Практична робота №4

№1

Асимптотичну складність алгоритму лінійного пошуку можна оцінити так:

1.У найгіршому випадку (елемент, який шукаємо, знаходиться в кінці списку або взагалі відсутній):

Алгоритм проходить усі елементи один за одним, тому часова складність буде O(n) , де n - кількість елементів у списку.

2.У найкращому випадку (елемент, який шукаємо, знаходиться на початку списку):

Алгоритм знайде елемент з першої спроби, тому часова складність буде O(1).

Способи покращення алгоритму лінійного пошуку

1.Сентинельний лінійний пошук: Додавання "сентинела" (особливого елемента, що дорівнює шуканому) до кінця списку може зменшити кількість перевірок на межу списку, що покращує продуктивність у великих списках.

2.Бінарний пошук: Якщо список відсортований, можна застосувати бінарний пошук, який має асимптотичну складність у найгіршому випадку. Це значно швидше, ніж лінійний пошук для великих даних.

3.Індексація або геш-таблиця: Якщо треба виконувати багато пошуків у великому списку, ефективним варіантом є створення індексу або використання геш-таблиці, що забезпечує середню складність пошуку у випадку геш-таблиць.

№2

Асимптотичну складність алгоритму бінарного пошуку можна оцінити так:

1.У найгіршому випадку:

Бінарний пошук працює, розділяючи відсортований список навпіл на кожній ітерації. Кількість ітерацій буде дорівнювати логарифму від кількості елементів у списку, тому часова складність у найгіршому випадку — O(log n) , де n — nкількість елементів у списку.

2.У найкращому випадку:

Найкращий випадок виникає, коли шуканий елемент знаходиться посередині списку при першій перевірці. У такому разі алгоритм завершиться за одну ітерацію, і часова складність буде O(1) .

№3

Алгоритм тернарного пошуку використовується для пошуку елемента у відсортованому масиві. Він поділяє масив на три частини, на відміну від бінарного пошуку, який ділить масив навпіл. Алгоритм працює наступним чином:

1.На кожній ітерації розділяє відсортований масив на три рівні частини за індексами:

перший розділовий індекс, m1 , визначає кінець першої третини масиву,

другий розділовий індекс, m2 , визначає кінець другої третини масиву.

2.Порівнює шуканий елемент із значеннями на індексах m1 та : m2

Якщо елемент дорівнює значенню на m1, пошук завершено.

Якщо елемент дорівнює значенню на m2, пошук також завершено.

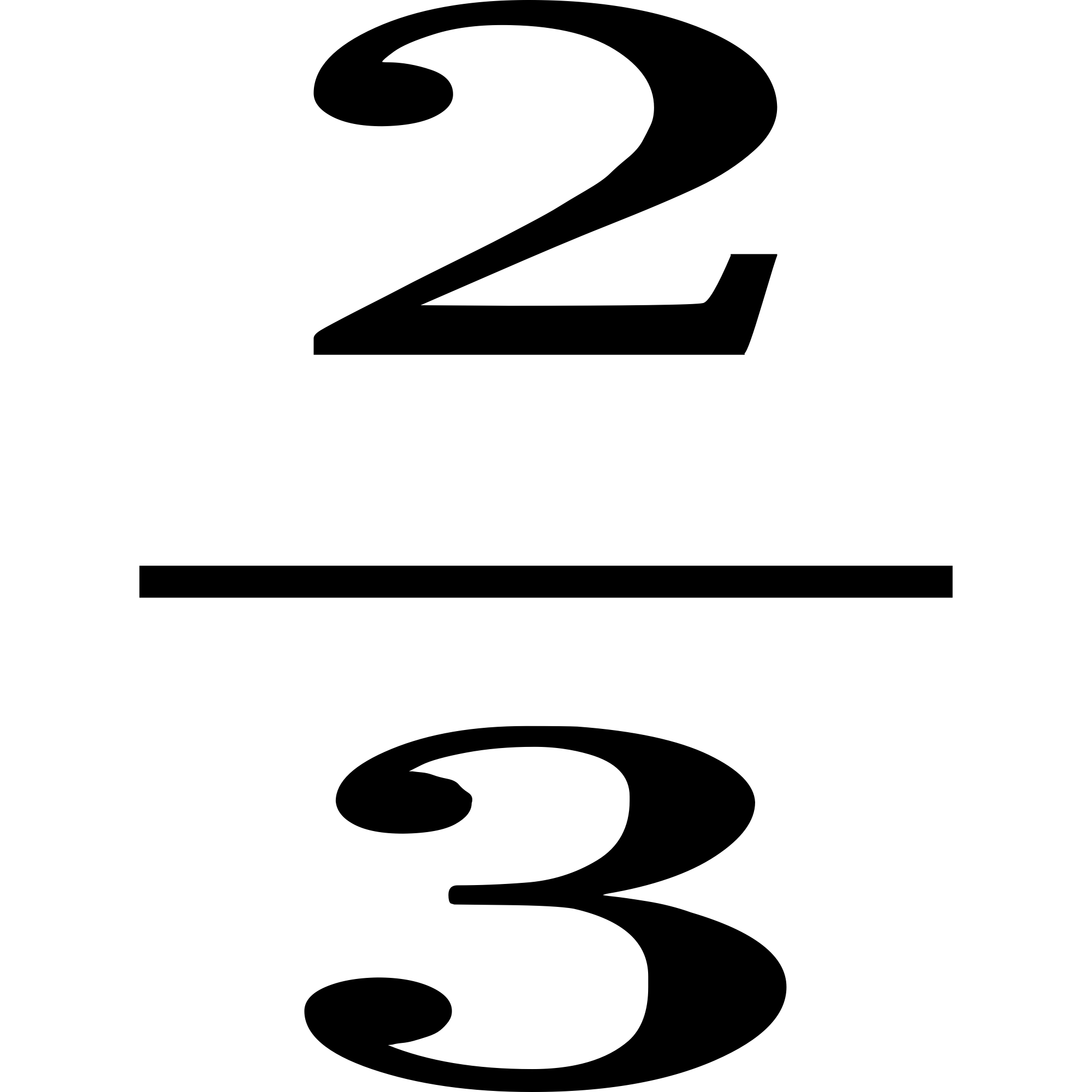
Якщо елемент менший за значення на m1 , пошук триває в першій третині масиву.

Якщо елемент знаходиться між m1 і m2, пошук триває у другій третині.

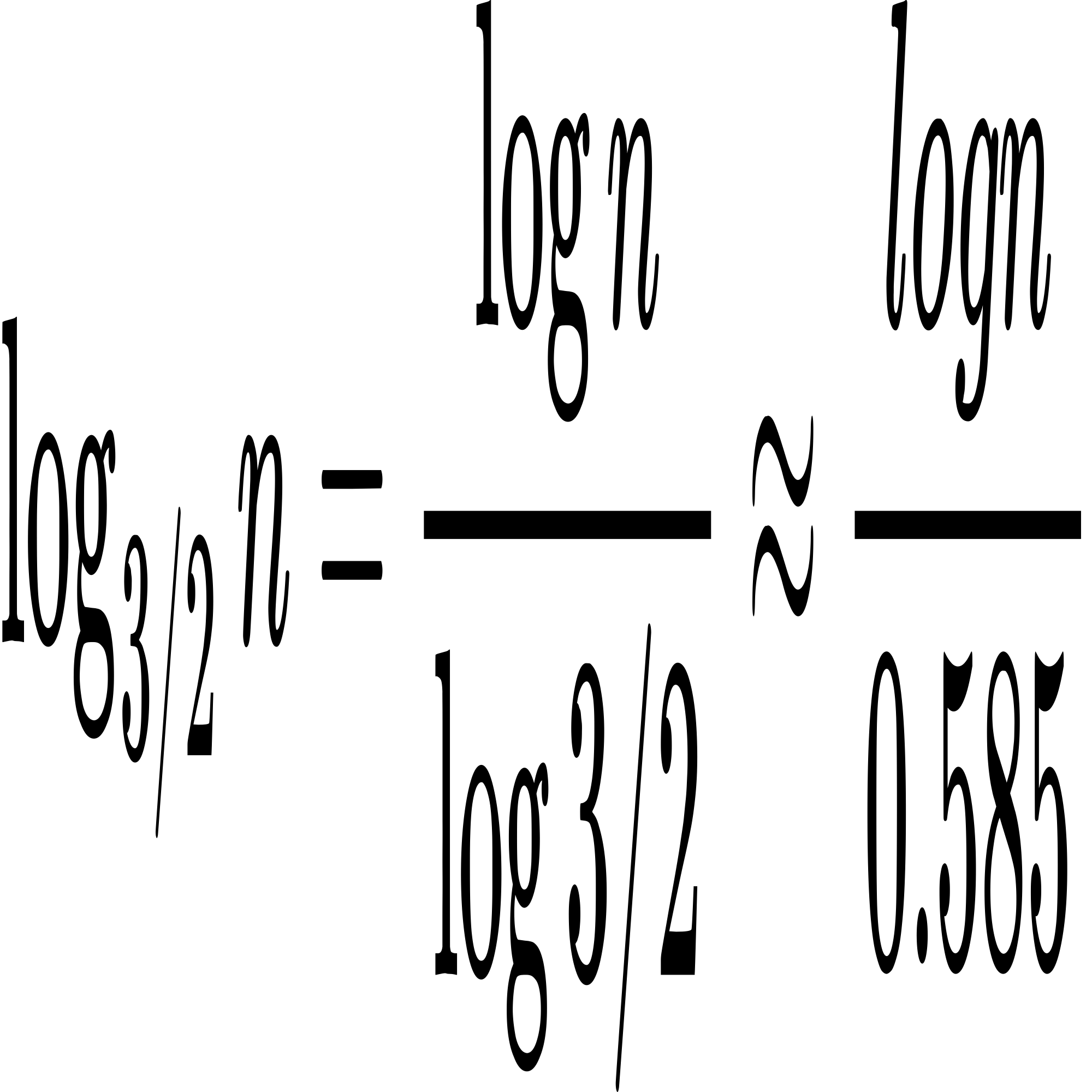
Якщо елемент більший за значення на m2 , пошук триває в третій третині.   
3.Повторює процес на зменшеній ділянці масиву, поки елемент не буде знайдено або не залишиться жодної ділянки для пошуку.

Асимптотична складність тернарного пошуку

1.У найгіршому випадку:

На кожній ітерації тернарний пошук скорочує розмір діапазону до  від поточного розміру.

Кількість ітерацій у тернарному пошуку буде приблизно wps.

Переписавши це у вигляді натурального логарифму: .

Отже, асимптотична складність тернарного пошуку в найгіршому випадку буде O(log n) , аналогічно до бінарного пошуку, але з більшою константою через поділ на три частини.

2.У найкращому випадку:

Як і в бінарному пошуку, найкращий випадок виникає, якщо шуканий елемент знаходиться одразу на одному з розділових індексів (wps або wps). У такому разі алгоритм завершує роботу за одну ітерацію, і складність буде O(1) .

Порівняння тернарного та бінарного пошуку

Для порівняння, розглянемо кількість ітерацій для бінарного і тернарного пошуку для великих n:

Бінарний пошук зменшує розмір масиву до wps кожного разу, і кількість ітерацій дорівнює wps

Тернарний пошук зменшує розмір масиву доwps кожного разу, що вимагає приблизно wps ітерацій..

Оскільки wps , тернарний пошук потребує трохи більше ітерацій для того ж , ніж бінарний пошук. Хоча обидва алгоритми мають складність O(log n) , бінарний пошук є ефективнішим, оскільки має меншу константу і вимагає менше порівнянь на кожному кроці.

№4

1. Реалізація алгоритмів пошуку

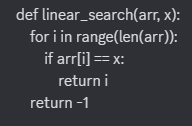
1.Лінійний пошук: перевіряє кожен елемент списку по черзі.

2.Бінарний пошук: шукає елемент у відсортованому списку, поділяючи його на половини.

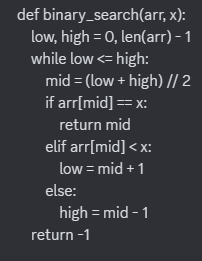
3.Тернарний пошук: аналогічний бінарному, але ділить масив на три частини.

Код для вимірювання часу виконання:

Лінійний



Бінарний



Тернарний

