МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МОЭВМ

отчет

по лабораторной работе №5

по дисциплине «Построение и анализ алгоритмов»

Тема: Алгоритм Ахо-Корасик

Студент гр.8304 _____ Порывай П.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2020

Цель работы.

Используя реализацию точного множественного поиска, решите задачу точного поиска для одного образца с *джокером*.

Задание.

Вариант 1

На месте джокера может быть любой символ, за исключением заданного.

В шаблоне встречается специальный символ, именуемый джокером (wild card), который "совпадает" с любым символом. По заданному содержащему шаблоны образцу *P* необходимо найти все вхождения P в текст T.

Например, образец a*b*??c? с джокером ? встречается дважды в тексте *xabvccbababcax*.

Символ джокер не входит в алфавит, символы которого используются в *T*. Каждый джокер соответствует одному символу, а не подстроке неопределённой длины. В шаблон входит хотя бы один символ не джокер, т.е. шаблоны вида ??? недопустимы. Все строки содержат символы из алфавита {*A*, *C*, *G*, *T*, *N*}

Описание алгоритма.

Алгоритм строит конечный автомат, которому затем передаёт строку поиска. Автомат получает по очереди все символы строки и переходит по соответствующим рёбрам. Если автомат пришёл в конечное состояние, соответствующая строка словаря присутствует в строке поиска.

Для того чтобы найти все вхождения в текст заданного шаблона с масками Q, необходимо обнаружить вхождения в текст всех его безмасочных кусков.

Пусть {Q1,…,Qk} — набор подстрок Q, разделенных масками, и пусть {l1, …,lk} — их стартовые позиции в Q. Например, шаблон abφφcφ содержит две подстроки без масок ab и cc и их стартовые позиции соответственно 1 и 5.

Для алгоритма понадобится массив C. C[i] — количество встретившихся в тексте безмасочных подстрок шаблона, который начинается в тексте на позиции i. Тогда появление подстроки Q_i в тексте на позиции j будет означать возможное появление шаблона на позиции $j-l_i+1$.

1. Используя алгоритм Ахо-Корасик, находим безмасочные подстроки шаблона Q: когда находим Q_i в тексте T на позиции j, увеличиваем на единицу $C[j-l_i+1]$.
2. Каждое i, для которого C[i] = k, является стартовой позицией появления шаблона Q в тексте.

Вычислительная сложность алгоритма: O(2m + n + a), где n – длинна шаблона, m – длинна текста, a – кол-во появлений подстрок шаблона.

Описание функций и структур данных.

Структура для хранения вершины бора, а сам бор хранится в векторе таких вершин:

```
struct Vertex
{
      std::vector<int> next;
      bool is_leaf = false;
      std::vector<size_t> str_nums;
      int link = -1;
      int from = -1;
      char how = 0;
      std::vector<int> go;
   };
```

Функция добавления строки в бор:

```
void inpBor(std::string& str, std::vector<Vertex>& bor, std::map<char,
int>&
      alphabet, int str_num)
```

Функция проверки на наличие строки в боре:

```
void inpBor(std::string& str, std::vector<Vertex>& bor, std::map<char,
int>&
      alphabet, int str_num)
```

Функция выявления суффиксной ссылки:

```
int get_link(int v, std::vector<Vertex>& bor);
```

Функция для перехода из вершины v:

```
int go(int v, char c, std::vector<Vertex>& bor)
```

Функция поиска:

```
void AhoCorasik(std::ostream& out,std::istream& in )
```

Тестирование

Таблица 1 – результаты тестирования

Input	**Output**
NACGNTTACGGTCACNN AC$$T$AC$$ $ C	2
NACGNTTACGGTCACNN AC$$T$AC$$ $ A	2 8
ACTANCA A$$A$ $ G	1

Выводы.

В ходе выполнения работы, была написана программа, находящая вхождение образца с джокером, получены знания о такое структуре данных как бор.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
#include <string>
#include <vector>
#include <algorithm>
#include <fstream>
#include <map>
#include <iostream>



struct Vertex
{
	std::vector<int> next;
	bool is_leaf = false;
	std::vector<size_t> str_nums;
	int link = -1;
	int from = -1;
	char how = 0;
	std::vector<int> go;
};



Vertex make_bor_vertex(int from, char how)
{
	Vertex vert;
	vert.next = { -1, -1, -1, -1, -1 };
	vert.go = { -1, -1, -1, -1, -1 };
	vert.from = from;
	vert.how = how;
	return vert;
}



void inpBor(std::string& str, std::vector<Vertex>& bor, std::map<char,
int>&
	alphabet, int str_num)
{
	int borInd = 0;
	for (auto c : str)
	{
		char cInd = alphabet[c];
		if (bor[borInd].next[cInd] == -1)
		{
			bor.push_back(make_bor_vertex(borInd, cInd));
			bor[borInd].next[cInd] = bor.size() - 1;
		}

		borInd = bor[borInd].next[cInd];
	}
```

```
	bor[borInd].is_leaf = true;
	bor[borInd].str_nums.push_back(str_num);
}



int get_link(int v, std::vector<Vertex>& bor);


int go(int v, char c, std::vector<Vertex>& bor)
{
	if (bor[v].go[c] == -1)
	{
		if (bor[v].next[c] != -1)
			bor[v].go[c] = bor[v].next[c];
		else
		{


			if (v != 0)
				bor[v].go[c] = go(get_link(v, bor), c, bor);
			else if(v==0)
				bor[v].go[c] = 0;

		}
	}


	return bor[v].go[c];
}

int get_link(int v, std::vector<Vertex>& bor)
{

	if (bor[v].link == -1)
	{
		if (v == 0 || bor[v].from == 0)
			bor[v].link = 0;

		else
			bor[v].link = go(get_link(bor[v].from, bor),
bor[v].how, bor);
	}
	return bor[v].link;
}
void AhoCorasik(std::ostream& out,std::istream& in )
{
	std::map<char, int> alphabet;

	alphabet['N'] = 4;
	alphabet['A'] = 0;
	alphabet['C'] = 1;
	alphabet['G'] = 2;
	alphabet['T'] = 3;

	std::string text;
	std::string pat;
```

```
char J;
in >> text;
in >> pat;
in >> J;
char no_joker;//Символ который не считается Джокером
in >> no_joker;
pat += J;
std::vector<std::string> q;
std::vector<size_t> l;
std::string cur;


for (size_t i = 0; i < pat.length(); ++i)
{
	if (pat[i] == J)
	{
		if (!cur.empty())
		{
			q.push_back(cur);
			l.push_back(i - cur.size() + 1);
		}
		cur.clear();
	}
	else

		cur += pat[i];
}
std::vector<Vertex> bor;
bor.push_back(make_bor_vertex(0, 0));
out << "Подстрока паттерна - ";
for (size_t i = 0; i < q.size(); ++i)
{
	inpBor(q[i], bor, alphabet, i);
	if (i == q.size() - 1)
		out << q[i];
	else
		out << q[i] << ", ";
}
out << "\n\nБор создан\n";
std::vector<size_t> c(text.size());
bool f;
int vert_num;
int u = 0;
for (size_t i = 0; i < text.length(); ++i)
{
	out << "Идем из вершины с данным значением " << u;
	u = go(u, alphabet[text[i]], bor);
	out << " в вершину с значением " << u;
	f = false;
	for (int v = u; v != 0; v = get_link(v, bor))
	{
		out << v << " -> ";
		if (bor[v].is_leaf)
		{
			f = true;
			vert_num = v;
			for (auto& str_num : bor[v].str_nums)
			{
```

```
                        int j = i - q[str_num].length() + 1;
                        if (j >= l[str_num] - 1)
                            ++c[j - l[str_num] + 1];
                    }
                }
            }
            out << "0";
            if (f == true)
                out << "\nЛист найден, его номер - " << vert_num ;


            out << "\n";
        }


        for (size_t i = 0; i < text.size(); ++i)
            if (c[i] == q.size())
            {
                bool is_correct = true;
                for (size_t k = i; k < i + pat.size() - 1; ++k)
                {
                    if (pat[k - i] == J && text[k] == no_joker)
                    {
                        is_correct = false;
                        break;

                    }
                }
                if (is_correct)
                    out << i + 1 << "\n";
            }
}
int main()
{
    setlocale(LC_ALL, "Russian");
    std::ifstream in("input.txt");
    std::ofstream out("output.txt");


    int a, b;
    std::cout << "Чтение из файла,консоли(1/2) \n";
    std::cin >> a;
    std::cout << "\nЗапись в файл, консоль (1/2)\n";
    std::cin >> b;


    if (a == 1 && b == 1)
        AhoCorasik(out, in);
    if (a == 1 && b == 2)
        AhoCorasik( std::cout, in);
    if (a == 2 && b == 2)
        AhoCorasik(std::cout, std::cin);
    if (a == 2 && b == 1)
        AhoCorasik( out, std::cin);


    return 0;
```

```
}
```