МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ ІМЕНІ МИХАЙЛА
ОСТРОГРАДСЬКОГО

Кафедра комп’ютерної інженерії та електроніки

ЗВІТ З ПРАКТИЧНОЇ РОБОТИ №4

з навчальної дисципліни

«Алгоритми та структури даних**»**

Тема «Алгоритми пошуку та їх складність»

Студентка гр. КН-24-1 Бояринцова П. С.
Викладач Сидоренко В. М.

Кременчук 2025

Тема роботи: Алгоритми пошуку та їх складність

1.1 Постановка завдання

Мета роботи: опанувати основні алгоритми сортування та навчитись методам аналізу їх асимптотичної складності.

Завдання: реалізувати алгоритми пошуку, визначити асимптотичну складність, порівняти алгоритми.

1.2 Розв’язання задачі

Завдання

1. Оцінити асимптотичну складність алгоритму лінійного пошуку у O-нотації в найгіршому і в найкращому випадку. Як можна покращити алгоритм лінійного пошуку?

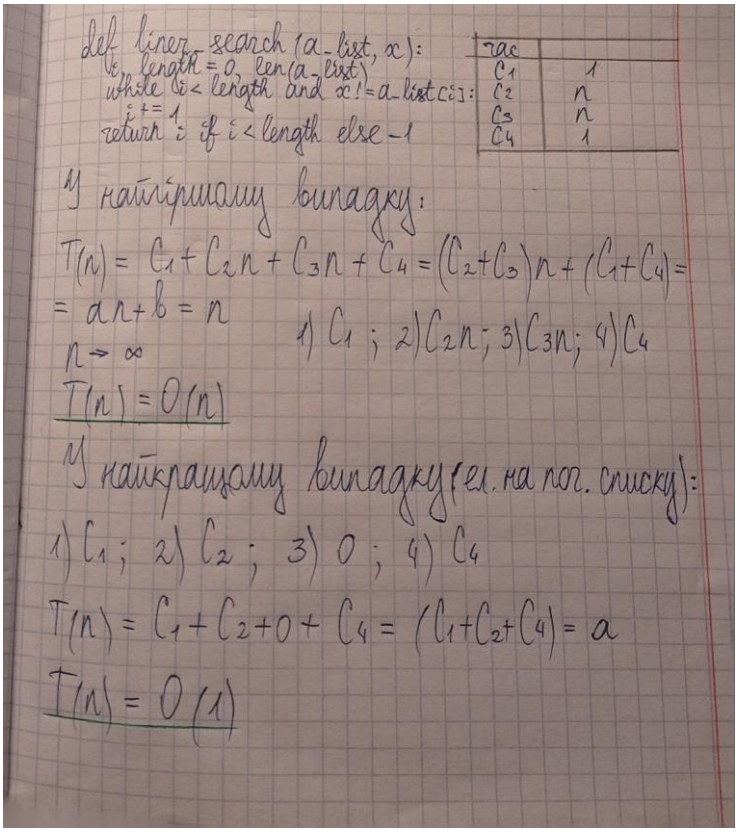

Рисунок 1 – Лінійний пошук

Алгоритм лінійного пошуку можна покращити, якщо попередньо впорядкувати масив.

2. Оцінити асимптотичну складність алгоритму бінарного пошуку у O-нотації в найгіршому і в найкращому випадку.

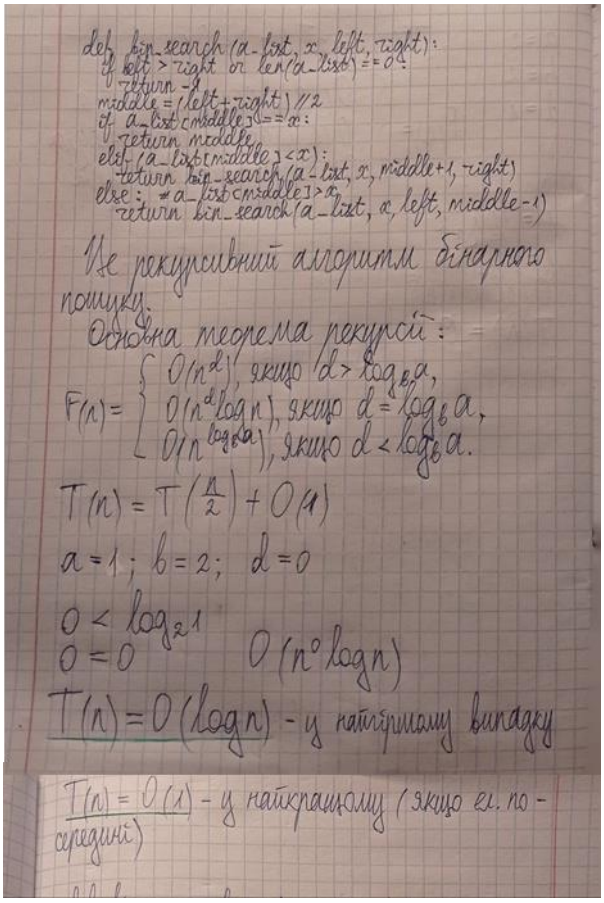

Рисунок 2 – Бінарний пошук

3. Побудувати алгоритм тернарного пошуку і оцінити його асимптотичну складність алгоритму у O-нотації в найгіршому і в найкращому випадку. Який з алгоритмів є оптимальнішим: бінарний, чи тернарний? Обґрунтувати відповідь відповідними обчисленнями.

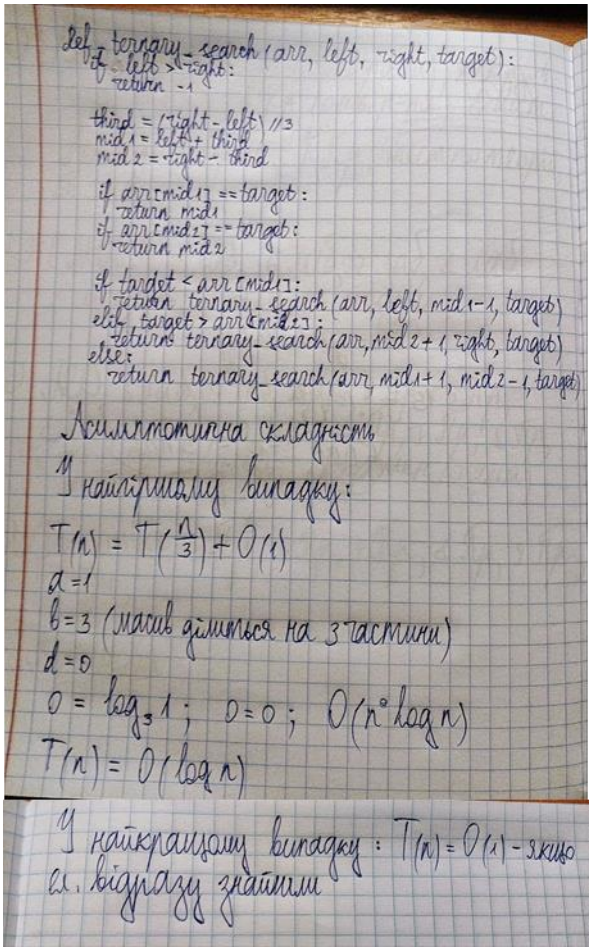

Рисунок 3 – Тернарний пошук

4. Порівняти ефективність алгоритмів лінійного, бінарного та тернарного пошуку для різних розмірів вхідного списку.
5. Порівняти алгоритми пошуку за їхньою здатністю працювати з відсортованими та не відсортованими списками.

Розв’язання:

1. Порівняння ефективності алгоритмів лінійного, бінарного та тернарного пошуку

Теоретична складність:

Алгоритм	Складність (найгірший випадок)
Лінійний пошук	$O(n)$
Бінарний пошук	$O(\log_2 n)$
Тернарний пошук	$O(\log_3 n)$

Порівняння логарифмів:

Оскільки

$$\log_2 n = \frac{\log_3 n}{\log_3 2} \approx 1.5849 * \log_3 n$$

тобто **бінарний пошук теоретично швидший**, бо виконує менше порівнянь.

2. Експериментальне дослідження

Ми виконали заміри часу для трьох алгоритмів пошуку на списках розмірів:

Для кожного розміру:

- Згенерували випадковий масив чисел.
- Відсортували для бінарного й тернарного пошуку.
- Виміряли час виконання кожного алгоритму.

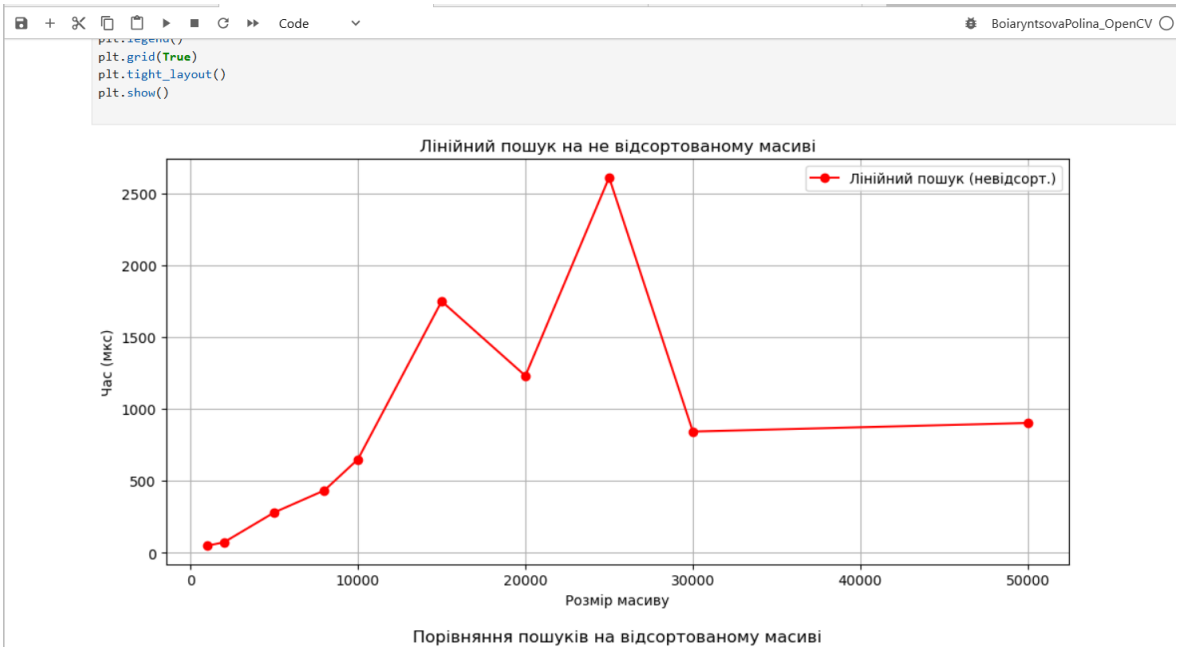

Рисунок 4 – Графік лінійного пошуку у відсортованому масиві

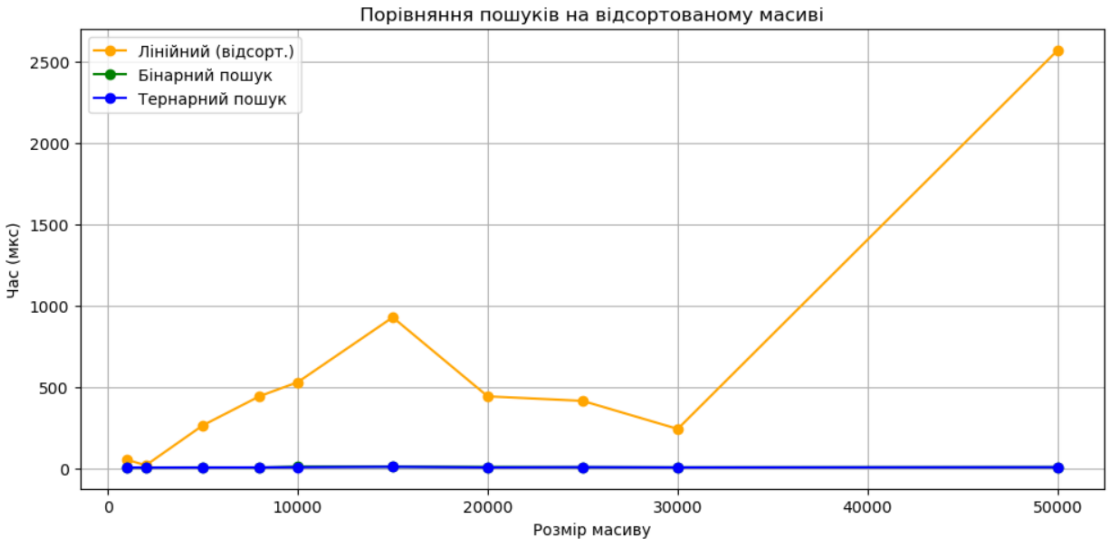

Рисунок 5 – Графіки пошуків у не відсортованому масиві

Висновок щодо ефективності:

- Бінарний пошук виявився найоптимальнішим у практиці: Менше кроків у порівнянні з тернарним (бо ділить масив на 2 частини, а не 3, але з меншими витратами на порівняння).
- Тернарний пошук має складність $O(\log_3 n)$, але в реальності виконує 2 порівняння на кожному кроці, тоді як бінарний — тільки 1.

3. Вплив відсортованості масиву на час пошуку

Алгоритм	Працює на несортованому?	Працює на відсортованому?
Лінійний	Так	Так
Бінарний	Ні	Так
Тернарний	Ні	Так

Остаточні висновки:

1. **Бінарний пошук є оптимальнішим**, ніж тернарний як теоретично, так і на практиці.
2. **Тернарний пошук** має гіршу ефективність через **2 перевірки замість 1**.
3. **Відсортованість масиву значно покращує час пошуку** для бінарного й тернарного пошуків.

4. **Лінійний пошук** — універсальний, але **неефективний** на великих масивах.

6. Розглянути сценарії використання кожного з алгоритмів пошуку у практичних задачах і обґрунтувати вибір кожного алгоритму в конкретному випадку.

1. Лінійний пошук

Як працює: перевіряє кожен елемент списку по черзі.

Коли використовувати:

- Список **не відсортований**
- Список **маленький**
- Потрібно **просте рішення**

Приклади:

- Шукаєш своє **ім'я** в списку присутніх
- Перевіряєш, чи є **помилка** у текстовому файлі
- Шукаєш **потрібну книгу** в купі книжок на столі

Плюси:

- Працює з будь-яким списком
- Дуже простий

Мінуси:

- Повільний для великих списків

2. Бінарний пошук

Як працює: ділить відсортований список навпіл і вирішує, в якій половині шукати далі.

Коли використовувати:

- Список **відсортований**
- Потрібно знайти **щось швидко**
- Часто виконується пошук

Приклади:

- Шукаєш слово в **алфавітному словнику**

- Вибираєш **товар по ID** в онлайн-магазині
- Шукаєш **номер** у відсортованому телефонному списку

Плюси:

- Дуже швидкий (особливо для великих списків)

Мінуси:

- Список має бути **відсортований**

3. Тернарний пошук

Як працює: ділить список на три частини замість двох, як у бінарному.

Коли використовувати:

- Список **відсортований**
- Потрібно знайти **максимум або мінімум**
- Задача на **оптимізацію**

Приклади:

- Шукаєш **оптимальну ціну**, щоб купити товар
- Потрібно знайти **найменше** або **найбільше** значення функції
- Визначаєш найкращий **час для поїздки**, щоб уникнути заторів

Плюси:

- Добрий для **спеціальних задач**

Мінуси:

- У звичайному пошуку **не кращий за бінарний**
- Складніший у реалізації

Висновок:

Алгоритм	Потрібно сортувати?	Працює з великими списками?	Коли використовувати
Лінійний	Ні	Ні	Для простих задач
Бінарний	Так	Так	Для швидкого пошуку
Тернарний	Так	Так	Для задач з максимумом/мінімумом

1.3 Відповіді на контрольні питання

1. Що таке алгоритм пошуку і чому він важливий у контексті комп'ютерних наук?

Алгоритм пошуку – це метод знаходження елемента в структурі даних. Важливий для обробки великих обсягів інформації, баз даних, пошукових систем.

2. Які основні критерії оцінки ефективності алгоритмів пошуку?

Швидкодія (час), пам’ять, адаптивність до типу вхідних даних.

3. Що таке лінійний пошук, і як він працює?

Алгоритм, що перебирає елементи один за одним. Простий, але повільний при великих об’ємах.

4. Які умови повинні бути виконані для успішного застосування бінарного пошуку?

Масив має бути відсортованим, без повторень для класичної реалізації.

5. Які переваги та недоліки використання бінарного пошуку порівняно з іншими алгоритмами пошуку?

– Перевага: Висока швидкість.
– Недолік: Потребує попереднтого сортування.

6. Що таке тернарний пошук, і в чому його відмінність від бінарного пошуку?

Тернарний розбиває список на три частини, бінарний – на дві. На практиці тернарний зазвичай повільніший, хоча має теоретично схожу складність.