**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 Кнута-Морриса-Пратта.**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент гр. 6304 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Ковынев М.В. | |
| Преподаватель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | Балтрашевич В.Э. |

Санкт-Петербург

2018

# **Цель работы.**

В данной лабораторной работе необходимо реализовать поиск подстроки в тексте и определение того, является ли одна строка циклическим сдвигом другой, используя алгоритм Кнута-Морриса-Пратта.

# **Постановка задачи.**

**Задача 1: Алгоритм Кнута-Морриса-Пратта**

Реализуйте алгоритм КМП и с его помощью для заданных шаблона **P (|P|≤15000)** и текста **T (|T|≤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

**Задача 2: Циклический сдвиг**

Заданы две строки **A (|A|≤5000000)** и **B (|B|≤5000000)**. Определить, является ли **А** циклическим сдвигом **В** (это значит, что **А** и **В** имеют одинаковую длину и **А** состоит из суффикса **В**, склеенного с префиксом **В**). Например, defabc является циклическим сдвигом abcdef.

**Вход:**

* Первая строка - **A**
* Вторая строка - **B**

**Выход:**

* Если **A** вляется циклическим сдвигом **B**, индекс начала строки **B** в **A**, иначе вывести −1. Если возможно несколько сдвигов вывести первый индекс.

# **Теоретические сведения**

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

Основной идеей алгоритма Кнута-Морриса-Пратта является использование информации о предыдущем сравнении. После частичного совпадения начальной части образа с символами строки при каждом несовпадении двух символов образец сдвигается на максимально возможное количество символом (d >= 1) и сравнение строки с образцом (с начала образца) продолжается.

Алгоритм требует порядка *N+M* сравнений символов даже в худшем случае. В наивном алгоритме поиска подстроки в тексте мы всегда будем итерироваться по каждому символу текста, несмотря на информацию, полученную в предыдущих сравнениях.

Для использования алгоритма КМП необходимо реализовать префикс функцию. Эта величина *p*[*j*] равна размеру самой длинной последовательности символов образца, непосредственно предшествующей *j*-му символу, которая совпадает с началом образца (для *j*=0 принимают *p*[*j*] = -1).

# **Ход работы.**

1. Написан алгоритм для вычисления префикс функции:

vector<int> prefixFunction(const string& text)

{

if (text.empty())

return {};

vector<int> prefixVector(text.size(), 0);

for (int i = 1; i < text.size(); i++)

{

int k = prefixVector[i - 1];

while (k > 0 and text[i] != text[k])

k = prefixVector[k - 1];

if (text[i] == text[k])

k++;

prefixVector[i] = k;

}

return prefixVector;

}

1. Реализован КМП-алгоритм для поиска строки в тексте:

vector<int> algoKnuthMorrisPratt(const string& pattern, const string& text)

{

vector<int> resultVector;

vector<int> prefixVector = prefixFunction(pattern);

if (pattern.empty() || text.empty())

return { -1 };

int pos = 0;

for (int i = 0; i < text.size(); i++)

{

while (pos > 0 and (pos >= pattern.size() || pattern[pos] != text[i]))

{

pos = prefixVector[pos - 1];

}

if (pattern[pos] == text[i])

pos++;

if (pos == pattern.size())

resultVector.push\_back(i - pos + 1);

}

if (resultVector.empty())

resultVector.resize(1, -1);

return resultVector;

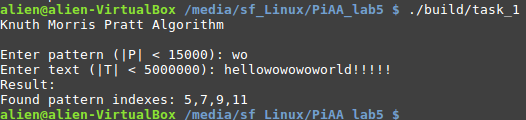
}

Рисунок 1 — Результат работы алгоритма КМП.

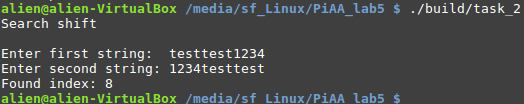
1. Для решения задачи с циклическим сдвигом реализуем функцию, которая будет явно вызывать функцию КМП.

Рисунок 2 — Результат работы поиска циклического сдвига.

1. Для сборки проекта был создан CMakeLists и скрипт для запуска сборки.
2. Реализованы параметризованные Gtest для проверки корректности результатов работы программ, решающих поставленные задачи.
3. Произведен анализ эффективности КМП/наивный алгоритма для случайных строк. За число операций было принято число операций сравнения символов.
   1. Реализован генератор строк:

string generateString(int size)

{

string result;

for (int i = 0; i < size; i++)

{

result += rand() % 26 + 'A';

}

return result;

}

* 1. Реализован наивный алгоритм:

vector<size\_t> \_naive(const string& pattern, const string& text, int& count)

{

if (!pattern.size() || !text.size())

return {};

vector<size\_t> resultVector;

for (size\_t i = 0; i < text.size() - pattern.size() + 1; i++)

{

for (size\_t j = 0; j < pattern.size(); j++)

{

count++;

if (pattern[j] == text[j + i])

{

if (j == pattern.size() - 1)

resultVector.push\_back(i);

}

else

break;

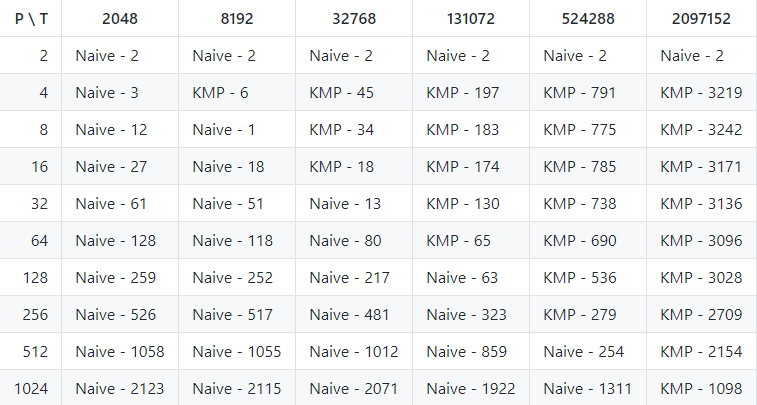
}

}

return resultVector;

}

* 1. Реализованы тесты для КМП/наивного алгоритмов.
* Варианты длин текста *text* для тестов - 2048, 8192, 32768, 131072, 524288, 2097152.
* Варианты длин подстроки *pattern* для тестов - от 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048.

Для каждой пары (*pattern,text*) было проведено по 10 тестов со случайными строками заданной длины. Результат в таблице - среднее значение. В ячейке отображается название алгоритма, справившегося за наименьшее число сравнений для данного теста, и разница в количестве сравнений:

# Таблица 1 — Эффективный алгоритм

# **Вывод.**

В ходе выполнения данной лабораторной работы изучен и реализован алгоритм Кнута-Мориса-Пратта, посредством выполнения задач поиска подстроки в тексте и определение того, является ли одна строка циклическим сдвигом другой. Также проведено исследование эффективности наивного алгоритма и КМП-алгоритма для поиска шаблона в тексте, где шаблон и текст являются случайными строками, в ходе которого выяснилось, что КМП-алгоритм гораздо эффективнее в случаях, когда длина текста значительно больше длины шаблона, но при этом проигрывает в эффективности, когда размеры текста и шаблона довольно близки.