**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №5**

**по дисциплине «Построение и анализ алгоритмов»**

Тема: «**Алгоритм Ахо-Корасик**»

|  |  |  |
| --- | --- | --- |
| Студент гр. 6381 |  | Попов Н.В. |
| Преподаватель |  | Филатов А.Ю. |

Санкт-Петербург

2018

**Цель работы.**

Решение задачи точного поиска набора образцов и для одного образца с джокером.

**Описание алгоритма.**

В процессе работы алгоритма Ахо-Корасика используется структура данных – вершина бора и построения по нему конечного детерминированного автомата. Строится бор последовательным добавлением исходных строк. Изначально есть 1 вершина, корень - пустая строка. Добавление строки происходит так: начиная в корне, двигаемся по дереву, выбирая каждый раз ребро, соответствующее очередной букве строки. Если такого ребра нет, то мы создаем его вместе с вершиной. Так как процесс добавления строки может остановиться во внутренней вершине, то для каждой строки будем дополнительно хранить признак того является она строкой из условия или нет. Далее, строим конечный детерминированный автомат. Состояние автомата — это какая-то вершина бора. Переход из состояний осуществляется по 2 параметрам — текущей вершине v и символу ch по которому нам надо сдвинуться из этой вершины. Назовем суффиксной ссылкой вершины v указатель на вершину u, такую что строка u — наибольший cобственный суффикс строки v, или, если такой вершины нет в боре, то указатель на корень. В частности, ссылка из корня ведет в него же. Если из текущей вершины есть ребро c символом v, то пройдем по нему, в обратном случаем пройдем по суффиксной ссылке и запустимся рекурсивно от новой вершины. Алгоритм завершится, когда мы дойдем до конца строки.При поиске с джокером строка-шаблон разделяется по символу-джокеру на строки, каждая из которых добавляется в бор.

**Описание функций.**

* void readTextAndPatt(string &text) – чтение текста и шаблонов.
* void addStrToBohr() – добавление шаблонов в бор.
* int getSuffLink(int ind) – вычисление суффиксной ссылки вершины ind.
* int getAutoMove(int ind, char ch) – вычисление перехода от вершины ind по ребру ch.
* int getGoodSuffLink(int ind) - вычисление «хорошей» суффиксной ссылки вершины ind.
* void check(int curr, int end\_symb) – хождение по «хорошим» суффиксным ссылкам из позиции curr.
* void findAllPos(string text) – поиск всех вхождений шаблонов в text.
* vector<int> splitPattern(stringstream& str\_pat, char joker) – разделение шаблона с джокером на подстроки.
* void print(const vector<int>& count, int text\_size, int pattern\_size) – вывод строки с номерами позиций вхождений шаблона.

Т**естирование.**

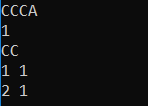


Рисунок 1. Aho.exe. Пример №1.

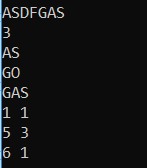


Рисунок 2. Aho.exe. Пример №2.

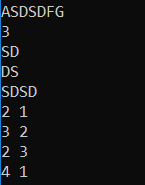


Рисунок 3. Aho.exe. Пример №3.



Рисунок 4 Joker.exe. Пример №1.

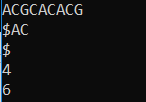


Рисунок 5. Joker.exe. Пример №2.

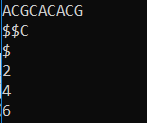


Рисунок 6. Joker.exe. Пример №3.

**Вывод.**

В ходе выполнения лабораторной работы был реализован алгоритм, решающий задачу точного поиска набора образцов с помощью алгоритма Ахо-Корасик.