Лабораторная работа № 5 по курсу дискретного анализа: Суффиксные деревья

Выполнил студент группы 08-208 МАИ *Левштанов Денис*.

Условие

1. Необходимо реализовать алгоритм Укконена построения суффиксного дерева за линейное время. Построив такое дерево для некоторых из входных строк, необходимо воспользоваться полученным суффиксным деревом для решения своего варианта задания.

   Алфавит строк: строчные буквы латинского алфавита (т.е., от a до z).

2. Поиск в известном тексте неизвестных заранее образцов

Метод решения

Считывается первая строка входных данных, добавляется к ней терминальный символ и строится суффиксное дерево по алгоритму Укконена. Затем производится поиск поступающих на вход слов в дереве. При этом в дереве воспроизводиться путь равный строке, если это удалось, то из текущей вершины обходятся все листы и выводятся их номера в порядке возрастания.

Описание программы

1. TSuffixTree.h(Объявление дерева и функций):

   map<char, TNode*> childs; - переходы

   TNode* suffLink; - суффиксная ссылка

   int start; - индекс начала строки

   int *end; - указатель на индекс конца строки

   int suffixIndex; - позиция в суффикса в тексте

2. TNode.cpp(Описание функций объявленных в TNode.h)

   void BuildSuffixTree(); -Построить суффиксное дерево

   void ExtendSuffixTree(int pos); - выполнение фазы добавления в дерево

   void DeleteSuffixTree(TNode* node); - удаление дерева

   void Search(string& pat, int patNum); - поиск в тесте всех вхождений образца

   void CountIndex(TNode* node, vector<int>& vec); - посчитать индексы всех листьев, исходящих из данной вершины

   int EdgeLength(TNode *node); - длинна дуги

3. main.cpp(Считывание данных, управление программой)

Дневник отладки

Программа не проходила тест №10 по нехватке памяти, решилось всё заменой массива переходов из узла на ассоциативный массив(map).

Тест производительности

Во всех рассматриваемых тестах одинаковые строки для поиска, но во втором исходный текст в 3 раза больше первого, а в третьем в 9 раз больше.

1. Time of working: 0.582
2. Time of working: 1.452
3. Time of working: 0.177

Выводы

Самый универсальный алгоритм для различного рода задач, связанных с поиск в строке, и при этом самый эффективный - это алгоритм Укконена. Он основывается на наивном алгоритме построения дереве с кубической сложностью, но с помощью различных улучшений работает за линейную сложность. из-за большого количества модификаций наивного алгоритма, алгоритм Укконена становится сложен для понимания, однако остается самым лучшим из существующих.