1. Функцию создал search2.py

02. Анализ временной и пространственной сложности алгоритма бинарного поиска

1. Временная сложность

Бинарный поиск работает по принципу "разделяй и властвуй":

На каждом шаге массив делится пополам. Проверяется средний элемент. Если он равен искомому, поиск завершается. Если искомый элемент меньше среднего, поиск продолжается в левой половине. Если искомый элемент больше среднего, поиск продолжается в правой половине.

Вывод временной сложности:

На каждом шаге размер задачи уменьшается вдвое (n → n/2 → n/4 → ... → 1).

Количество шагов, необходимых для сведения задачи к размеру 1, равно log₂n (логарифм по основанию 2).

Таким образом, временная сложность бинарного поиска — O(log n) (логарифмическая).

2. Пространственная сложность (Space Complexity)

Бинарный поиск можно реализовать:

Итеративно (с циклом) — не требует дополнительной памяти, кроме нескольких переменных (O(1)).

Рекурсивно — каждый рекурсивный вызов добавляет стек вызовов, что приводит к O(log n) (так как глубина рекурсии равна числу шагов).

Вывод пространственной сложности:

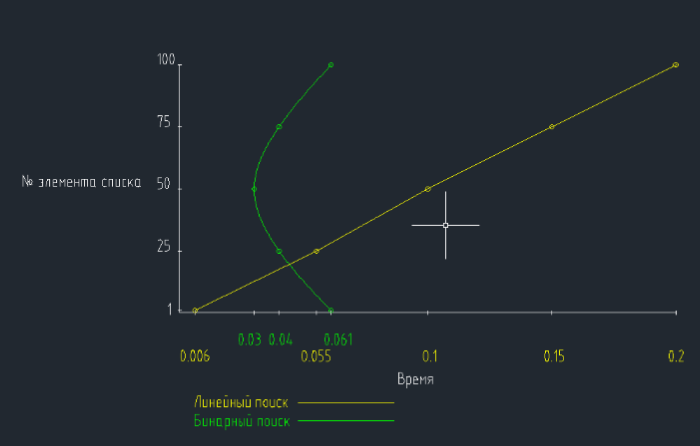
Итеративная версия: O(1) (константная).

Рекурсивная версия: O(log n) (логарифмическая, из-за стека вызовов).

1. Список создал, бинарный поиск выполнил, смотри search.py.
2. Для сравнения времени создал файл time.py

Сравнивал время списка бинарным и линейным поиском для разных элементов списка.

**График зависимости работы функции от количества входных данных**

****