

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу
«Дискретный анализ»**

Суффиксные деревья

Студент: Эсмедляев Федор Романович

Группа: М8О–312Б–22

Преподаватель: Н.Д.Глушин

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2024.

Вариант 2

С. 2 Поиск с использованием суффиксного массива

Ограничение времени	15 секунд
Ограничение памяти	1Gb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Найти в заранее известном тексте поступающие на вход образцы с использованием суффиксного массива.


Формат ввода

Текст располагается на первой строке, затем, до конца файла, следуют строки с образцами.


Формат вывода

Для каждого образца, найденного в тексте, нужно распечатать строку, начинающуюся с последовательного номера этого образца и двоеточия, за которым, через запятую, нужно перечислить номера позиций, где встречается образец в порядке возрастания.

Пример

Ввод 

```
abcdabc  
abcd  
bcd  
bc
```

Вывод 

```
1: 1  
2: 2  
3: 2, 6
```

Метод решения

Добавим к тексту дополнительный символ \$. Далее отсортируем строку по буквам в алфавитном порядке, но с запоминанием индекса в начальной строке в массив `idx`. Далее создадим массив `eq`, в который будет писать значение бакета, в который относится символ, если символы отличаются, то значение бакета увеличиваем на 1, но для первого символа всегда 0. Далее каждого индекса отнимаем степень

двойки, но по модулю длины строки, т.е. при получении -1 , берем значение длины строки, на первом шаге это 0 , т.е. отнимаем $2^0 = 1$. Сортируем получившийся массив, через сортировку подсчетом, потому что она 1) за $O(n)$, 2) устойчивая. После этого в соответствие каждому значению `idx` восстанавливаем с прошлого шага значение `eq`. Такой алгоритм повторяем до того момента, когда все бакеты будут содержать всего 1 элемент, т.е. будут числа от 0 до длины строки, получившийся массив `idx` будет суффиксным массивом. Далее создаем массив LCP, он заключается в нахождении для каждого суффикса сколько символов вначале совпадает с предыдущим. Далее с помощью этого массива мы сможем гораздо быстрее искать бинарным поиском.

Сложность: $O(N * \log N)$

Описание программы

1. Построение суффиксного массива (`suff_mas`):

- В функции `suff_mas` строится суффиксный массив, представляющий все суффиксы строки в отсортированном виде.
- Изначально символы строки сортируются лексикографически с помощью `getSortedIndices`. Далее массив `suff_m` заполняется структурами `suffix`, содержащими индексы начала суффиксов (`idx`) и их классы эквивалентности (`eq`).
- Сортировка выполняется с помощью `countingSort` до полного лексикографического упорядочения суффиксов.

2. Построение LCP массива (`buildLCPArray`):

- Функция `buildLCPArray` строит массив LCP, хранящий длины общих префиксов для каждой пары соседних суффиксов в `suff_mas`.
- `rank` содержит позиции суффиксов в массиве `suff_mas`, а `lcp` — длины префиксов.

3. Поиск подстроки (`searchPatternWithLCP`):

- Функция `searchPatternWithLCP` выполняет бинарный поиск позиции паттерна в суффиксном массиве `suff_mas`.
- Использует переменные `left`, `right`, `mid` для поиска, сравнивая подстроку с паттерном.
- Если совпадение найдено, все вхождения добавляются в `positions`.

4. Основная программа (`main`):

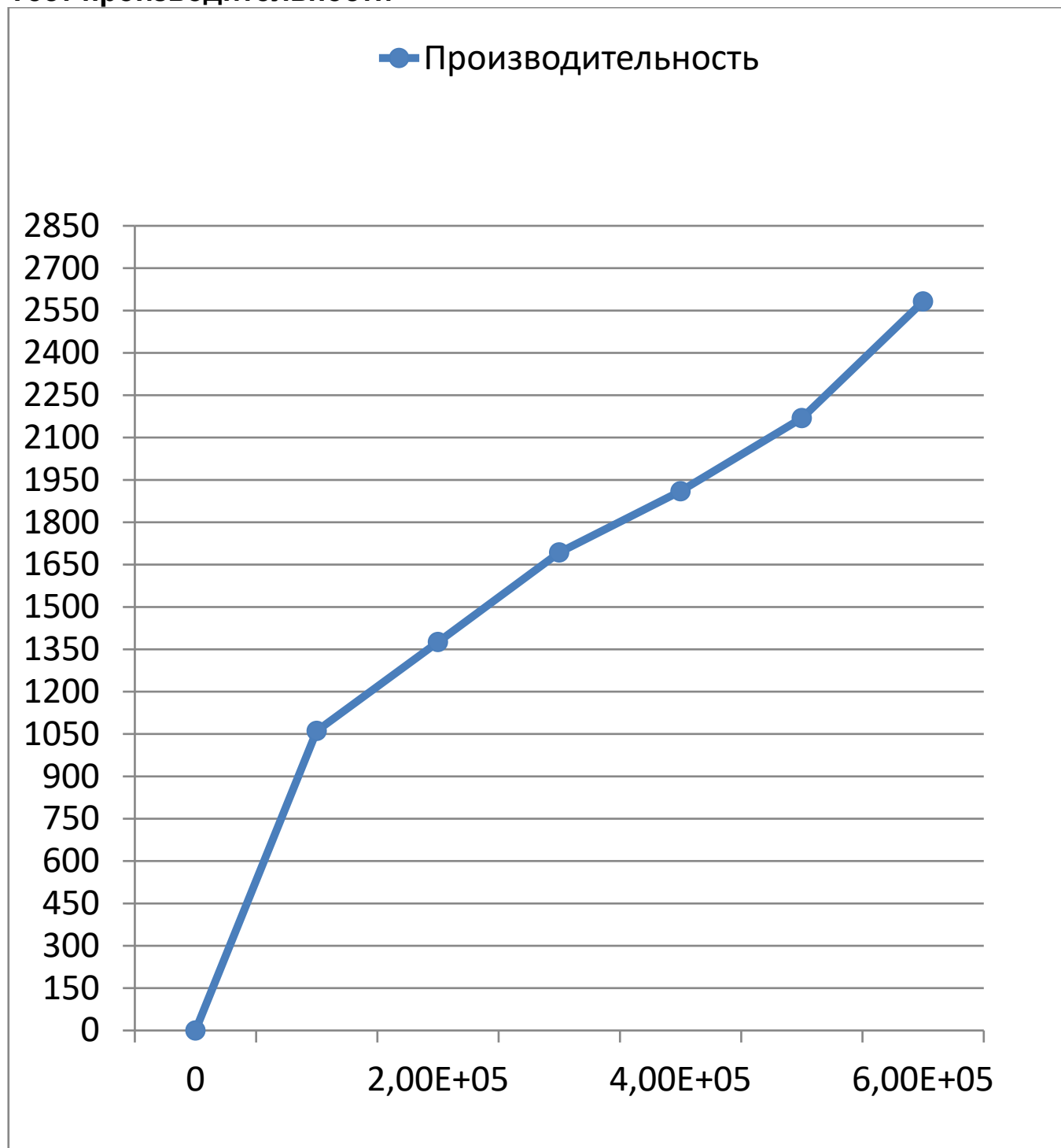
- Строится суффиксный массив `suff` и массив LCP `lcp` для строки `s`.
- Принимается каждый паттерн, и выполняется поиск с использованием `searchPatternWithLCP`.
- Найденные вхождения выводятся.

Дневник отладки

- 1) Готовый код, который имел WA на тесте 4.
- 2) Оказалось, что был просто неправильный вывод и надо было вынести переменную `kol` из `if` для вывода, просто в весь цикл, чтобы

каждый раз прибавлять 1. Итог - ОК

Тест производительности



Вывод:

Я научился строить суффиксный массив без использования алгоритма Уконнена(суффиксного дерева). Также смог использовать полученный массив и бинарный поиск для поиска паттерна в тексте.