Лабораторная работа № 4 по курсу Дискретный Анализ. Строковые алгоритмы

Выполнил студент группы М8О-207Б-21 МАИ *Друхольский Александр.*

Условие

Кратко описывается задача:

1. Необходимо реализовать один из стандартных алгоритмов поиска образцов для указанного алфавита.

2. Вариант задания 4-1: Поиск одного образца основанный на построении Z-блоков. Слова не более 16 знаков латинского алфавита (регистронезависимые).

Метод решения

В основу алгоритма легло построение z-блоков. Z-блоком назовем подстроку с началом в позиции i и длиной Z[i]. Для работы алгоритма заведём две переменные: left и right — начало и конец Z-блока строки S с max right (если таких несколько, выбирается наибольший). Изначально left=0 и right=0. Пусть нам известны значения Z-функции от 0 до i1. Найдём Z[i]. Рассмотрим два случая.

1. $i > right$: пробегаемся по строке S и сравниваем символы на позициях S[i+j] и S[j].Пусть j первая позиция в строке S для которой не выполняется равенство S[i+j]=S[j], тогда j - Z-функция для позиции i. Тогда left=i,right=i+j1. Это наивный алгоритм, поэтому Z-функция будет определена корректно

2. $i <= right$: Сравним Z[ileft]+i и right. Если right меньше, то надо наивным алгоритмом пробежаться по строке начиная с позиции right и вычислить значение Z[i]. Иначе мы уже знаем верное значение Z[i], так как оно равно значению Z[ileft].

Чтобы искать в тексте подстроку с помощью z-функции осуществляем конкатенацию нашего паттерна и текста, разделяя их специальным символом. Далее ищем z-функцию получившейся строки. С поцизий, в которых Z[i] == |S|, происходит совпадение паттерна и подстроки текста. Сложность алгоритма O(|S|), так как каждая позиция пробегается не более двух раз: при попадании в диапазон left right и при подсчитывании z-функции наивным алгоритмом.

Описание программы

Основные моменты:

1. *char ConvertToLow(char c)* - вспомогательная функция, чтобы обеспечить регистронезависимость строк

2. $int* \ ZFunc(string \ pattern)$ - строит z-функцию входной строки при помощи алгоритма построения z-блоков

3. $void \ SearchZ(string \ pattern, \ string \ text, \ vector < int > cntwords)$ - осуществляет работу всего алгоритма. pattern - образец, text - текст, в которым мы ищем, cntwords - это массив с количеством слов для каждой строки (для ориентации по тексту, который мы преобразовали в одну строку с пробелами-разделителями)

Дневник отладки

WA - 1 Отправил неверную версию кода с лишними выводами.

WA - 5 Ошибка с пустыми строками и разделителями

WA - 7 Оказывается по условию надо было делать только для одного паттерна, что я не увидел и поэтому иногда выводило не то, что надо.

Тест производительности

Замерялось только время работы алгоритма. Во втором столбце количество строк в тексте.

№	Кол-во строк	Время, с
1	10	0.000017
2	100	0.000045
3	1000	0.001725
4	10000	0.012574
5	100000	0.117933
6	1000000	1.102949

Каждый последующий тест больше предыдущего в 10 раз, что делает результат замера более наглядным. Как мы видим, время увеличивается +- линейно.

Выводы

Я познакомился с эффективным алгоритмом z-функции, который хоть и не является лучшим при поиске подстроки в строке, но всё равно является очень полезным и эффективным. Теория касательно алгоритмов бралась с лекций и интернет ресурсов. Замеры времени показали, что время работы реализованного алгоритма действительно линейно зависит от размера входных данных.