

## Тема: Алгоритм бинарного поиска

### Порядок выполнения работы:

1. Написал программу (binary\_poisk.py), в котором реализовал алгоритм бинарного поиска и посчитал время необходимое в среднем для выполнения поиска любого элемента в массиве

```
0.0000012711
0.0000013428
0.0000015331
0.0000016449
0.0000017098
0.0000018421
0.0000018453
0.0000019016
0.0000019373
```

Рисунок 1 – Результат выполнения программы binary\_poisk.py

Далее исходя из этих данных составил систему уравнений, состоящую из уравнений  $2850000a + 4500b = 0,009272$  и  $4500a + 9b = 0,0000184943$ . Решил её и получилось, что  $a = 0,0000001155$   $b = 0,0000012034214$ . Построил график функции  $y = 0,0000001155 * \log(x, 2) + 0,00000012034214$

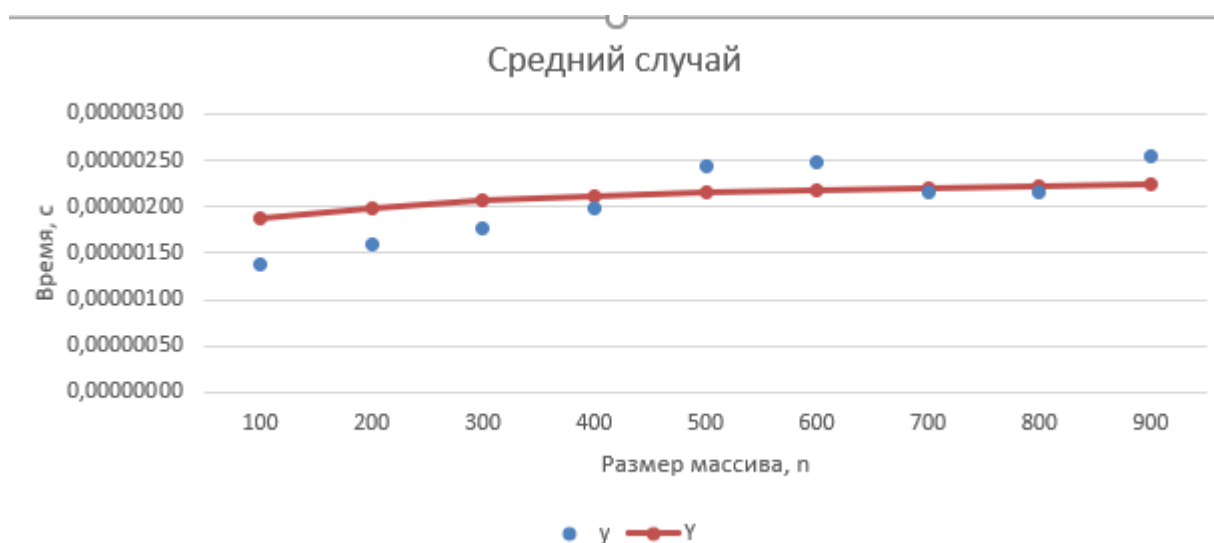
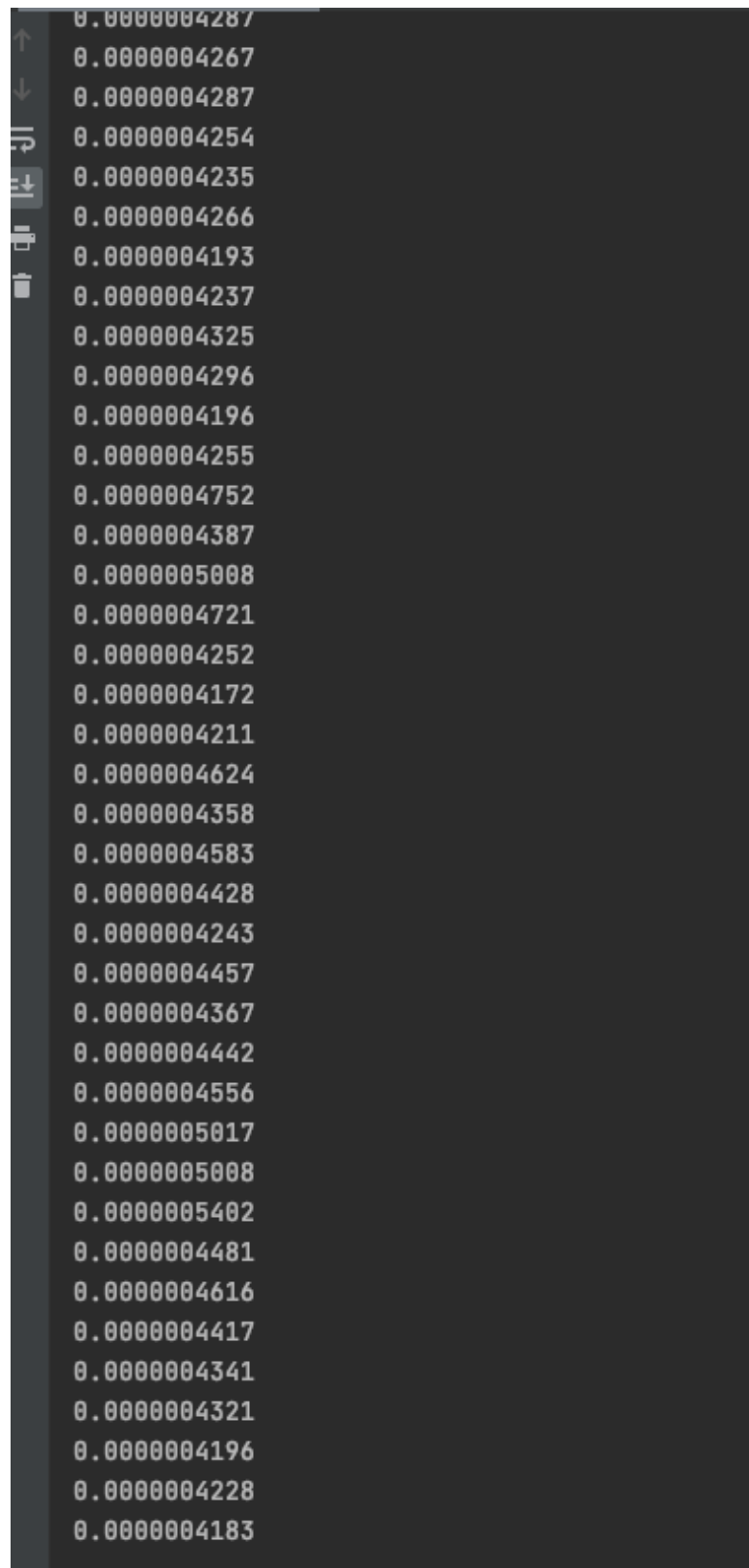


Рисунок 2 – График функции  $y = 0,0000001155 * \log(x, 2) + 0,00000012034214$

2. Написал программу (binary\_poisk\_py.py), в котором реализовал алгоритм бинарного поиска и посчитал время необходимое в среднем для выполнения поиска любого элемента в массиве



A screenshot of a terminal window with a dark background. On the left side, there is a vertical toolbar with icons for navigation (up, down), search, and other standard terminal functions. The main area of the terminal displays 40 lines of floating-point numbers, representing search times. The numbers are in scientific notation, with a mantissa of 0.000000 and an exponent of 4. The values range from 0.0000004183 to 0.0000005008. The numbers are listed in a seemingly random order, reflecting the 'any element' requirement mentioned in the text.

Search Time
0.0000004287
0.0000004267
0.0000004287
0.0000004254
0.0000004235
0.0000004266
0.0000004193
0.0000004237
0.0000004325
0.0000004296
0.0000004196
0.0000004255
0.0000004752
0.0000004387
0.0000005008
0.0000004721
0.0000004252
0.0000004172
0.0000004211
0.0000004624
0.0000004358
0.0000004583
0.0000004428
0.0000004243
0.0000004457
0.0000004367
0.0000004442
0.0000004556
0.0000005017
0.0000005008
0.0000005402
0.0000004481
0.0000004616
0.0000004417
0.0000004341
0.0000004321
0.0000004196
0.0000004228
0.0000004183

Рисунок 3 – Результат выполнения программы binary\_poisk\_py.py

Далее исходя из этих данных составил систему уравнений, состоящую из уравнений  $404250000a + 122500 = 0,0506374$  и  $122500a + 49b = 0,000019844$ . Решил её и получилось, что  $a = 0,00000000018029$   $b = 0,00000041146$ . Построил график функции  $y = 0,00000000018029 * \log(x, 2) + 0,00000041146$



Рисунок 4 – Входные данные и график функции  $y = 0,00000000018029 * \log(x, 2) + 0,00000041146$

### 3. Результаты выполнения для алгоритма линейного поиска

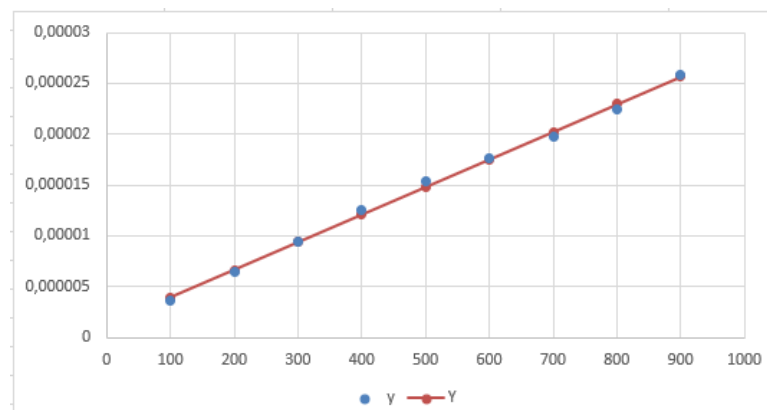


Рисунок 5 – График алгоритма линейного поиска

Вывод: в ходе проведенных исследований было установлено, что алгоритм бинарного поиска работает значительно быстрее алгоритма линейного поиска, а встроенный в Python алгоритм работает ещё быстрее.