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-Морриса-Пратта.

Студент гр. 7304 Моторин Е.В.

Преподаватель Филатов А.Ю.

Санкт-Петербург

2019

Цель работы:

Изучения алгоритма Кнута-Морриса-Пратта - алгоритм поиска подстроки в строке.

Задача:

Заданы две строки AA ($|A|\leq 5000000$|A|≤5000000)
и BB ($|B|\leq 5000000$|B|≤5000000).
Определить, является ли AA циклическим сдвигом BB (это значит,
что AA и BB имеют одинаковую длину и AA состоит из суффикса BB,
склеенного с префиксом BB). Например, defabc является циклическим
сдвигом abcdef.
Вход:
Первая строка - AA
Вторая строка - BB
Выход:
Если AA вляется циклическим сдвигом BB, индекс начала строки BB в AA,
иначе вывести $-1-1$. Если возможно несколько сдвигов вывести первый
индекс.

Sample Input:

defabc
abcdef

Sample Output:

3

Основные теоретические положения:

Алгоритм Кнута — Морриса — Пратта (КМП-алгоритм) — эффективный алгоритм, осуществляющий поиск подстроки в строке. Время работы алгоритма линейно зависит от объёма входных данных, то есть разработать асимптотически более эффективный алгоритм невозможно.

Ход работы:

1. **Реализована префикс-функция.**
   Префикс функция от строки S и позиции i в ней – длина k наибольшего префикса подстроки S[1..i].

```
vector<int> prefixFunction(string str) {
    vector<int> entries(str.length());

    for (int i = 1; i < str.length(); i++) {
        int j = entries[i - 1];

        while ((j > 0) && (str[i] != str[j]))
            j = entries[j - 1];

        if (str[i] == str[j])
            ++j;

        entries[i] = j;
    }

    return entries;
}
```

2. **Алгоритм Кнута-Морриса-Пратта.**
   Функция реализует алгоритм нахождения подстроки в строке. Строка и подстрока разделяются символом-разделителем и вычисляется префикс-функция. Проходится массив значений, полученных от префикс-функции, если значение равно длине подстроки – оно добавляется в ответ.

```
vector<int> KMPA(string str, string substr) {
    vector<int> result;
    vector<int> p = prefixFunction(substr + "+" + str);

    for (unsigned i = 0; i < p.size(); i++)
        if (p[i] == substr.length())
            result.push_back(i - 2 * int(substr.length()));

    return result.size() > 0 ? result : vector<int>(1, -1);
}
```

3. **Нахождение циклического сдвига.**
   Строки соединяется, используя символ-разделитель. Находится префикс-функция для данной строки. Искомое значение должно лежать на последнем месте, сдвиг проверяется путем посимвольного сравнения строк с учетом сдвига.

```
int cycleShiftIndex(string s1, string s2) {
    if (s1.length() != s2.length()) return -1;
    if (s1 == s2) return 0;
    vector<int> p = prefixFunction(s1 + "+" + s2);
    int shift = p[p.size() - 1];

    for (int i = shift; i < s1.length(); i++)
        if (s1[i] != s2[i - shift]) return -1;
    return shift;
}
```

Результат:

Из рисунков 1 и 2 видно, что разработанная программа выполняют поставленные задачи, а именно: программа находит вхождения подстроки в строку и циклический сдвиг строк.

```
ab
abcdcacjvbabablkvnlsnvababbabab
0,10,12,22,24,27,29Program ended with exit code: 0
```

Рисунок 1.

```
defabc
abcdef
3Program ended with exit code: 0
```

Рисунок 2.

Вывод:

Таким образов, в ходе данной лабораторной работы было подробно изучено написание префикс функции, алгоритма Кнута-Морриса-Пратта и алгоритма нахождения циклического сдвига строк. КМП алгоритм - эффективный алгоритм, осуществляющий поиск подстроки в строке. Время работы алгоритма линейно зависит от объёма входных данных, то есть разработать асимптотически более эффективный алгоритм невозможно. Сложность данного алгоритма по времени O(m + n), по памяти O(n).