МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

Отчет

Практическая работа №10

Дисциплина Структуры и алгоритмы обработки данных

Тема. Поиск в тесте образца. Алгоритмы. Эффективность алгоритмов.

Выполнил студент Дамарад Д.В.

Фамилия И.О.

Группа ИКБО-13-21

Номер группы

Ответы на вопросы

1) Строкой называют последовательность символов.

2) Префиксом строки называют символы, стоящие на позиции от начала строки и до предпоследнего символа.

3) Постфиксом строки называют символы, стоящие на позиции от конца строки до второго символа.

4) Асимптотическая сложность последовательного поиска подстроки в строке – $O(n * m)$, где n – длина строки, а m – длина подстроки.

5) Особенностью в том, что он выполняет сравнения в шаблоне справа налево.

6) Сложность – $O(n + m)$, где n – длина строки, а m – длина подстроки.

7) Строка abcdefdhwq, подстрока abc являются входными данными для эффективного поиска подстроки в тексте, так как подстрока находится в самом начале.

8) Для алгоритма Кнута-Мориса-Пратта строка – abababaaabaaa

Для алгоритма Бойера и Мура - aaaaaaabaaaaa

9) Чем больше таблица кодов, тем дольше ее заполнять.

10) За счет того, что первый символ в слове не повторяется и позволяет пропустить наибольшее количество шагов.

Задание №1

1.1 Описание модели решения: для нахождения количества вхождений подстроки в строку строки необходимо перебрать всевозможные строки подаваемой подстроки, на очередном переборе используем алгоритм Кнута-Морриса-Пратта (КМП).

1. Этап КМП: Префикс-функция для i-ого символа образа возвращает значение, равное максимальной длине совпадающих префикса и суффикса подстроки в образе, которая заканчивается i-м символом. Это значение будет хранить в векторе pi[i].
2. Этап КМП: поиск образа строке.

1.2 Количество сравнений для поиска вхождения в текст: (N+M), где M - количество букв в подстроке, N – количество букв в строке поиска.

1.3 Функции для выполнения алгоритма:

Функция для заполнения массива смещений:

```
void KMP_stage1(string subline, vector<int>& pi) {
    pi[0] = 0;
    int i = 1, j = 0;
    for (; i < subline.size();) {
        if (subline[i] == subline[j]) {
            pi[i] = j + 1;
            i++;
            j++;
        }
        else if (j == 0) {
            pi[i] = 0;
            i++;
        }
        else {
            j = pi[j - 1];
        }
```

```
    }
}
```

Алгоритм поиска:

```
int KMP_stage2(string line, string subline, vector<int> pi) {
    if (line == "") {
        return -1;
    }
    int i = 0, j = 0;
    for (;;) {
        if (line[i] == subline[j]) {
            j++;
            i++;
            if (j == subline.size() - 1) return i - 1;
        }
        else {
            if (j == 0) {
                i++;
                if (i >= line.size() - 1) return -1;
            }
            else {
                j = pi[j - 1];
            }
        }
    }
}
```

Алгоритм поиска количества вхождений подстроки в строку:

```
int CountInput(string line, string subline) {
    vector<int> pi;
    pi.resize(subline.size());
    KMP_stage1(subline, pi);
```

```
    int count = 0;
    int indexInput = 0;
    indexInput = KMP_stage2(line.substr(indexInput), subline, pi);
    while (indexInput != -1 && indexInput + subline.size() <= line.size()) {
        count++;
        indexInput = KMP_stage2(line.substr(indexInput + subline.size()),
subline, pi) + subline.size() + indexInput;
    }
    return count;
}
```

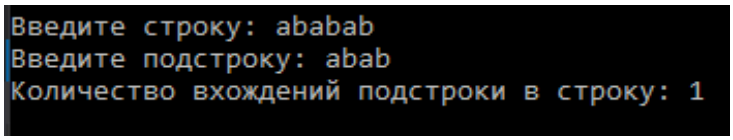
1.4 Предусловие: подстрока и строка, в которой будет считаться кол-во вхождений подстроки.

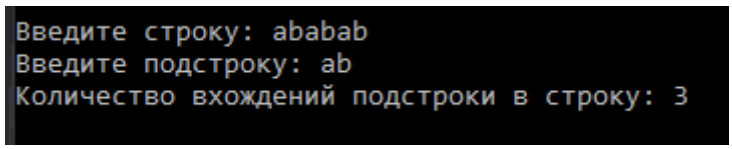
Постусловие: количество вхождений подстроки в строку.

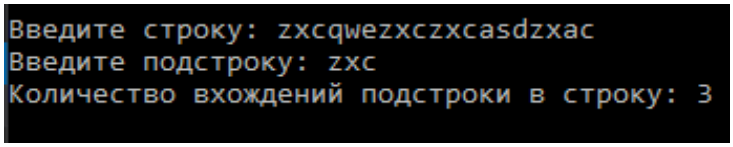
2. Таблица тестов:

Номер теста	Входные данные	Выходные данные	Ожидаемый результат	Пройден/Не пройден
1	ababab ab	3	3	Пройден
2	ababab abab	1	1	Пройден
3	zxcqwezxczxcasdzxac zxc	3	3	Пройден

2.1 Скриншоты тестирования:







3. Таблица сложности алгоритма в зависимости от длины текста и образца:

Длина подстроки равняется 10% от самой строки.

n	**T(n)**	**Тт=f(C+M)**	**Тп=Сф+Мф**
100	0 мс	110	305
1000	0 мс	1100	3039
10000	0 мс	11000	30377
100000	5 мс	110000	303744
1000000	43 мс	1100000	3037107

Длина подстроки равняется 1.

n	**T(n)**	**Тт=f(C+M)**	**Тп=Сф+Мф**
100	0 мс	101	305
1000	0 мс	1001	3040
10000	0 мс	10001	30396
100000	4 мс	100001	303769
1000000	36 мс	1100000	3038685

4. Полный код программы

```
#include <vector>
#include <string>
#include <iostream>
using namespace std;

void KMP_stage1(string subline, vector<int>& pi) {
    pi[0] = 0;
    int i = 1, j = 0;
    for (; i < subline.size();) {
        if (subline[i] == subline[j]) {
            pi[i] = j + 1;
            i++;
            j++;
        }
        else if (j == 0) {
            pi[i] = 0;
            i++;
        }
        else {
            j = pi[j - 1];
        }
    }
}

int KMP_stage2(string line, string subline, vector<int> pi) {
    if (line == "") {
        return -1;
    }
    int i = 0, j = 0;
    for (;;) {
        if (line[i] == subline[j]) {
            j++;
            i++;
            if (j == subline.size() - 1) return i - 1;
        }
        else {
            if (j == 0) {
                i++;
                if (i >= line.size() - 1) return -1;
            }
            else {
                j = pi[j - 1];
            }
```

```
            }
        }
    }

    int CountInput(string line, string subline) {
        vector<int> pi;
        pi.resize(subline.size());
        KMP_stage1(subline, pi);
        int count = 0;
        int indexInput = 0;
        indexInput = KMP_stage2(line.substr(indexInput), subline, pi);
        while (indexInput != -1 && indexInput + subline.size() <= line.size())
{
            count++;
            indexInput = KMP_stage2(line.substr(indexInput + subline.size()),
subline, pi) + subline.size() + indexInput;
        }
        return count;
    }

    int main() {
        system("chcp 1251 > null");
        string text;
        string subline;
        cout << "Введите строку: ";
        getline(cin, text);
        cout << "Введите подстроку: ";
        getline(cin, subline);
        cout << "Количество вхождений подстроки в строку: " <<
CountInput(text, subline) << endl;
        return 0;
    }
```

Вывод

В ходе проделанной работы были получены знания и навыки применения алгоритмов поиска в тексте подстрок (образца).