МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Построение и анализ алгоритмов»

Тема: Алгоритм Кнута-Мориса-Пратта

Студент гр. 3388 _____ Снигирев А.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2025

Цель работы:

Изучить алгоритмы, реализующие поиск подстроки в строке. Узнать, что такое префикс-функция и реализовать алгоритм Кнута-Мориса-Пратта.

Задание:

Реализуйте алгоритм КМП и с его помощью для заданных шаблона P ($|P| \leq 15000$) и текста T ($|T| \leq 5000000$) найдите все вхождения P в T.

Вход:
Первая строка - P
Вторая строка - T
Выход:
индексы начал вхождений P в T, разделенных запятой, если P не входит в T, то вывести -1

Рис.1 - Задание

Реализация

Префикс функция

Префикс-функция принимает строку и возвращает массив. Массив совпадает со строкой по длине.

Алгоритм рассматривает набор подстрок, каждая из которых содержит предыдущую подстроку и следующий элемент. В каждую ячейку массива записывается число, обозначающее максимальную длину суффикса, совпадающего с префиксом для текущей подстроки.

`prefixFunction()` строит вектор не по определению, а более эффективно. Для каждой новой длины префикса используется уже посчитанная предыдущая длина для уменьшения числа операций.

Алгоритм КМП

Алгоритм принимает строку и паттерн, вхождения которого необходимо найти. Он конкатенирует паттерн со строкой, между ними добавляя разделитель. Для разделения подойдет любой символ, не содержащийся в используемом алфавите.

Для этой строки вызывается префикс-функция.

Далее — все просто. Паттерн является префиксом, поэтому чтобы найти все его вхождения в строку, достаточно просто в префикс-функции найти числа, которые совпадают с длиной паттерна. Если из из индекса вычесть длину паттерна, то можно получить индекс вхождения.

Этот алгоритм реализован в функции `kmp()`.

Исходный код программы см. в **ПРИЛОЖЕНИИ А.**

Оценка сложности алгоритма:

Временная сложность: O(n+k)

Вычисление префикс-функции происходит за O(n).

Линейный поиск вхождений по префикс-функции происходит за O(k)

Память: O(n)

Две строки по O(n), сумма этих строк также O(n). Вектор префикс-функции O(n), вектор вхождений аналогично.

Итого: O(n+n+n+n)=O(n)

Вывод

В ходе лабораторной работы был реализован поиск подстроки в строке с помощью алгоритма Кнута-Мориса-Пратта. Он работает за линейное время, поэтому чисто ассимптотически лучшего алгоритма просто нет. КМП значительно превосходит наивный алгоритм.

Исходный код см. **в ПРИЛОЖЕНИИ А.**

Тестирование см. в **ПРИЛОЖЕНИИ Б.**

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД ПРОГРАММЫ

kmp.cpp

```
#include <iostream>
#include <vector>
#define DEBUG 0
using namespace std;

void print(vector<int> vec){
    cout<<"["<<vec[0];
    for(int i=1;i<vec.size();i++){
        cout<<", "<<vec[i];
    }
    cout<<"]"<<endl;
}
vector<int> prefixFunction(string str){
    vector<int> result;
    result.push_back(0);
       if(DEBUG)  cout<<"         Первую позицию инициализируем 0 по
определению"<<endl;
    if(DEBUG){ cout<<"     Вектор: ";
    print(result);}
    for(int i = 1; i<str.size(); i++){
        int k = result[i - 1];
        while(k > 0 && str[i] != str[k]){
            k = result[k - 1];
        }
        if(str[i] == str[k])
            k++;
        result.push_back(k);
        if(DEBUG){
            cout<<"     Текущая подстрока: "<<str.substr(0,i)<<endl;
            cout<<"     Префикс-функция: ";
            print(result);
        }
    }
    return result;
}

vector<int> kmp(string P, string T)
{
    if(DEBUG) cout<<"Запуск алгоритма Кнутта-Мориса-Пратта..."<<endl;

    int pl = P.size();
    int tl = T.size();
       if(DEBUG)  cout<<"Длина паттерна:  "<<pl<<endl<<"Длина строки:
"<<tl<<endl;
    vector<int> answer;
          if(DEBUG)   cout<<"Вычисление префиксной функции строки
"<<P+"#"+T<<endl;
    vector<int> p = prefixFunction(P + "#" + T);
    int count = 0;
    for (int i = 0;i<tl;i++){
```

```
        if(p[pl + i + 1] == pl){
            answer.push_back(i - pl + 1);
            if(DEBUG)
            {
                        cout<<"Найден элемент с префикс-функцией ==
"<<pl<<endl;
                   cout<<"Добавляем в ответ индекс нового вхождения ==
"<<pl<<endl;
                cout<<"Найденные индексы вхождений: "<<endl;
                print(answer);
            }
        }

    }
    if(answer.empty()){
          if(DEBUG) cout<<"Ни одного вхождения не найдено, результат
инициализируем -1"<<endl;
        answer.push_back(-1);
    }
    return answer;
}

int main(){
    string str1;
    string str2;
    cout<<"Введите строку: ";
    cin>>str1;
    cout<<"Введите паттерн: ";
    cin>>str2;
    cout<<"Все вхождения: "<<endl;
    print(kmp(str2, str1));
}
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

in	out
lannister goyda	[-1]
thelordoftherings the	[0, 9]
moevm m	[0, 4]
Algorithm rit	[4]