Давиденко Алексей Ильич, БПИ214

КДЗ №2 по курсу “Построение и анализ алгоритмов”

Отчёт

Состав файлов

Все алгоритмы поиска, вспомогательные функции, генерации текстов и замеры времени алгоритмов приведены в единственном main.cpp. Архив содержит данный отчёт, 4 текстовых файла с текстами для поиска, 20 .csv файла (замеры времени для каждого из видов текстов и количества подстановочных символов) и файл .ipynb со всеми графиками (и кодом построения). Все файлы, в том числе и этот отчёт находятся в репозитории.
https://github.com/davalex2003/construction-and-analysis-of-algorithms/tree/main/kdz_2

Ход работы

Сначала я прописал код, который устанавливает seed для рандома в зависимости от текущего времени. Затем я сгенерировал 4 текста (бинарный и днк алфавиты размерами 10^4 и 10^5 символов) и записал их в соответствующие файлы, имеющие название алфавит_размер.txt. Затем я прописал сами алгоритмы поиска: Brute Force алгоритм, алгоритм Кнута-Морриса-Пратта с обычными гранями и уточненными и алгоритм Рабина-Карпа (про него будет рассказано ниже). Затем с помощью вложенного цикла я провел замер времени для всех текстов и размеров строк без подстановок и с помощью второго цикла последовательное добавление во все строки от 1 до 4 символов подстановки. Все замеры проводились 100 раз с последующим усреднением для более плавных графиков.

Алгоритм Рабина-Карпа

Суть алгоритма заключается в подборе удобной хэш функции, с помощью которой будет возможно добавлять символ к строке, для которой уже посчитан хэш и не пересчитывать этот хэш полностью. Сама хэш функция выглядит так: умножаем существующий хэш на размер алфавита и прибавляем к нему значение нового символа. Чтобы избежать переполнение, все операции проводятся по модулю 9973 (большое простое число). Алгоритм выглядит так: просчитываем хэш подстроки и хэш текста от 0 символа размером в нашу подстроку. Затем сверяем их, если они совпали, то возможно, что наша строка является подстрокой текста. Для того, чтобы посчитать хэш новой строки, мы изначально считаем значение равное размер алфавита в степени размер строки и при добавлении символа вычитаем из существующего хэша новый символ умноженный на это значение и прибавляем наш модуль 9973. Из-за существования коллизий, требуется написать дополнительную функцию которая линейно проверяет действительно ли наша строка является подстрокой или у них просто совпали хэши.

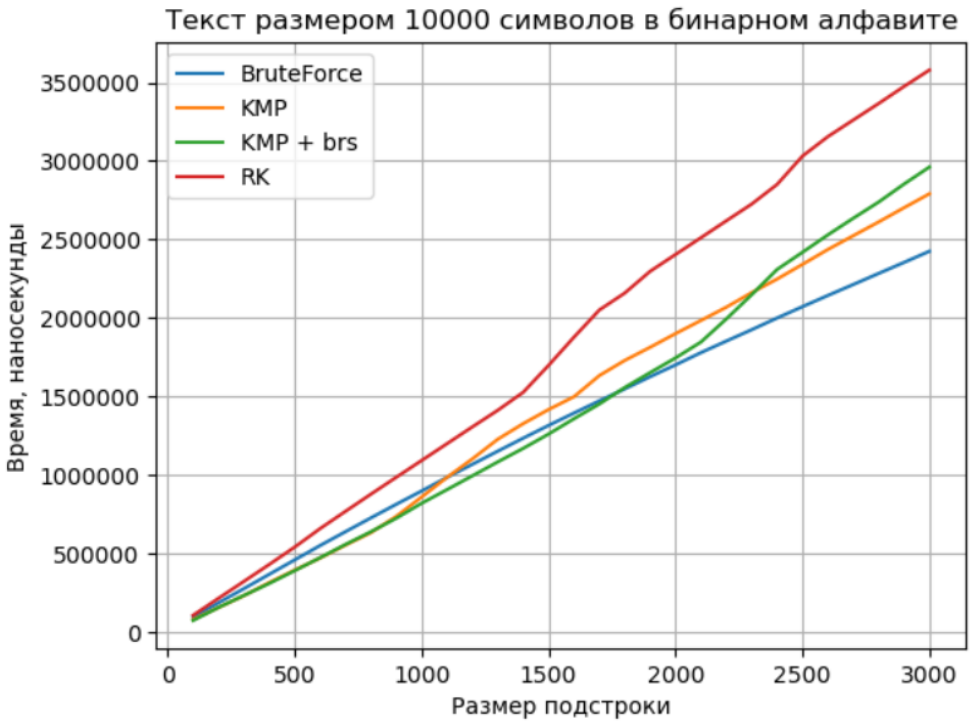
В лучшем случае - O(n), худшие случаи скатываются до O(mn), где n - размер текста, m - размер искомой подстроки.

Работа с символами подстановки

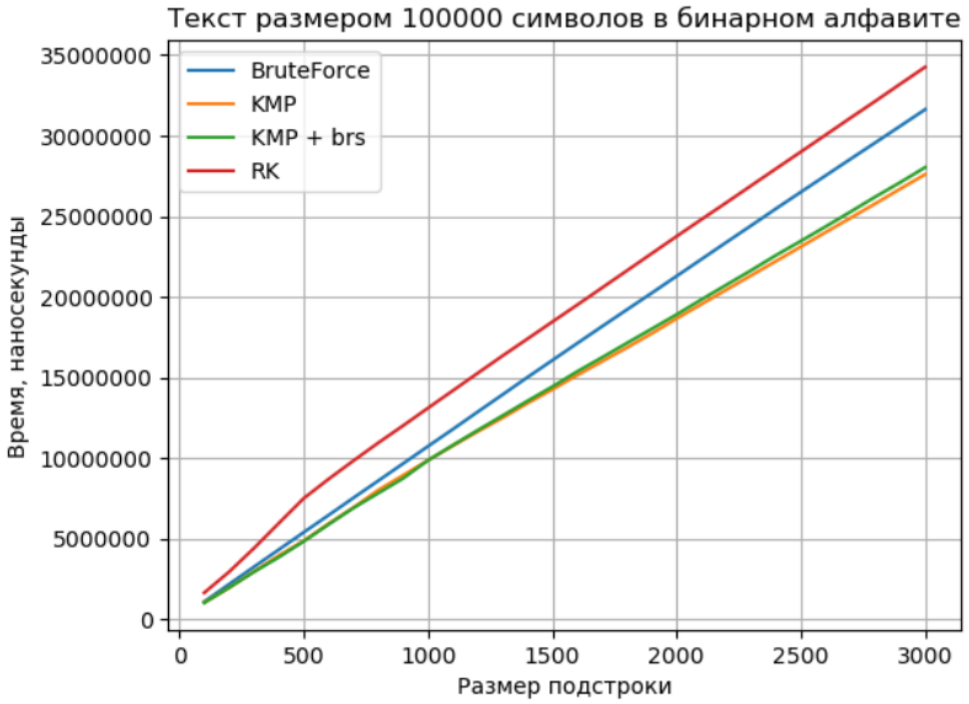
Все алгоритмы были модернизированы для работы с символами подстановки. Для полного перебора просто добавляем условие на равенство символа подстановки любому другому символу. Для обеих разновидностей алгоритма Кнута-Морриса-Пратта во время работы префикс функции предполагаем, что символ подстановки не равен любому другому символу для нахождения всех возможных подстрок, а также, что символ подстановки равен любому другому символу уже во время сравнения подстроки с текстом. Для алгоритма Рабина-Карпа дополнительно передаем в функцию размер алфавита (2 или 4 по заданию). Вызывается дополнительная функция, которая генерирует все возможные строки, количество которых равно размер алфавита в степени количество символов подстановки. Затем создается не один хэш, а массив хэшей и хэш части текста сравнивается со всеми возможными хэшами. Функция, которая исключает коллизии, также рассматриваем равенство знака подстановки любому другому символу.

Полученные графики

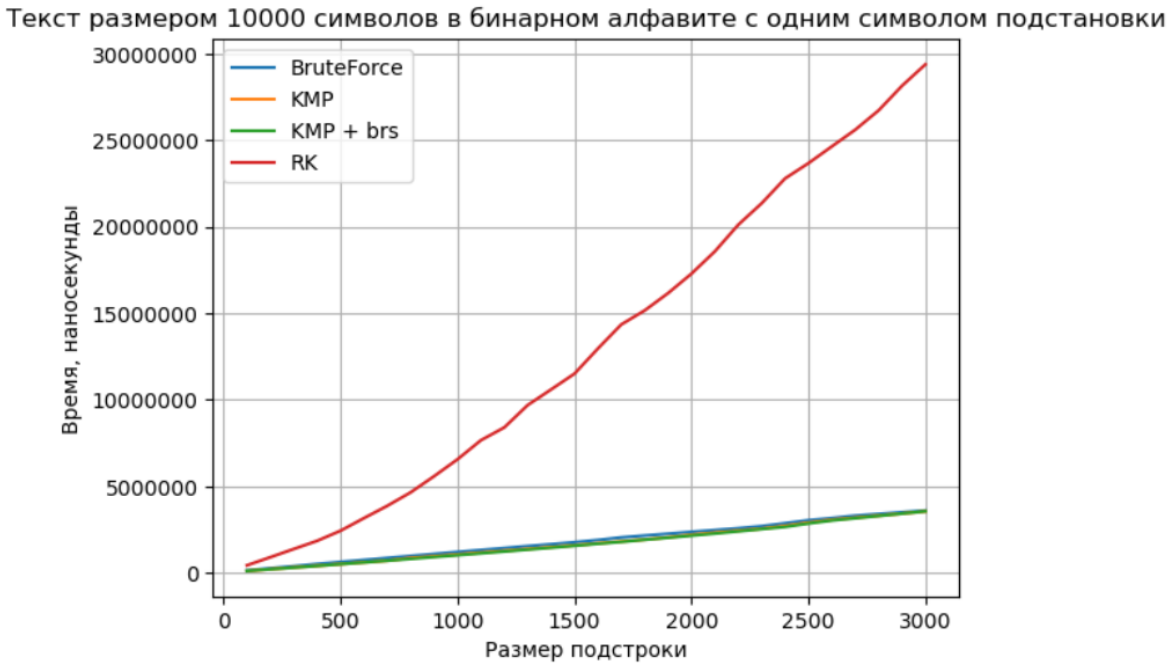
Результаты замеров времени выводились в csv файлы с названиями вида алфавит_размер_подстановка.csv. Графики строились с помощью Python и Jupyter Notebook. Достаточно было для всех 20 файлов считать значения и построить графики. Далее будут приведены только отличающиеся графики, все графики можно увидеть в файле БПИ214_Давиденко_Алексей_Ильич.ipynb.



Первый график уже показывает плохое время для алгоритма Рабина - Карпа и превосходство наивного алгоритма над обоими алгоритмами КМП.

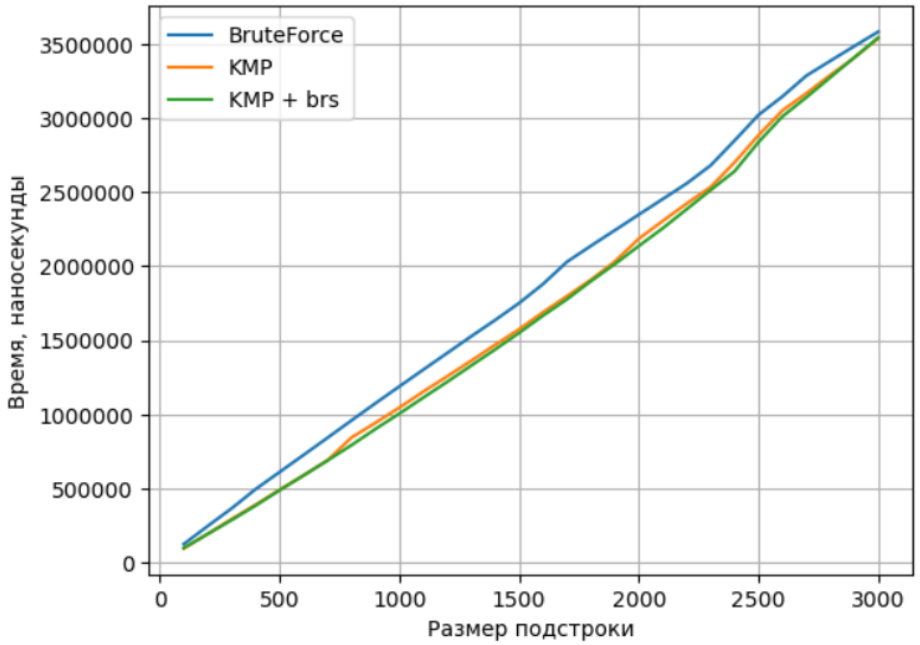


Увеличиваем размер текста и замечаем уже более логичную картину, которая подтверждается и 4-буквенным алфавитом. КМП-алгоритмы начинают превосходить наивный алгоритм.



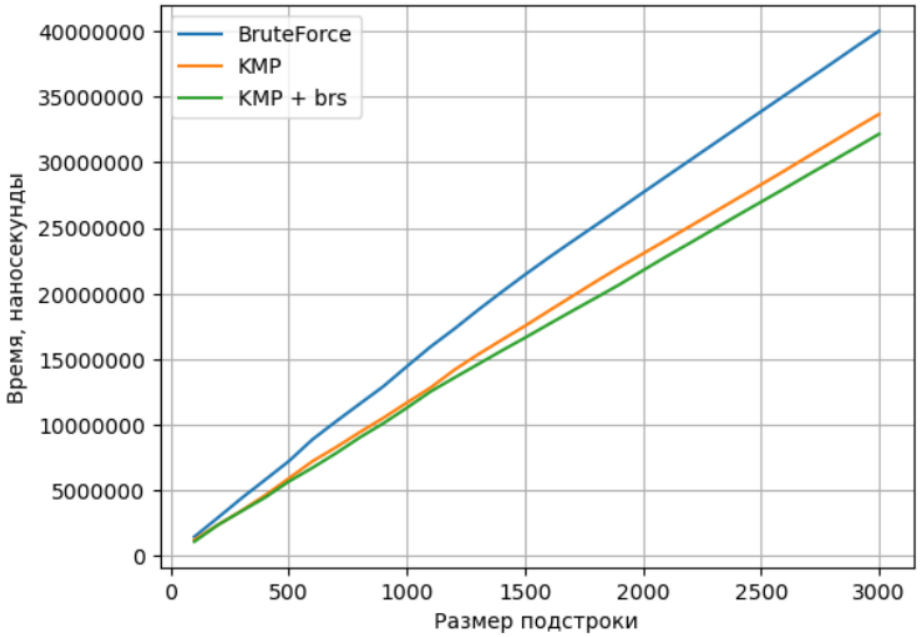
При появлении символов подстановки, алгоритм Рабина - Карпа показывает очень плохое время, поэтому отсутствует на следующих графиках с подстановками для того, чтобы увидеть разницу между наивным алгоритмом и КМП - алгоритмами.

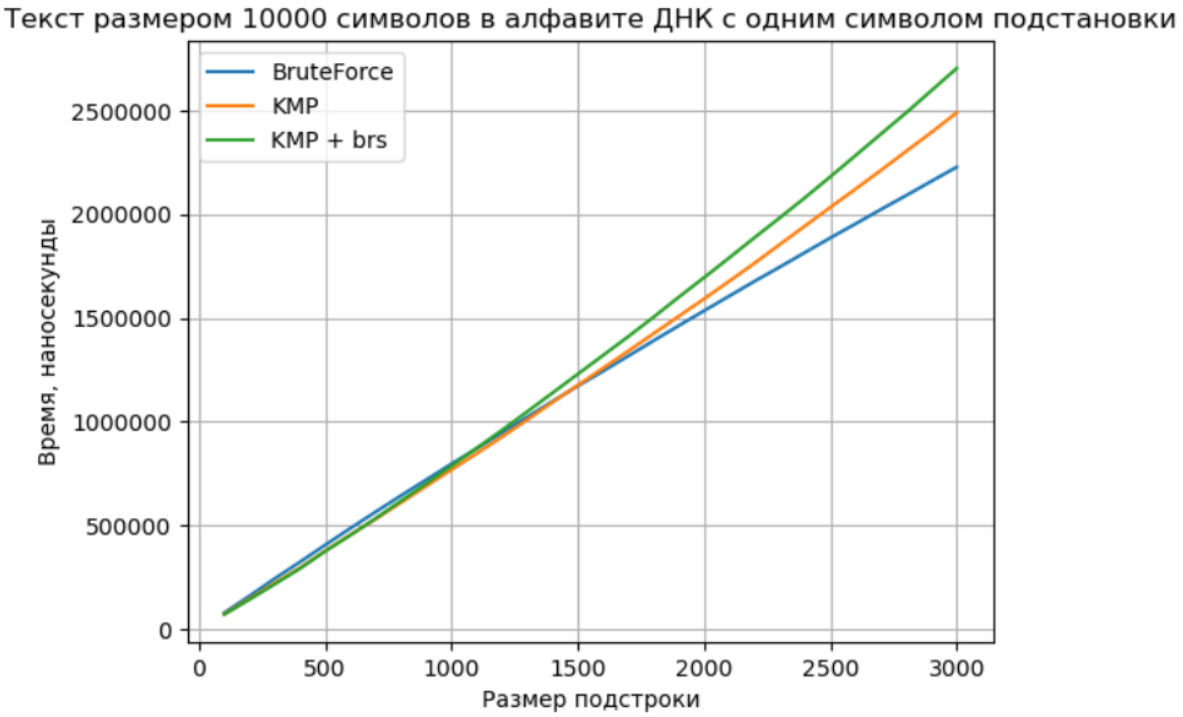
Текст размером 10000 символов в бинарном алфавите с одним символом подстановки

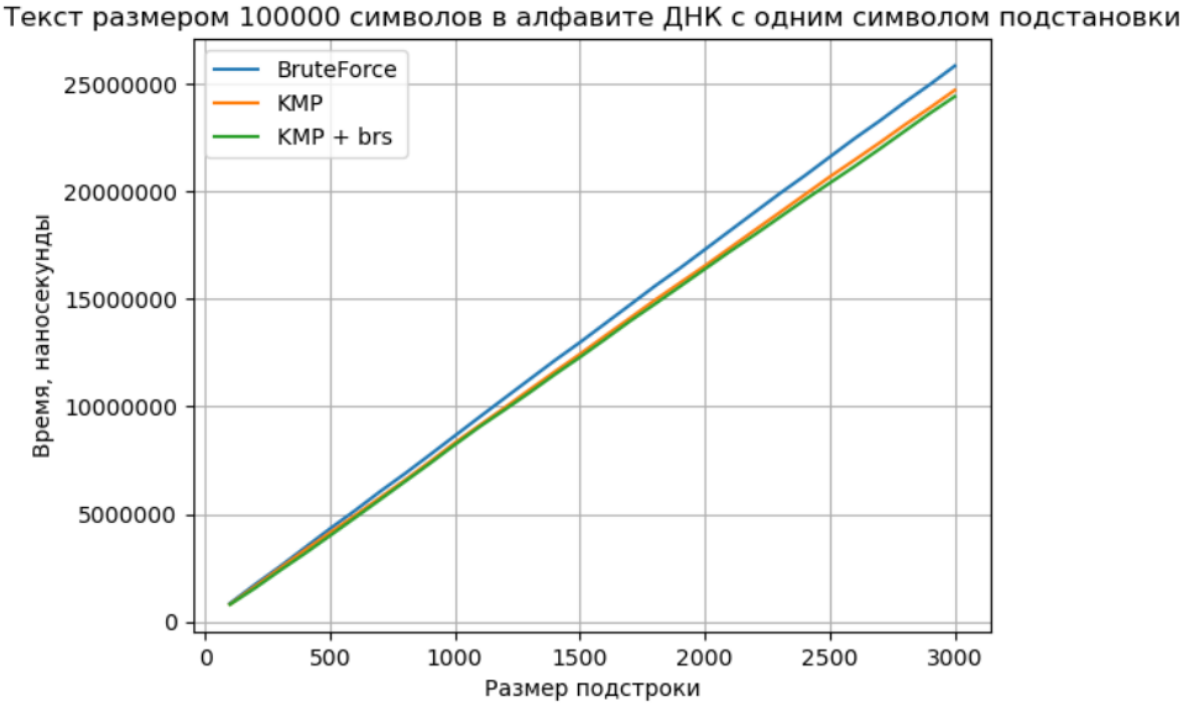


Несмотря на то, что с появлением символов подстановки, КМП - алгоритмы скатываются к наивному, они всё равно показывают чуть лучшее время.

Текст размером 100000 символов в бинарном алфавите с одним символом подстановки







Вывод

Начнем с алгоритма Рабина - Карпа. Поскольку он использует хэш функцию, то в основном предназначен для поиска сразу нескольких строк в тексте, поэтому проигрывает и наивному алгоритму. С появлением символов подстановки из-за необходимости генерации всех возможных строк показывает очень большое время.

Можно заметить общую тенденцию: с меньшим размером текста (как с символами подстановки, так и без них) наивный алгоритм чаще всего показывает

лучшее время, чем КМП - алгоритмы. Это происходит из-за необходимости (в случае КМП) сначала обработать строку и только затем искать все ее вхождения. С увеличением размера текста, КМП - алгоритмы начинают превосходить наивный алгоритм, так как нагоняют упущенное время на обработку строки из-за бОльшего размера текста.

Разница между обычным КМП - алгоритмом и оптимизированным с помощью уточненных граней на графиках почти отсутствует из-за небольшого размера всех текстов. Время, потраченное на запуск еще одной префикс функции превосходит выигрышное время от пропуска некоторых индексов в самом тексте.

С появлением символов подстановки, КМП - алгоритмы скатываются к наивному алгоритму. Это происходит из-за принципа работы префикс-функции. С нахождением первого знака подстановки, она в дальнейшем будет выдавать только нули, поэтому оптимизация зависит от символов до первого знака подстановки.