Лабораторная работа № 4 по курсу дискретного анализа: Поиск образца

Выполнил студент МАИ группы М8О-201Б *Ефимов Александр.*

Постановка задачи

Необходимо реализовать один из стандартных алгоритмов поиска образцов для указанного алфавита.
Вариант алгоритма: Поиск одного образца при помощи алгоритма Кнута-Морриса-Пратта.
Вариант алфавита: Слова не более 16 знаков латинского алфавита (регистронезависимые)

Метод решения

В соответствии с требованиями необходима реализация следующих функций

1. Приведение заглавных букв к строчным
2. Препроцессорная обработка
3. Сам поиск

Приведение заглавных букв к строчным происходит с помощью встроенных в язык алгоритмов.

Перед поиском необходимо прочитать строку, содержащую образец. Так как размер этого образца неизвестен, этот образец следует занести в вектор. Следующий этап – препроцессорная обработка образца с теперь известным размером.

Препроцессорная обработка будет проводиться с помощью Z-функции, после чего последняя будет преобразована в суффикс-префиксный массив (в соответствии с алгоритмами, описанными в книге Algorithms on Strings, Trees, and Sequences - Dan Gusfield).

Процесс самого поиска разбит на два этапа: в первом этапе текст считывается посимвольно и заполняется буфер слов размером, равным размеру образца. Если считанный символ - пробел или символ табуляции, то счетчик слов увеличивается (если слово состоит как минимум из одной буквы - повторяющиеся пробелы игнорируются), и начинается запись нового слова. Если считанный символ - символ новой строки, то в созданный дек заносится число слов на одной строке, а счетчик слов сбрасывается. Любой другой символ заносится в текущее слово буфера.

Между первым и вторым этапом сравниваются значения дека, содержащего размеры строк, и переменной, содержащей номер начального слова текущего буфера. Пока значение номера начального слова больше значения в деке, то последние вычитаются из этого номера, параллельно выталкиваясь, до тех пор, пока первое число в деке меньше номера первого слова (т.е. если номер больше значений, то при чтении текста начало

образца перешло на новую строку). За каждое вытолкнутое значение счетчик строк увеличивается на 1.

На втором этапе происходит сам поиск КМП в соответствии с алгоритмом, но, кроме удаления первых элементов при совпадении и несовпадении, увеличивается номер первого слова образца на размер сдвига.

Описание программы

1. **CircleBuffer.h** Кольцевой буфер. Нужен для хранения текста. В основной программе его размер равен размеру образца. Его использование позволяет сохранить память и время, так как он статичен.

2. **main.cpp** Главная программа, содержащая поиск КМП.

   - *ToLower* – функция, переводящая заглавные буквы строки в строчные.
   - *RetrieveKey* – функция, возвращающая вектор слов, написанных на одной строке (он и будет содержать образец).
   - *Zfunc* и *BuildSP* – функции, строящие по образцу сначала Z-массив, а затем в суффиксно-префиксный массив.

   После того, как образец был считан и Z- и SP-массивы были построены, начинается сам поиск в цикле, который идет до достижения конца файла. Используемые переменные:

   - *iter* – индекс, указывающий на позицию текущего сравниваемого слова.
   - *lineNum* – номер строки, на которой находится текущее начало образца.
   - *wordNum* – номер слова, на которой находится текущее начало образца.
   - *currentWord* – номер текущего читаемого слова.
   - *buff* – строка, в которой накапливаются прочитанные буквы.
   - *text* – статическая очередь, равная размеру образца. Накапливает слова для последующего сравнения их с образцом.
   - *newLineWordNum* – размеры строк, находящихся между началом образца и текущего читаемого слова.

Дневник отладки

Номер	Ошибка	Обнаруженная причина	Исправление
1	Ошибка выполнения	Использование неинициализированного суффикс-префиксного массива	Занулить массив перед его использованием
2	Неправильный ответ	Логическая ошибка построения Z функции	Строка 58 – заменить pat[zLeft] на pat[zRight]

Тест производительности

На рисунках 1-3 показаны время поиска в текстах, в которых соответственно – нет совпадений; совпадения расположены случайным образом; весь текст состоит из образца. Полученные результаты совпадают с заявленной линейной сложность, хотя и с разными константами.

Выводы

Алгоритм КМП читает каждую букву всего один раз, и таким образом, быстро проходит тексты больших размеров. Его можно использовать, как поисковик в текстовых редакторах. Однако, стандартно, он не поддерживает поиск регулярных выражений.

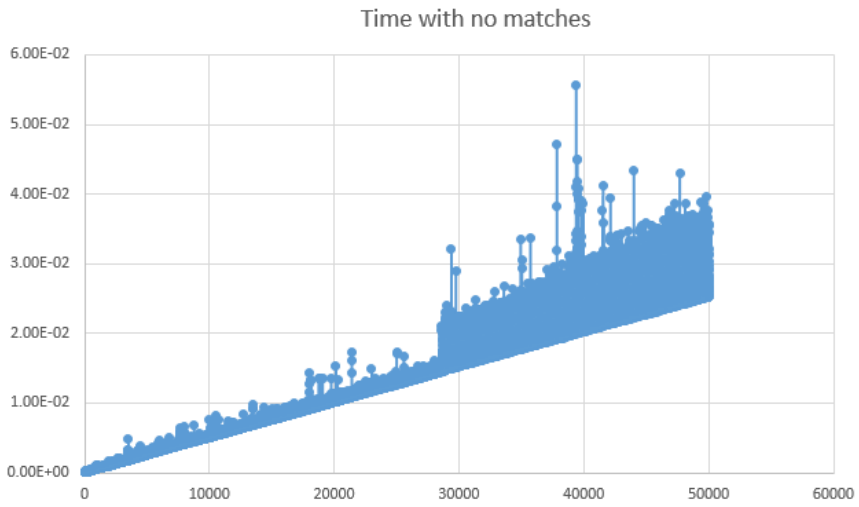

Рис. 1: Поиск без совпадений

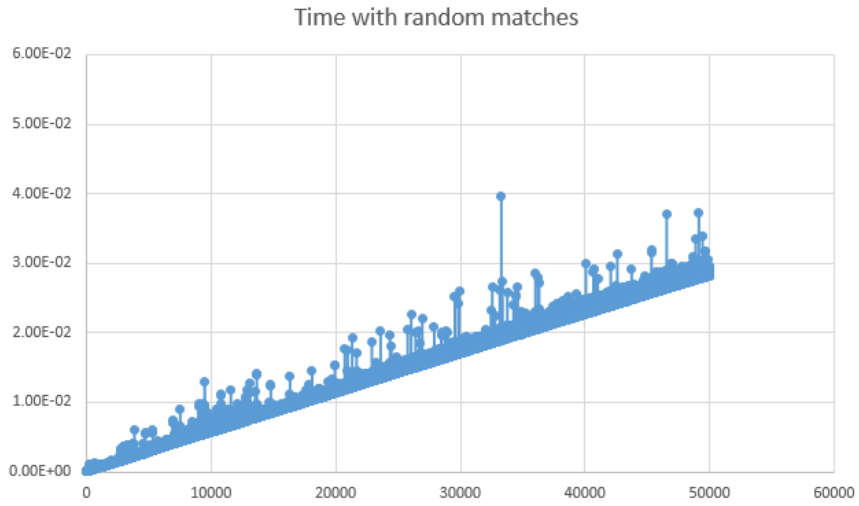

Рис. 2: Поиск со случайными совпадениями

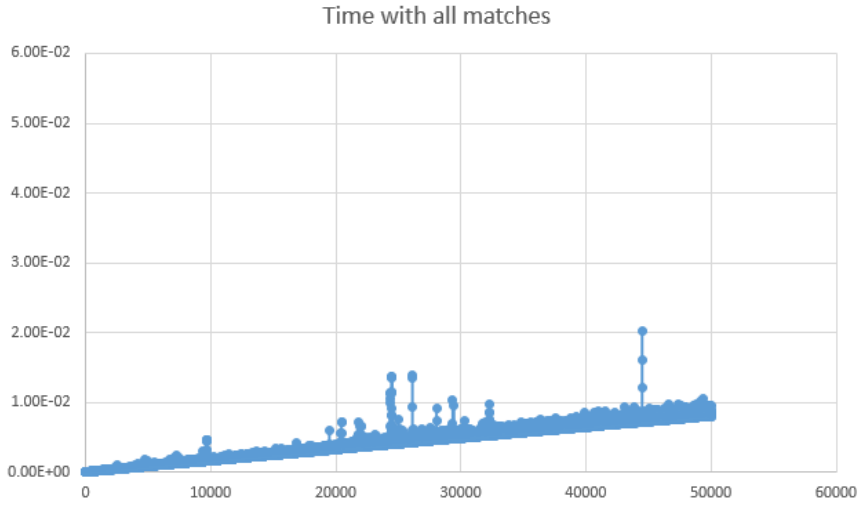

Рис. 3: Поиск со всеми совпадениями