Лабораторная работа № 5 по курсу дискретного анализа: Суффиксные деревья

Выполнил студент группы 08-308 МАИ *Попов Николай*.

Условие

Реализовать поиск подстрок в тексте с использование суффиксного дерева. Суффиксное дерево можно построить за $O(n^2)$ наивным методом.

Метод решения

В алгоритме вставки суффикса сначала ищем совпадение его первой буквы с дочерними элементами корня. Если совпадение отсутствует, вставляем суффикс от корня. При нахождении совпадения движемся по ребру до различия. Затем разделяем ребро, уменьшая его и создавая два новых: одно продолжает путь, другое несет оставшийся суффикс. Если достигли конца ребра без различий, ищем следующую букву среди детей, чтобы продолжить поиск, иначе вставляем суффикс от текущего положения.

Поиск паттерна схож с вставкой и прерывается при несовпадении или полном чтении паттерна. Для определения всех позиций вхождения паттерна проводим рекурсивный поиск от вершины, формируя массив индексов вхождений в текст.

Описание программы

Для хранения элемента была создана структура, состоящая из двух полей: ключа и значения.

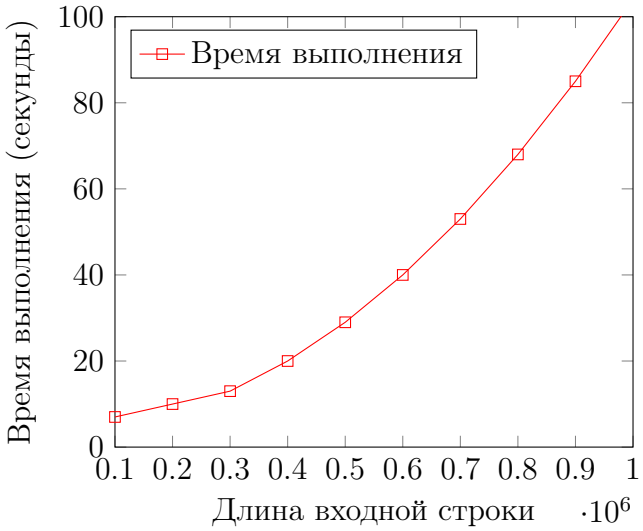
```
struct Node {
    int start;
    int end;
    int position;
    vector<Node*> next;

    Node(int, int, int);
    ~Node();
    void Insert(int, int);
    vector<int> Find(string);
    void AllPositions(vector<int>&);
    void Print(int);
};
```

Были реализованы функции вставки и удаления, использующие функции *Insert* и *Find AllPosistions*.

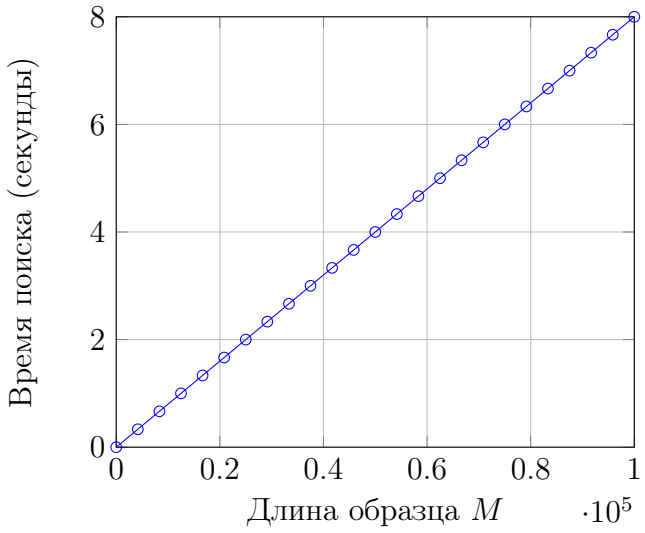
Тест производительности

Время построения суффиксного дерева в зависимости от длины входной строки



Оценка сложности: O(n^2) где, n - длина строки.

Зависимость времени поиска от длины образца



Оценка сложности: O(m) где, m - длина образца.

Выводы

В ходе выпопления данной лабораторной работы было реализовано суффиксное дерево с использованием наивного метода построения. Затем было выполнено задание по поиску всех вхождений паттернов в тексте, используя построенное суффиксное дерево.