велике O)** описує **верхню межу** часу або простору, який алгоритм може використовувати. Вона дає оцінку найгіршого випадку складності алгоритму. Наприклад, якщо алгоритм має складність , це означає, що час виконання алгоритму не перевищить для деякої константи c, де n — це розмір вхідних даних.
* **Ω-нотація (Велике Омега)** описує **нижню межу**, тобто найкращий випадок, коли алгоритм працює швидше. Якщо алгоритм має складність Ω(n), це означає, що в найкращому випадку час виконання алгоритму не буде меншим за c⋅n для деякої константи c.

**Основна різниця** полягає в тому, що O-нотація оцінює найгірший випадок (верхню межу), а Ω-нотація — найкращий випадок (нижню межу).

1. **Яку часову складність має алгоритм сортування методом вставляння за найсприятливіших умов, коли масив відсортовано? Відповідь обґрунтувати.**

Коли масив відсортований, алгоритм сортування методом вставляння працює дуже ефективно. У найкращому випадку, кожен новий елемент порівнюється тільки з першим елементом відсортованої частини і вставляється на його місце. Тому для кожного елемента достатньо одного порівняння, і це дає лінійну складність. Тому часова складність алгоритму сортування методом вставляння у найкращому випадку дорівнює O(n) де n — це кількість елементів у масиві.

1. **Надати визначення ефективного алгоритму.**

**Ефективний алгоритм** — це алгоритм, який використовує мінімальну кількість ресурсів (часу та пам'яті) для виконання задачі в залежності від розміру вхідних даних. Алгоритм вважається ефективним, якщо його часова складність не перевищує допустимі межі для даної задачі, а також якщо він використовує оптимальну кількість пам'яті.

1. **Пояснити поняття головного параметра (розміру) завдання**.

**Головний параметр (розмір) завдання** — це змінна, яка визначає обсяг вхідних даних, які потрібно обробити в алгоритмі. Це може бути кількість елементів у масиві, кількість вершин і ребер у графі, кількість символів у рядку тощо. Від цього параметра залежить, як швидко буде виконуватися алгоритм, і яким буде його час виконання чи використання пам'яті.

1. **Функція часової складності має вигляд:** . **Як записати асимптотичну складність у O-нотації?**

Для визначення асимптотичної складності у **O-нотації** потрібно залишити тільки термін, що має найбільший порядок зростання, оскільки саме він буде визначати поведінку алгоритму для великих значень N. У даному випадку це , тому асимптотична складність функції F(N) у **O-нотації** буде:

Це означає, що для великих NNN інші терміни (наприклад, та −) будуть мати незначний вплив на час виконання алгоритму.

Інтерактивне посилання на репозиторій: <https://github.com/ASD-122-2025/LevchenkoDmytro/tree/main/LAB_3>