Отчёт о выполнении работы по оценке полезности применения AVL-деревьев

Сыркин Дмитрий, группа Б01-201, ФРКТ

11 декабря 2022 г.

Цель работы: Реальзация бинарного дерева поиска, создание его модификации AVL-дерева, реализация поиска по префиксу в дереве, оценка скорости создания, балансировки дерева и поиска в зависимости от типа дерева и от степени отсортированности первоначальных данных. Определения области применения AVL-деревьев.

1 Теоретические данные.

Для AVL-дерева характерны следующие оценки времени выполнения(худшее время):

поиск $O(\log n)$
вставка $O(\log n)$
удаление $O(\log n)$

Для бинарного дерева характерны следующие оценки времени выполнения:
среднее значение:

поиск $O(\log n)$
вставка $O(\log n)$
удаление $O(\log n)$

худшее время:

поиск $O(n)$
вставка $O(n)$
удаление $O(n)$

Из выше указанного следует, что AVL-дерево должно быть быстрее. Тем не менее данный факт не является тривиальным и требует практической проверки.

2 Выоплнение работы.

2.1 Ход работы.

В ходе работы в первую очередь было реализовано бинарное дерево и поиск по префиксу в нём. Ввиду нехватки ресурсов и времени было принято решение писать программу в императивном стиле, что повышало(по крайней мере для меня) читаемость кода, а также возможность быстро его модифицировать*.

Также с целью оптимизации, были попытки сократить объем кода за счёт встроенных функций, но по не выявленной пока причине, логика выполнения программы полностью ломалась, делая такие замены бессмысленными.(корректировка данного пункта остаётся на неопределённый срок)

Как и при замерах времени сортировки был обнаружен некий обрыв. При более подробном рассмотрении данной проблемы, выяснилось что данный деффект наблюдается и на других устройствах, с другими реалтизациями программы. Поэтому основной причиной возможно является несовершнная реализация функции очистки памяти после использования дерева.

Для избежания появления сильно выраженного обрыва, также как и в прошлый раз, программа выполнялась на одних и тех же данный по 5 раз. Поскольку вычеслительные мощности весьма ограничемы, шаг,с которым проводились замеры составлял 500000 значений(от 500000 до 10000000)

Из полученных результатов видно, что на создание AVL-дерева тратится значительно больше времени,чем на создание обычного дерева(по крайней мере в рассматриваемом диапозоне). Такде оба алгоритма работали значительно быстрее на отсортированных файлах.

*Весь код в итоге был переписан. Отдельно реализовано бинарное дерево(далее просто bin), изменены аргументы функций балансировки, инициализациии, убраны лишние функции, ограничено использование операции взятия адреса, что позволило несколько ускорить avl.c и bin.c. В итоге для avl.c изменение скорости выполнения было не столь значительным, в то время как bin.c стал работать на 25% быстрее.

Проверка скорости выполнения выполнялась на трёх текстовых файлах:

1)Неотсортированные случайные числа от 1 до 10000000

2)сортированные в порядке возрастания(1,2,3...) (проверка на prefixSearch по данному файлу не проводилась)

3)Сортированные+ (или просто сортированные для prefixSearch) (1, 10, 100, 101, 2, 20...)

3 Вывод.

Как видно в большинстве случаев, AVL дерево медленее модифицируется, давая при это мне столь значительный прирост скорости, за исключением случая когда Bin вырождается в список. Таким образом AVL деревья наиболее эффективны когда задача поиска более приоритетна чем добавление или удаление эллементов (графики приложены ниже).

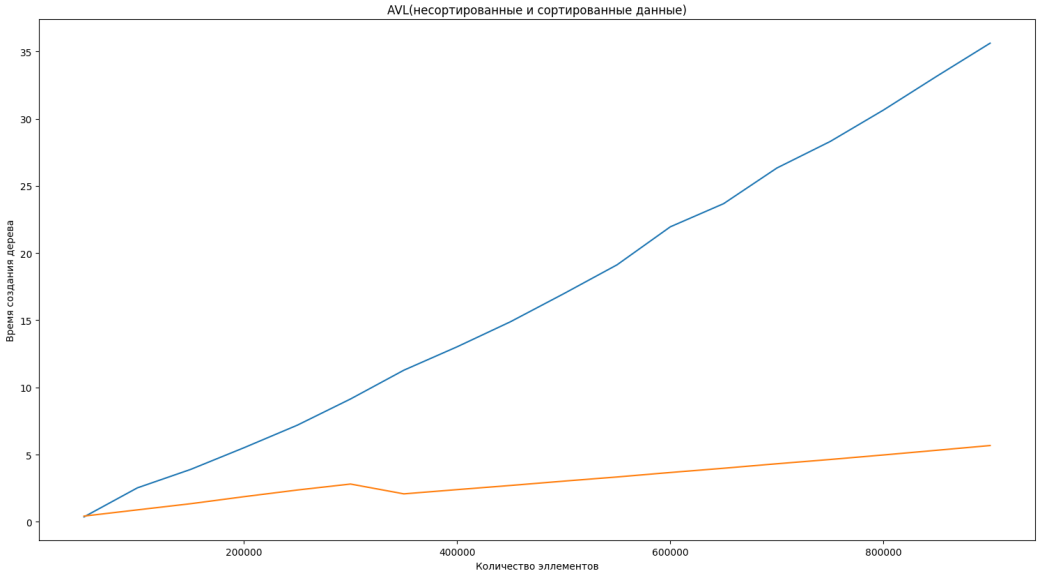

Рис. 1: Сравнение AVL(сортированные и несортированные данные)

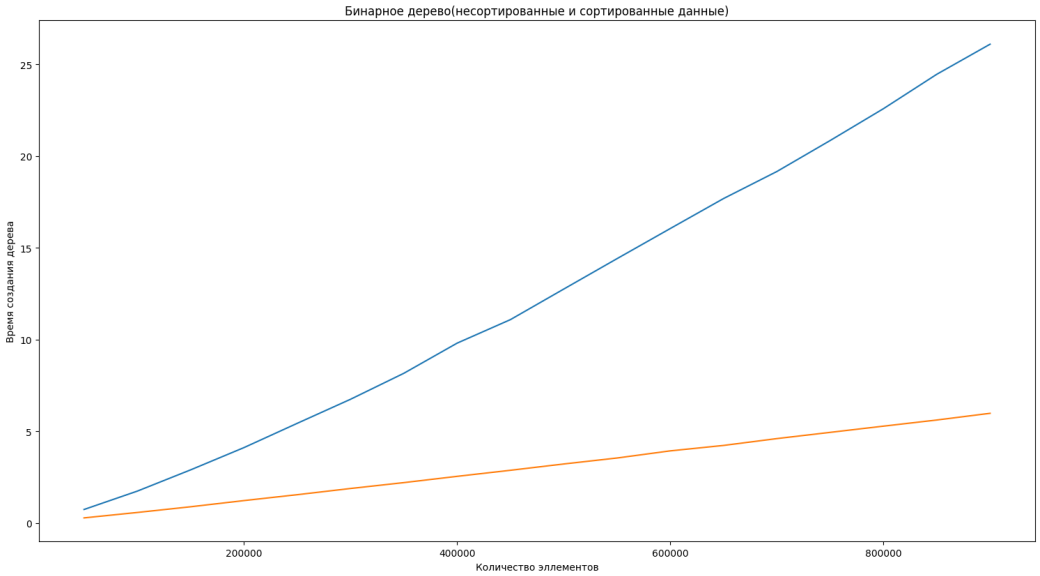

Рис. 2: Сравнение , бинарного дерева(сортированные и несортированные данные)

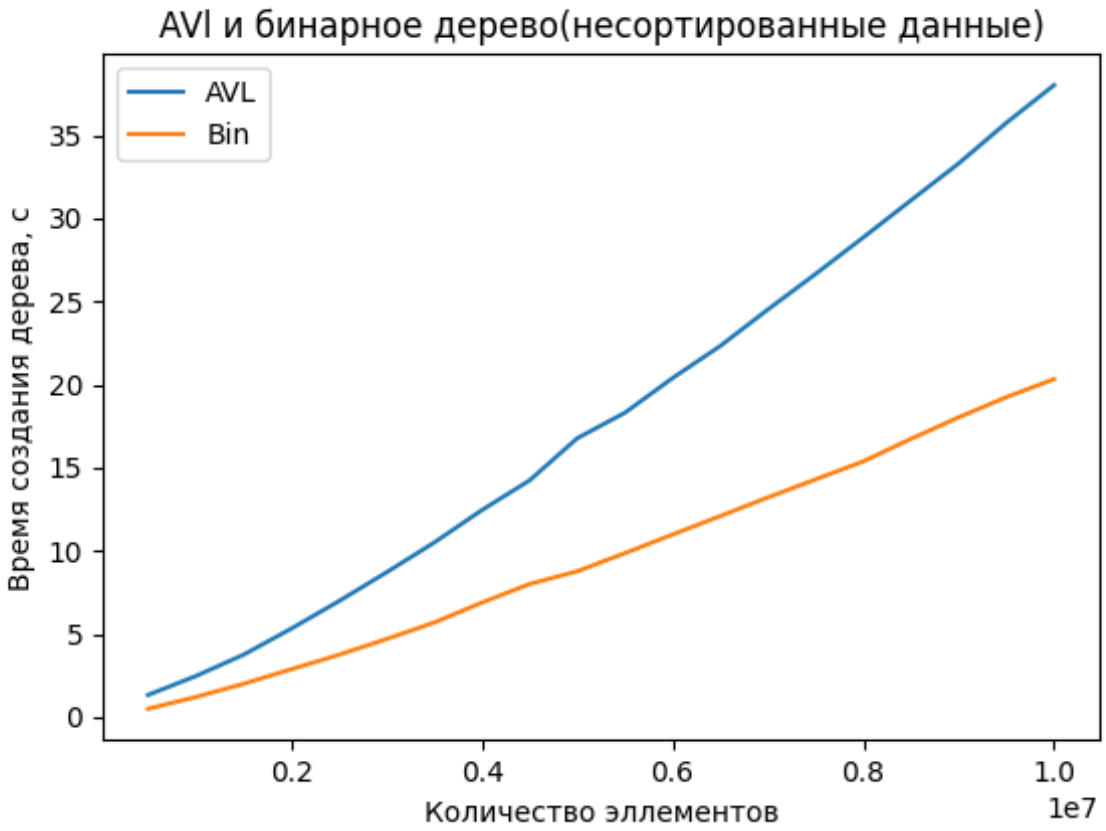

Рис. 3: Сравнение AVL и бинарного дерева(несортированные данные)

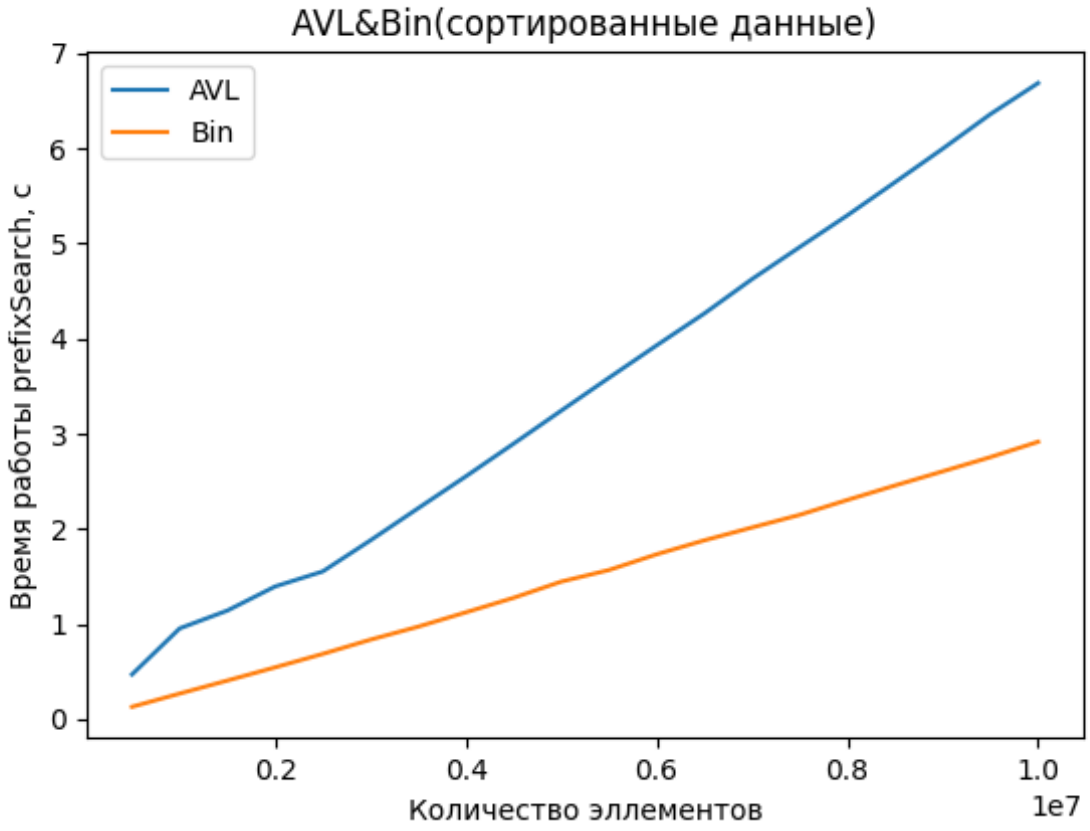

Рис. 4: Сравнение AVL и бинарного дерева(сортированные данные)

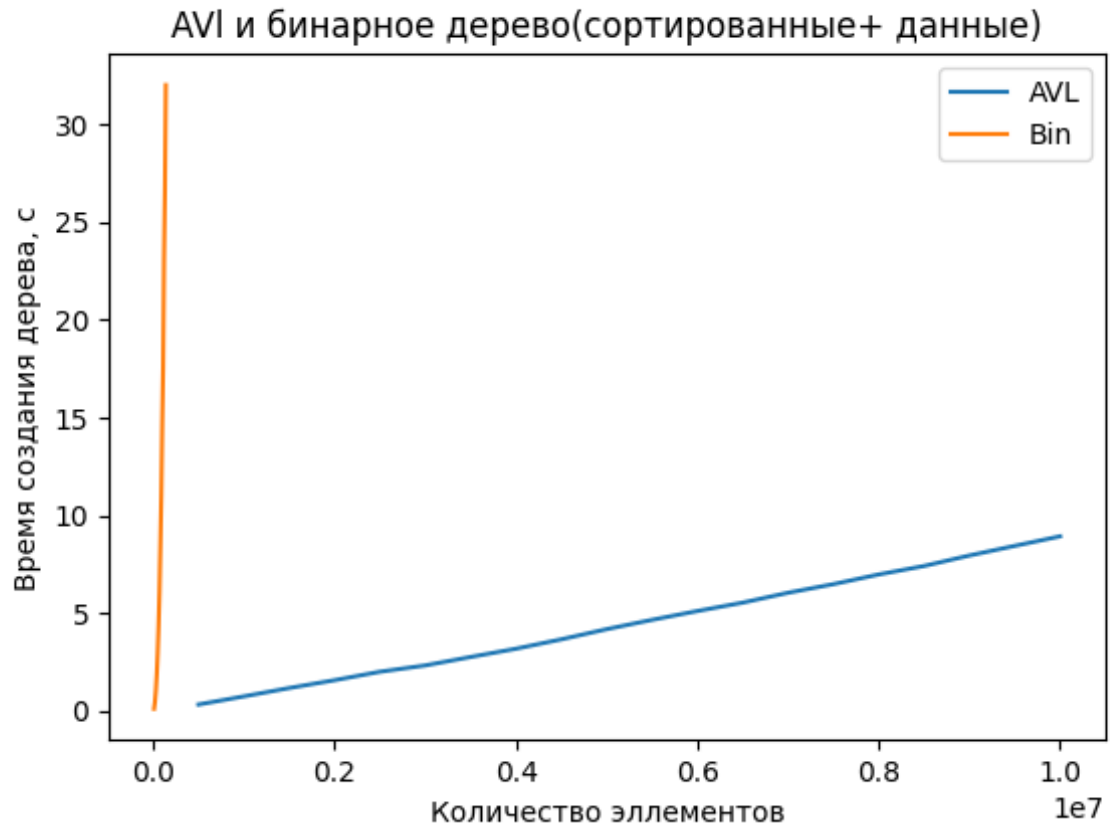

Рис. 5: Сравнение AVL и бинарного дерева(сортированные+ данные)

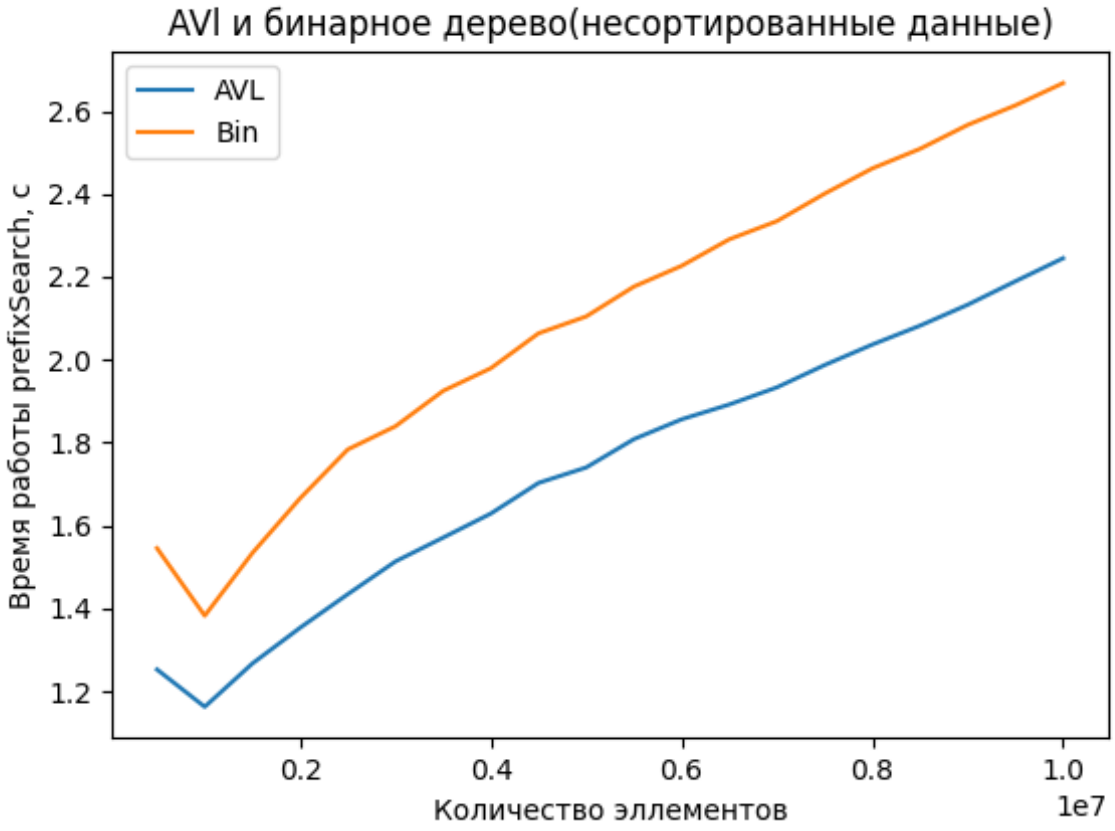

Рис. 6: Сравнение поиска (1000000 эллементов) AVL и бинарного дерева(несортированные данные)

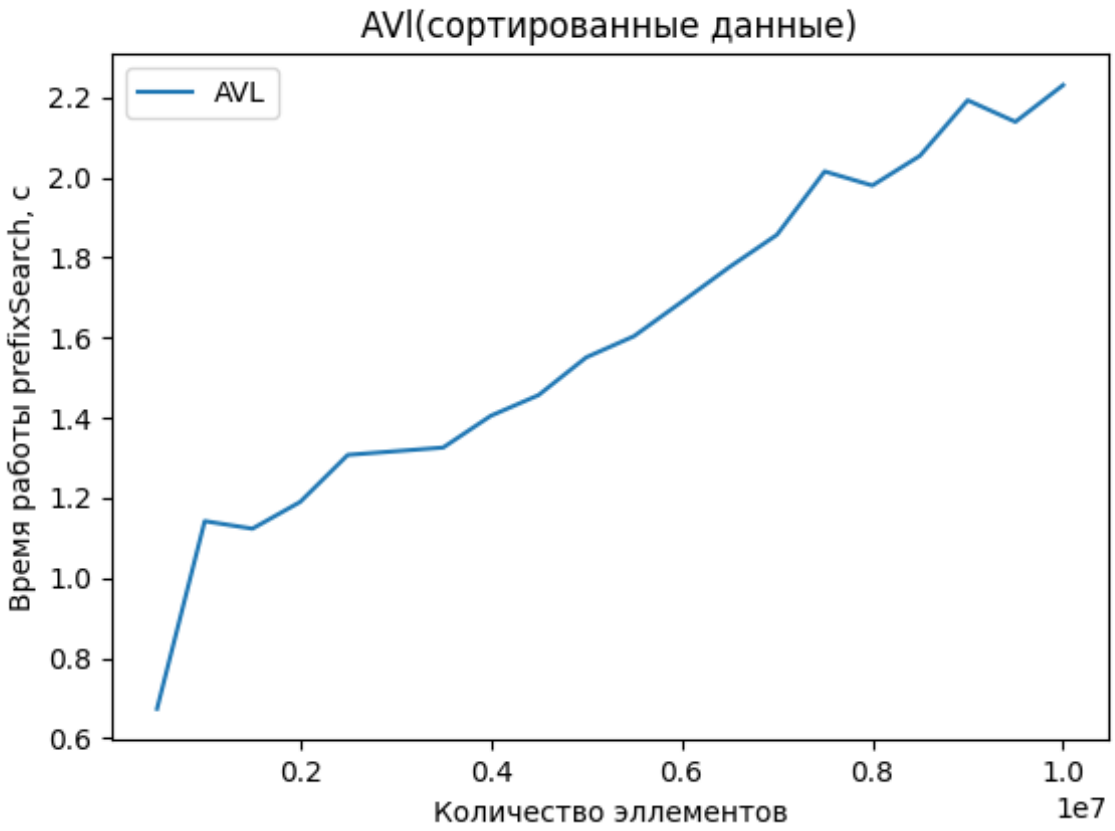

Рис. 7: поиск в AVL 1000000 эллементов (сортированные+ данные)

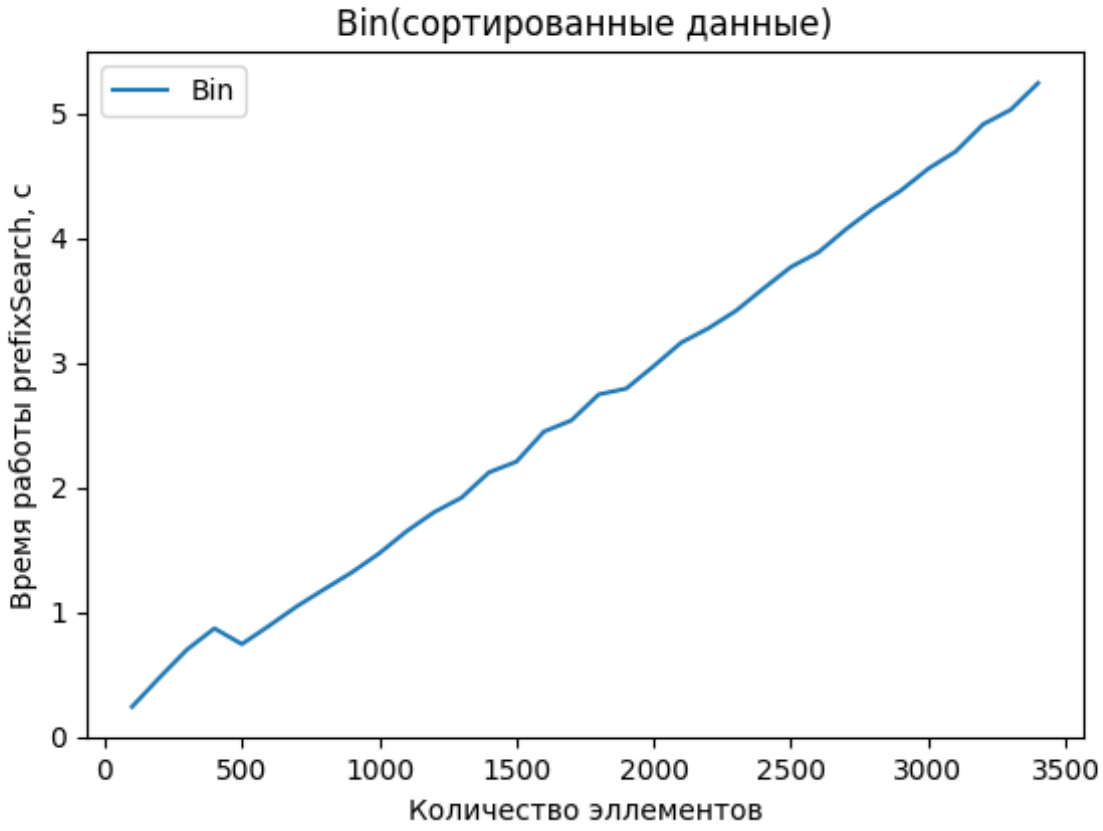

Рис. 8: поиск в Bin 1000000 эллементов (сортированные+ данные)