Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу
«Дискретный анализ»

Строковые алгоритмы

Студент: Эсмедляев Федор Романович
Группа: М8О–212Б–22
Преподаватель: Н.Д.Глушин
Оценка:____
Дата:_____
Подпись:_____

Москва, 2024.

Вариант 2-1

Необходимо реализовать один из стандартных алгоритмов поиска образцов для указанного алфавита.
Вариант алгоритма: Поиск одного образца при помощи алгоритма Бойера-Мура.

Вариант алфавита: Слова не более 16 знаков латинского алфавита (регистронезависимые).

Запрещается реализовывать алгоритмы на алфавитах меньшей размерности, чем указано в задании.

Формат ввода

Искомый образец задаётся на первой строке входного файла.
В случае, если в задании требуется найти несколько образцов, они задаются по одному на строку вплоть до пустой строки.

Затем следует текст, состоящий из слов или чисел, в котором нужно найти заданные образцы.

Никаких ограничений на длину строк, равно как и на количество слов или чисел в них, не накладывается.

Формат вывода

В выходной файл нужно вывести информацию о всех вхождениях искомых образцов в обрабатываемый текст: по одному вхождению на строку.
Для заданий, в которых требуется найти только один образец, следует вывести два числа через запятую: номер строки и номер слова в строке, с которого начинается найденный образец. В заданиях с большим количеством образцов, на каждое вхождение нужно вывести три числа через запятую: номер строки; номер слова в строке, с которого начинается найденный образец; порядковый номер образца.

Нумерация начинается с единицы. Номер строки в тексте должен отсчитываться от его реального начала (то есть, без учёта строк, занятых образцами).

Порядок следования вхождений образцов несущественен.

Пример

Ввод	Вывод
cat dog cat dog bird CAT dog CaT Dog Cat DOG bird CAT dog cat dog bird	1, 3 1, 8

Метод решения

Изначально мы ставим наш индекс на конец паттерна и после этого начинаем сравнивать наше слово из текста и паттерн **справа на лево**, это удобно потому что если будет не совпадение мы не будем проверять оставшуюся часть паттерна, а просто выполним сдвиг, который определяется как max(ППС, ПХС, 1).
ППС – правило плохого символа. Оно заключается в нахождении в паттерне такого же слова и записи расстояния от края паттерна до последнего вхождения каждого слова. Есть сильное ППС, оно предлагает сохранять не только последнее вхождение, но и все предыдущие вхождения тоже.
ПХС – правило хорошего суффикса. Для начала найдем аналогично Z-функции(*длина максимального префикса подстроки, начинающейся с позиции* x *в строке* S*, который одновременно является и префиксом всей строки* S) максимальный суффикс. С помощью получившегося массива значений строим массив Li, который будет обозначать для каждой подстроки, где есть такая же подстрока, только с другим символом спереди. Этот массив поможет нам двигать на сразу нужную позицию, зная различие в символах паттерна и текста.

Сложность:
Средняя сложность O(n+m), где n – длина текста, m – длина паттерна
Максимальная сложность O(n*m) , это возможно достигнуть если очень большая частота совпадений.

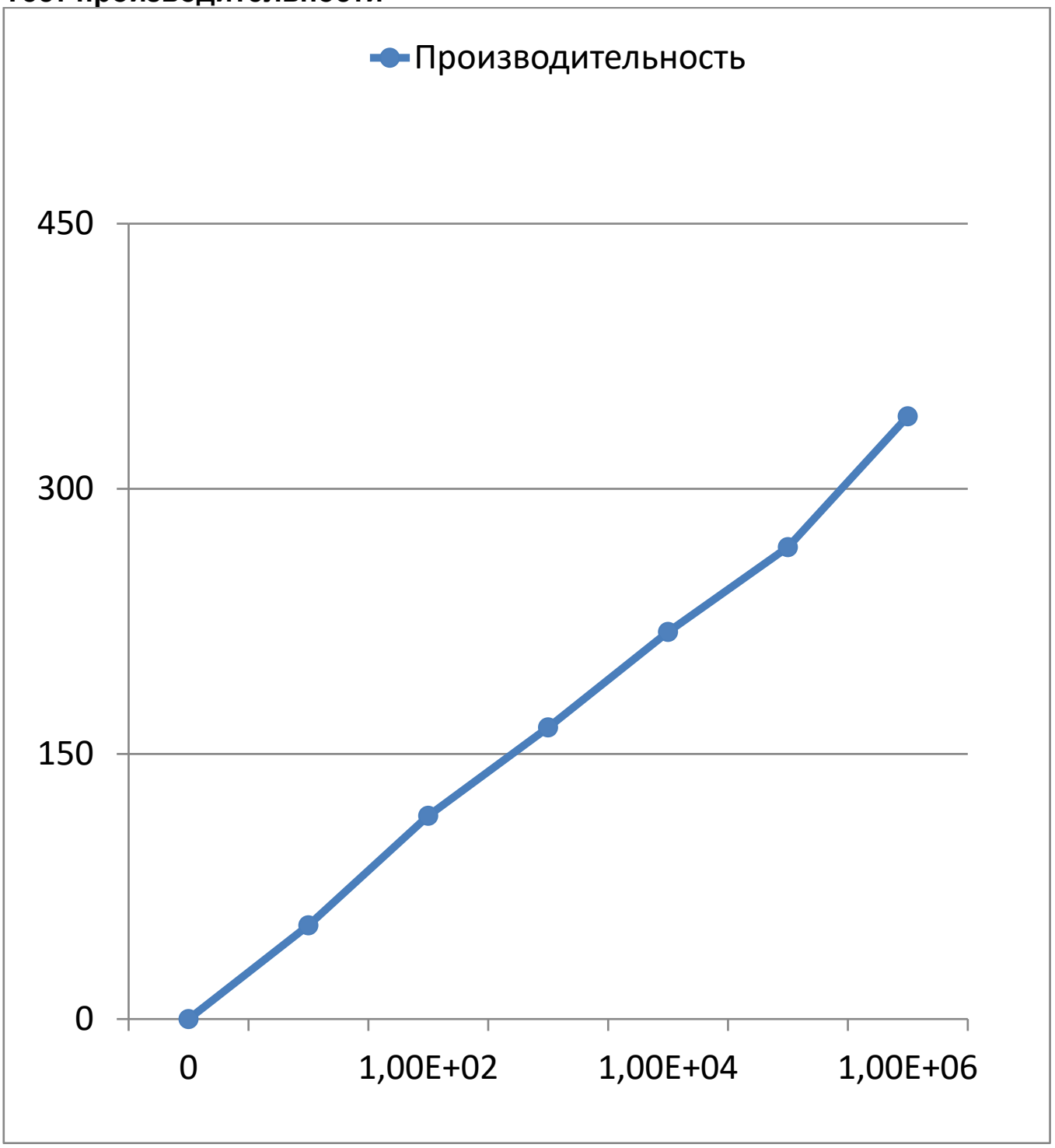
Описание программы

1)Разделяем вводимый паттерн на слова и сохраняем в отдельный вектор строк
2)Разделяем текст на слова + запоминаем их позицию в строке и на какой строке встречалось слово
3)Делаем вектор ППС
4)Делаем вектор ПХС:
    а) Z-функция для суффиксов
    б) формирование массива I
    в) получение финального массива ПХС(или Li)
5)Проходимся в цикле по всему тексту и сравниваем с паттерном с конца. И изменяем переменную обозначающее место сравнения на max(ППС, ПХС, 1)

Дневник отладки

1)Начальная реализация без чтения и записи позиций слов
2)Реализация с чтением и запоминанием позиций
3)Добавление вспомогательной функции I, как в книге Гасфилда
4)Исправление проблем с +1 -1 в различных местах

Тест производительности



Вывод:
Я реализовал функцию поиска слова в тексте с помощью алгоритма Бойера-Мура. Данная функция немного подводит по времени если много совпадений и много маленьких слов. Алгоритм Бойера-Мура считается наиболее эффективным алгоритмом поиска шаблонов в стандартных приложениях и командах, таких как Ctrl+F в браузерах и текстовых редакторах.