Шестиперстов Валентин 22ПИ3

Лабораторная работа 1

Язык программирования: Python

Дополнительные модули: time, matplotlib

Цель работы: провести эксперимент с использованием разных алгоритмов поиска в матрице и выяснить, как работают алгоритмы при разных входных данных

Ход работы:

Входные данные 1:

A[i][j] = (N/M * i + j)*2

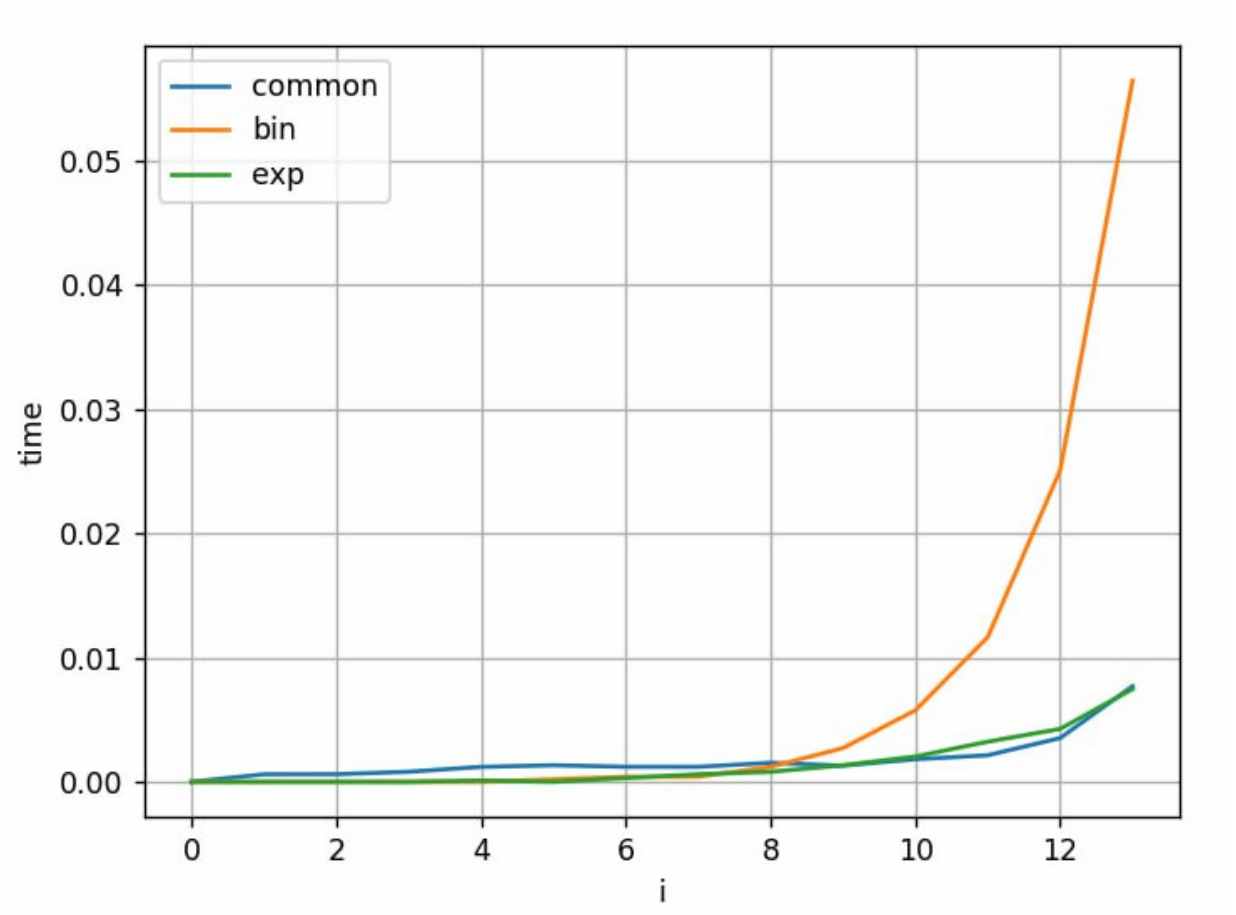
Target = 2*N + 1

N = 2^13

M = 2^i, где i = 0, 1, 2… 12, 13

Таргета в матрице нет, поэтому каждый алгоритм будет проходить матрицу полностью

График:



Вывод:

При i < 8, то есть количестве строк, меньшем 256 бинарный и экспоненциальный поиски работают за почти одинаковое время, а обычный поиск для отсортированной матрицы проигрывает во времени, но не сильно.

При i = 8 все три алгоритма работают за примерно одинаковое время

При i > 8 время для бинарного поиска резко начинает расти, то есть оно очень сильно зависит от количества строк, а экспоненциальный поиск начинает слегка проигрывать обыкновенному, скорее всего из-за того, что в нем присутствует бинарный поиск

Входные данные 2:

A[i][j] = (N/M * i * j)*2

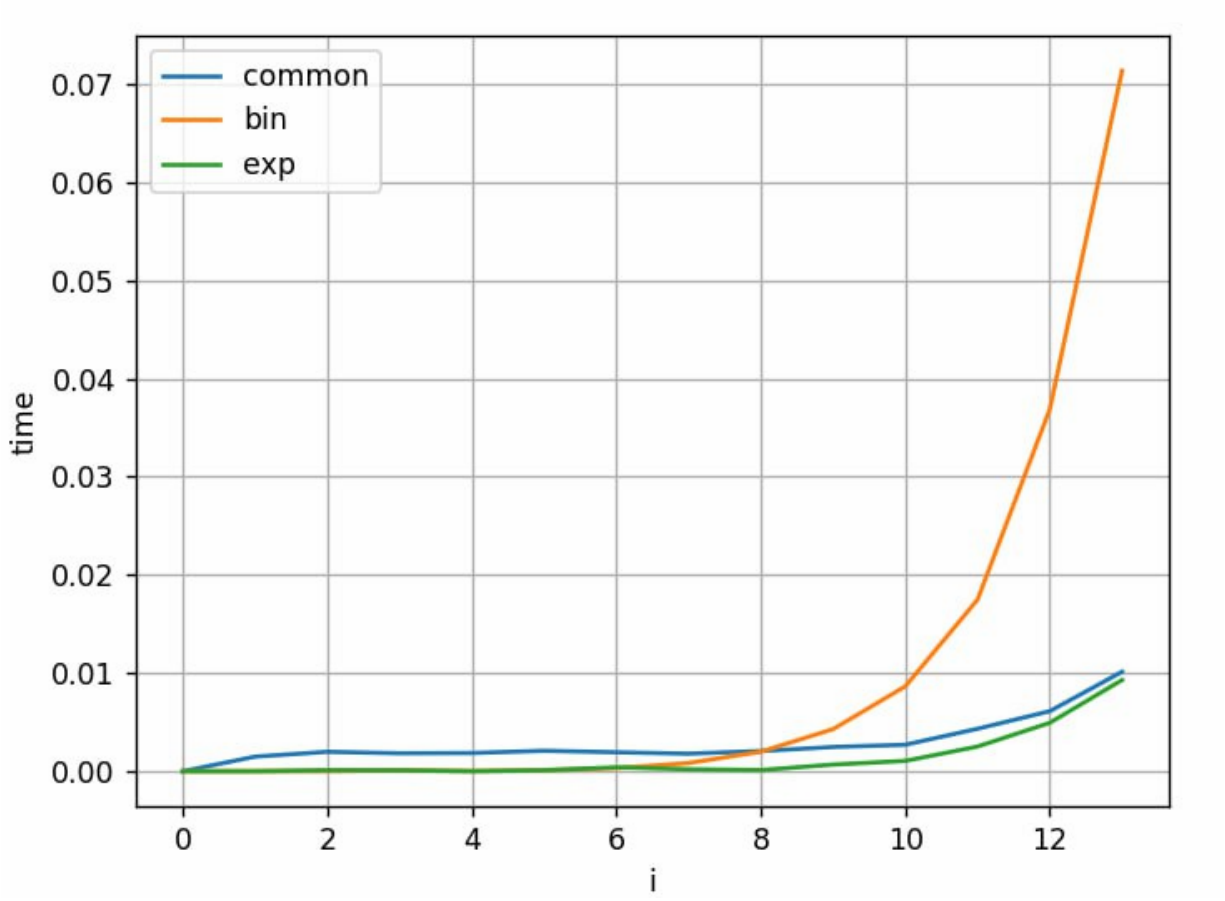
Target = 16*N + 1

N = 2^13

M = 2^i, где i = 0, 1, 2… 12, 13

Таргета в матрице нет, поэтому каждый алгоритм будет проходить матрицу полностью

График:



Вывод:

Бинарный поиск ведет себя ровно также, как и при первых входных данных

Обычный поиск же ухудшил свои показатели, это неудивительно, так как при вторых входных данных матрица заполнена неравномерно, то есть если при первых входных данных при больших i обычный алгоритм шел по диагонали, рисуя “лестницу”, то при вторых входных обычный алгоритм “даёт угла”.

Экспоненциальный поиск же наоборот улучшил свои показатели, как раз из-за неравномерности заполнения, на каждой строке данный поиск быстрее приходит к точке, из которой переходит на следующую строку

Общий вывод:

Очевидно, что лучшим решением этой задачи является экспоненциальный поиск, однако он сложен в реализации

Бинарный поиск прост в написании но при этом критически плохо работает с большим количеством строк

Обыкновенный поиск – компромисс, прост в написании и работает за разумное время при любом количестве строк, проигрывая экспоненциальному либо при малом количестве строк, либо при случаях с неравномерным заполнением (причем не всех, например, если ответ/значение, с которого мы переходим на следующую строку находится близко к концу строки, то обыкновенный поиск будет искать это значение не дольше экпоненциального).