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. 7383 _____ Корякин М.П.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

Изучить и реализовать на языке программирования c++ алгоритм Ахо–Корасик, который осуществляет поиск множества подстрок в тексте с помощью построения бора.

Формулировка задания.

Разработать программу, решающую задачу точного поиска набора образцов.

Вход:

Первая строка содержит текст (T, $1 \leq |T| \leq 100000$).

Вторая – число n ($1 \leq n \leq 3000$), каждая следующая из n строк содержит шаблон из набора $P = \{p_1, \dots, p_n\}$ $1 \leq |pi| \leq 75$

Все строки содержат символы из алфавита {A, C, G, T,N}

Выход:

Все вхождения образцов из P в T.

Каждое вхождение образца в текст представить в виде двух чисел – i p

Где i – позиция в тексте (нумерация начинается с 1), с которой начинается вхождение образца с номером p (нумерация образцов начинается с 1).

Строки выхода должны быть отсортированы по возрастанию, сначала номера позиции, затем номера шаблона.

Алгоритм.

Шаг 1: выбирается образец, если образцов не осталось, то переход на шаг 3.

Шаг 2: выбранный образец добавляется в бор, переход на шаг 1.

Шаг 3: текущий символ = первый символ текста; текущая вершина = корень.

Шаг 4: переход из текущей вершины по текущему символу с помощью перехода по автомату.

Шаг 5: проверка на встретившееся шаблоны в вершине, выбранной на шаге 4, с помощью перехода по хорошим суффиксальным ссылкам.

Шаг 6: если текст не закончился, то текущая вершина = вершина, выбранная на шаге 4, текущий символ = следующий символ текста, переход на шаг 4.

Выводы.

В ходе выполнения данной лабораторной работы были изучен и реализован на языке программирования C++ алгоритм Ахо–Корасик, результатом работы которого являются индексы вхождений подстроки в тексте и номер этой подстроки.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД.

```
#include <iostream>
#include <vector>
#include <string>

using namespace std;

struct bohr_vertex
{
    int next_vertex[5]; //{A, C, G, T, N} алфавит
    bool flag;          //является ли концом подстроки
    int num;            //номер подстроки
    int parent;         //индекс родителя
    int sufflink;       //индекс перехода по суффиксальной ссылке
    int gotosymbol[5];  //индекс перехода по каждому символу
    int symboltoparent; //возвращает индекс символа, по которому переходится из
родителя
    int goodsufflink;   //индекс перехода по хорошей суффиксальной ссылке
};

class Bohr
{
private:
    vector<bohr_vertex> bohr;
    string text;


public:
    Bohr()
    {
        bohr.push_back({{-1, -1, -1, -1, -1}, false, 0, 0, -1, {-1, -1, -1, -1,
-1}, -1, -1});
        cin >> text;
        int N;
        cin >> N;
        for(int i(0); i < N; i++)
        {
            string temp;
            cin >> temp;
            push(temp, i+1);
        }
    }

    void push(string str, int number)
    {
        int len = str.size();
        int index = 0;
        int symbol;
        for(int i(0); i < len; i++)
        {
            switch(str.at(i))
            {
                case 'A':
                            symbol = 0;
                            break;
                case 'C':
                            symbol = 1;
                            break;
```

```
                case 'G':
                            symbol = 2;
                            break;
                case 'T':
                            symbol = 3;
                            break;
                case 'N':
                            symbol = 4;
                            break;
            }
            if(bohr[index].next_vertex[symbol] == -1)
            {
                bool isEnd = false;
                if(i == len-1)
                    isEnd = true;
                bohr.push_back({{-1, -1, -1, -1, -1}, isEnd, number, index, -1,
{-1, -1, -1, -1, -1}, symbol, -1});
                bohr[index].next_vertex[symbol] = bohr.size() - 1;
            }
            index = bohr[index].next_vertex[symbol];
        }
    }

    int getSuffLink(int vertex)
    {
        if(bohr.at(vertex).sufflink == -1)
        {
            if(vertex == 0 || bohr.at(vertex).parent == 0)
                bohr.at(vertex).sufflink = 0;
            else
                bohr.at(vertex).sufflink =
getLink(getSuffLink(bohr.at(vertex).parent), bohr.at(vertex).symboltoparent);
        }
        return bohr.at(vertex).sufflink;
    }

    int getLink(int vertex, int symbol)
    {
        if(bohr.at(vertex).gotosymbol[symbol] == -1)
        {
            if(bohr.at(vertex).next_vertex[symbol] != -1)
                bohr.at(vertex).gotosymbol[symbol] =
bohr.at(vertex).next_vertex[symbol];
            else
                bohr.at(vertex).gotosymbol[symbol] = (vertex == 0) ? 0 :
getLink(getSuffLink(vertex), symbol);
        }
        return bohr.at(vertex).gotosymbol[symbol];
    }

    int getGoodSuffLink(int vertex)
    {
        if(bohr.at(vertex).goodsufflink == -1)
        {
            int temp = getSuffLink(vertex);
            if(temp == 0)
                bohr.at(vertex).goodsufflink = 0;
            else
                bohr.at(vertex).goodsufflink = (bohr.at(temp).flag) ? temp :
getGoodSuffLink(temp);
        }
        return bohr.at(vertex).goodsufflink;
    }
```

```
    void check(int v, int i)
    {
        for(int u(v); u != 0; u = getGoodSuffLink(u))
        {
            if(bohr.at(u).flag)
            {
                int delta = 0;
                int temp = u;
                while(bohr.at(temp).parent != 0)
                {
                    temp = bohr.at(temp).parent;
                    delta++;
                }
                cout << i - delta << " " << bohr.at(u).num << endl;
            }
        }
    }

    void AHO()
    {
        int vertex = 0, symbol = 0;
        for(int i(0); i < text.length(); i++)
        {
            switch(text.at(i))
            {
                case 'A':
                            symbol = 0;
                            break;
                case 'C':
                            symbol = 1;
                            break;
                case 'G':
                            symbol = 2;
                            break;
                case 'T':
                            symbol = 3;
                            break;
                case 'N':
                            symbol = 4;
                            break;
            }
            vertex = getLink(vertex, symbol);
            check(vertex, i + 1);
        }
    }
};

int main()
{
    Bohr object;
    object.AHO();
    return 0;
}
```