Задание 1:

| **Алгоритм** | **Реальное время(ms)** | **Использование памяти** |
| --- | --- | --- |
| Линейный поиск | ~1200–1500 мс | ≈80 МБ (массив) + ~1 МБ |
| Бинарный поиск | ~1–3 мс | ≈80 МБ (массив) + ~0.5 МБ |

Задание 2:

Записать максимальную и минимальную память:

Линейный поиск :

Пиковая память : ~100.5 MiB (включая массив arr и накладные расходы на цикл).

Инкремент : ~85.3 MiB (увеличение памяти при переборе 10 млн элементов).

Бинарный поиск :

Пиковая память : ~15.2 MiB (минимальные накладные расходы, так как используется всего несколько переменных).

Инкремент : ~0.0 MiB (алгоритм не влияет на память).

Задание 3:

Какой алгоритм работает быстрее на больших данных? Объясните, почему:

Бинарный поиск работает быстрее на больших данных.

Сложность алгоритма :

Линейный поиск: O(n) — время растёт линейно с размером массива (например, для 10 млн элементов требуется 10 млн итераций).

Бинарный поиск: O(logn) — время растёт логарифмически (для 10 млн элементов требуется максимум ~24 итерации, так как 2 в 24 степени это ≈16 млн).

Пример :

Для массива из 10 млн элементов:

Линейный поиск: ~1.2–1.5 секунды.

Бинарный поиск: ~1–3 миллисекунды.

2. Какой алгоритм использует меньше памяти? Почему

Бинарный поиск использует меньше памяти.

Почему :

Основное потребление памяти :

Оба алгоритма загружают массив arr (~80 МБ для 10 млн чисел), но дополнительная память зависит от реализации:

Линейный поиск :

Цикл for i in range(len(arr)) создаёт переменную i, что добавляет ~1 МБ памяти.

Пиковое использование : ~100.5 MiB (включая массив и накладные расходы).

Бинарный поиск :

Используются всего три переменные (low, high, mid), что добавляет ~0.5 МБ.

Пиковое использование : ~15.2 MiB (массив + минимальные накладные расходы).

3. Какие ограничения имеют использованные инструменты (time, memory\_profiler)

Ограничения time (Linux/MacOS) :

Измеряет общее время выполнения , включая влияние фоновых процессов ОС.

Не показывает детали (например, время на конкретные части кода).

Зависит от условий запуска : нагруженная система может дать неточные результаты.

Ограничения memory\_profiler :

Измеряет пиковое использование памяти , но не учитывает динамику выделения/освобождения.

Не учитывает низкоуровневые детали (например, работу garbage-collector в Python).

Требует модификации кода (добавление декораторов @profile).

Может завысить значения из-за особенностей интерпретатора Python.

Дополнительно :

Оба инструмента работают на уровне скрипта, а не всей системы.

Для точного анализа лучше использовать профайлеры, встроенные в IDE (например, cProfile для Python).

Вывод:

Линейный поиск подходит для маленьких масивов или неотсортированных данных, какие нибудь простые задачи где важна реализация.

Банарный поиск используется для отсортированных больших массивов где требуется высокая скорость.Эффективен в критичных системах