### **Теоретическая часть**

1. **Программный измерительный монитор** — инструмент для сбора данных о работе программы (время выполнения, использование памяти и др.).  
   **Примеры инструментов:**
   * time (замер времени выполнения).
   * memory\_profiler (анализ использования памяти).
   * cProfile (профилирование кода).
2. **Характеристики ПО для мониторинга:**
   * Время выполнения.
   * Использование оперативной памяти.
   * Загрузка CPU.
3. **Этапы работы:**
   * Запуск программы с монитором (например, time python script.py).
   * Сбор данных (логирование времени, памяти).
   * Интерпретация результатов (анализ таблиц, графиков).

### **Практическая часть**

#### **Задание 1: Измерение времени выполнения**

**Результаты (условные, неидеальные):**

| **Алгоритм** | **Реальное время (ms)** | **Использование памяти (MiB)** |
| --- | --- | --- |
| Линейный поиск | 1200 | 15.7 |
| Бинарный поиск | 5 | 22.3 |

**Примечание:**

* Линейный поиск медленнее, так как проверяет все элементы.
* Бинарный поиск быстрый, но требует отсортированный массив.

#### **Задание 2: Измерение использования памяти**

**Вывод**memory\_profiler**для**binary\_search.py**(пример):**

============================================================

1 45.2 MiB 45.2 MiB 1 @profile

2 def binary\_search(arr, target):

3 45.2 MiB 0.0 MiB 1 low, high = 0, len(arr) - 1

4 45.2 MiB 0.0 MiB 5 while low <= high:

5 45.2 MiB 0.0 MiB 4 mid = (low + high) // 2

6 45.2 MiB 0.0 MiB 4 if arr[mid] == target:

7 45.2 MiB 0.0 MiB 1 return mid

8 45.2 MiB 0.0 MiB 3 elif arr[mid] < target:

9 45.2 MiB 0.0 MiB 2 low = mid + 1

10 else:

11 45.2 MiB 0.0 MiB 1 high = mid - 1

12 45.2 MiB 0.0 MiB 1 return -1

**Пиковое использование памяти:** ~45.2 MiB (из-за генерации массива arr = list(range(1, 101))).

#### **Задание 3: Анализ результатов**

1. **Быстродействие:**
   * Бинарный поиск быстрее (O(log n)), но требует сортировки.
   * Линейный поиск медленнее (O(n)), но работает на любых данных.
2. **Память:**
   * Оба алгоритма используют мало памяти, но binary\_search показал большие цифры из-за генерации массива в коде.
3. **Ограничения инструментов:**
   * time: нет детализации по функциям.
   * memory\_profiler: добавляет накладные расходы, может искажать замеры.

### **Выводы**

1. **Когда использовать алгоритмы:**
   * **Линейный поиск:** для небольших или несортированных данных.
   * **Бинарный поиск:** для больших отсортированных массивов.
2. **Важность мониторинга:**
   * Критична при оптимизации высоконагруженных систем.
   * Помогает выявить узкие места (например, утечки памяти).