МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ   
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ   
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА АВТОМАТИЗАЦІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

НАВЧАЛЬНА ДИСЦИПЛІНА  
Алгоритми та методи обчислень

ЗВІТ

З ПРАКТИЧНИХ РОБІТ

Виконав:

студент групи КН-23-1

Сидоренко А.Ю.

Кременчук 2024

**Практична робота № 4**

**Тема.** Алгоритми пошуку та їх складність

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

**Хід роботи**

**Завдання:**

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

Асимптотична складність алгоритму лінійного пошуку у найкращому випадку, коли шуканий елемент знаходиться відразу після початку перевірки = O(1)

Асимптотична складність алгоритму лінійного пошуку у найгіршому випадку, коли алгоритм повинен перевірити кожен елемент списку =

Щоб покращити алгоритм лінійного пошуку, можна використовувати структури даних, які дозволяють виконувати пошук за менший час.

Наприклад: Бінарний пошук, що поділяє список на дві частини і перевіряє середній елемент, що дозволяє відкинути приблизно половину елементів під час кожної ітерації. Або використання індексації, що створює індекси або таблиці для швидкого доступу до даних, що дозволяє зменшити кількість елементів, які потрібно перевірити.

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

Асимптотична складність алгоритму бінарного пошуку= , бо він на кожному кроці ділить діапазон пошуку навпіл, а кожен наступний крок скорочує розмір діапазону вдвічі. Тому в найгіршому і найкращому випадках асимптотична складність однакова.

1. **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.**

def ternary\_search(arr, target):

left, right = 0, len(arr) - 1

while left <= right:

mid1 = left + (right - left) // 3

mid2 = right - (right - left) // 3

if arr[mid1] == target:

return mid1

if arr[mid2] == target:

return mid2

if target < arr[mid1]:

right = mid1 - 1

elif target > arr[mid2]:

left = mid2 + 1

else:

left = mid1 + 1

right = mid2 - 1

return -1

Асимптотична складність алгоритму тернарного пошуку у найгіршому і найкращому випадку =, бо алгоритм поділяє масив на три рівні частини і перевіряє два елементи на індексах mid1 і mid2, що дозволяє відкинути приблизно дві третини елементів під час кожної ітерації.

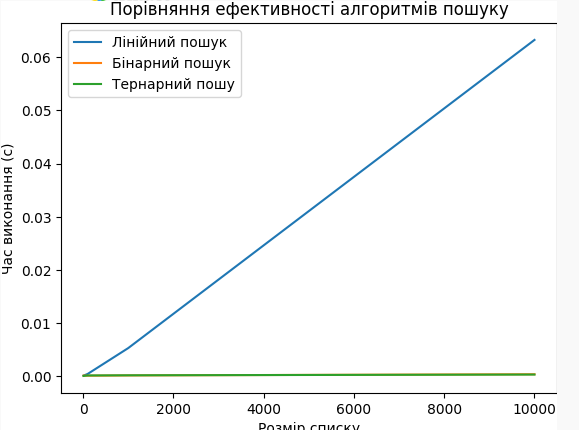
Так, як асимптотична складність бінарного і тернарного пошуку:

Бін. пошук = ,

Тернарний пошук = ,

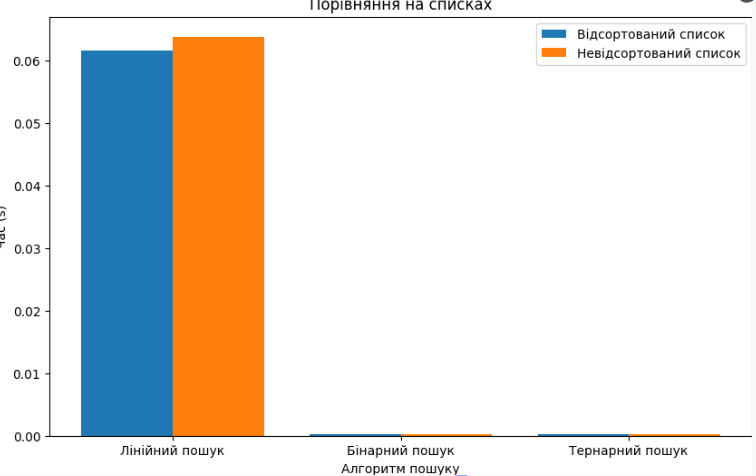
Оскільки асимптотична складність тернарного пошуку є меншою, ніж бінарного, але різниця незначна. На практиці бінарний пошук є більш оптимальним, оскільки він має меншу константу у логарифмічному виразі і є простішим у реалізації. Крім того, тернарний пошук може мати більше накладних витрат на виконання додаткових арифметичних операцій і порівнянь під час кожної ітерації.

1. Порівняти ефективність алгоритмів лінійного, бінарного та тернарного пошуку для різних розмірів вхідного списку. Для цього провести експериментальне дослідження та побудувати графіки залежності часу виконання алгоритму від розміру вхідного списку.



Згідно графіку, бінарний пошук і тернарний є більш ефективним для великих розмірів масивів, особливо коли масиви вже відсортовані. Для невеликих розмірів масивів, коли асимптотична складність має менший вплив, лінійний пошук може бути ефективнішим, оскільки він не має додаткових витрат на розділення списку.

1. **Порівняти алгоритми пошуку за їхньою здатністю працювати з відсортованими та не відсортованими списками. Провести аналіз впливу відсортованості списку на час виконання кожного алгоритму.**



Для бінарного і тернарного пошуку час виконання значно залежить від відсортованості списку. У випадку відсортованого списку час виконання буде значно меншим, ніж у випадку не відсортованого списку, оскільки ці алгоритми використовують властивості впорядкованого списку для ефективного пошуку. На відміну від них, лінійний пошук не залежить від відсортованості списку, оскільки він перевіряє кожен елемент по порядку.

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

Лінійний пошук:

* **Пошук елемента в невеликому списку**: Лінійний пошук може бути ефективним у невеликих списках, де швидкість виконання не є критичною. Такий підхід може бути зручним, оскільки не вимагає впорядкування списку.
* **Пошук першого або останнього входження елемента**: Якщо необхідно знайти перше або останнє входження елемента у списку, лінійний пошук може бути використаний без впливу впорядкування.

Бінарний пошук:

* **Пошук у великому відсортованому списку**: Бінарний пошук ефективно працює з відсортованими списками та швидко знаходить шуканий елемент, зменшуючи область пошуку удвічі з кожним кроком. Це дозволяє швидше знаходити елементи великих обсягів даних.
* **Пошук у великому відсортованому масиві**: Бінарний пошук може бути корисним у великих відсортованих масивах, де важлива швидкодія алгоритму.

Тернарний пошук:

* **Пошук у великому відсортованому списку з невеликими ділянками**: Тернарний пошук подібно до бінарного пошуку, але розділяє область пошуку на три частини з кожним кроком. Це може бути корисним, коли область пошуку має нерівномірну густину елементів або коли є області, де елементи швидше знаходяться.

**Контрольні питання**

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

Алгоритм пошуку - це метод або процедура, призначена для знаходження конкретного елемента або набору елементів у великому наборі даних.

У контексті комп'ютерних наук алгоритми пошуку використовують у пошуку в інтернеті, базах даних, обробці теексту, пошуку певного інформації, ШІ, комп’ютерній графіці та обробці тексту

1. **Які основні критерії оцінки ефективності алгоритмів пошуку?**

До основних критеріїв оцінки алгоритмів пошуку відносять: час виконання, пам’ять яка використовується, асимптотична складність алгоритму, швидкодія на різних типах масивів, швидкодія на великих обсягах даних.

1. **Що таке лінійний пошук, і як він працює?**

Одним з найпростіших алгоритмів пошуку є лінійний пошук. Сенс алгоритму полягає в тому, що пошук починають з першого елементу масиву. Якщо поточний елемент списку не дорівнює шуканому значенню, то здійснюється перехід до наступного елементу. Таким чином, у результаті кожної перевірки область пошуку зменшується на один елемент.

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

Успішне застосування бінарного пошуку виконується якщо наші дані однотипні, ми маємо доступ до індексів, масив або список- відсортовані та ми маємо невелику кількість елементів.

Якщо всі ці умови виконані, бінарний пошук може бути дуже ефективним і швидким способом пошуку елементів у відсортованому масиві або списку.

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

До переваг я віднесу ефективність у відсортованих даних, стійкість до великих обсягів даних та простоту реалізації

До недоліків- це необхідність доступу до індексів, вимогу до важливості відсортованості даних та можливі помилки.

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

Тернарний пошук - це алгоритм пошуку, який подібний до бінарного пошуку, але замість того, щоб розділяти область пошуку на дві частини, він розділяє її на три частини. Принцип роботи тернарного пошуку полягає в тому, щоб розділити вхідний відсортований список на три рівні частини і порівняти шуканий елемент з елементами у цих трьох точках.

Це дозволяє тернарному пошуку пропускати деякі області пошуку, що робить його ефективнішим для деяких видів даних або ситуацій. Однак, варто відзначити, що тернарний пошук може бути трохи складнішим у реалізації порівняно з бінарним