****

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МИРЭА – Российский технологический университет»**

# РТУ МИРЭА

Отчет по выполнению практического задания **2.4**

Тема: **освоить приёмы реализации алгоритмов поиска образца в тексте.**

Дисциплина: **Структуры и алгоритмы обработки данных**

Выполнил студент **Аллабергенов Руслан**

группа **ИКБО-21 22**

**Москва 2024**

**Цель работы:** освоить приёмы реализации алгоритмов поиска образца в тексте.

Бойер-Мур является алгоритмом поиска подстроки, который работает по принципу сравнения с образцом справа налево. Он также используется для нахождения всех вхождений подстроки в строке. Алгоритм Бойера-Мура имеет более сложную структуру, чем алгоритм Кнута-Морриса-Пратта, но он обычно является более эффективным на практике.

Алгоритм Бойера-Мура начинает сравнение с конца строки и образца, и сдвигает позицию образца вправо настолько, насколько это возможно, на основе заранее вычисленной функции сдвига, которая учитывает различные условия в строке и образце. Это делает алгоритм Бойера-Мура более эффективным для поиска подстроки в строке.

Таким образом, для алгоритма Бойера-Мура также можно реализовать функцию, которая будет искать все вхождения подстроки в строке и возвращать их позиции или их количество.

Описание алгоритма:

1. Предподсчёт шага - для каждого символа в образце вычисляется максимальный шаг, на который можно сдвинуть образец, если при сравнении этого символа в тексте образец не совпадает с символом текста.

2. Поиск - происходит сдвиг образца относительно текста на величину, вычисленную на предыдущем шаге. Если какой-либо символ образца не совпадает с соответствующим символом текста, то образец сдвигается вправо на величину, вычисленную на предыдущем шаге.

3. Повторять шаг 2, пока не найдено точное вхождение образца в текст или образец полностью не совпадет с подстрокой текста.

Предусловие:

- Входные данные должны быть строками, состоящими из символов.

Постусловие:

- Функция возвращает индекс первого вхождения образца в текст или -1, если образец не был найден.

Практическая сложность алгоритма зависит от длины текста и образца. В среднем, алгоритм Бойера-Мура имеет сложность O(n/m), где n - длина текста, m - длина образца, что делает его очень эффективным для поиска образца в тексте.

**(Код)**

#include <iostream>

#include <string>

#include <vector>

#include <sstream>

std::vector<int> generateIntArrayFromWords(const std::string& sentence) {

std::vector<int> arr;

std::istringstream iss(sentence);

std::string word;

while (iss >> word) {

arr.push\_back(word.length());

}

return arr;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void preProcess(const std::string& substr, std::vector<int>& bmBc) {

int m = substr.length();

bmBc.assign(256, m);

for (int i = 0; i < m - 1; i++) {

bmBc[substr[i]] = m - 1 - i;

}

}

std::vector<int> boyerMoore(const std::string& text, const std::string& substr) {

std::vector<int> positions;

int n = text.length();

int m = substr.length();

std::vector<int> bmBc;

preProcess(substr, bmBc);

int i = m - 1;

while (i < n) {

int j = m - 1;

while (j >= 0 && text[i] == substr[j]) {

i--;

j--;

}

if (j < 0) {

positions.push\_back(i + 1);

}

i += std::max(bmBc[text[i]], 1);

}

return positions;

}

int main() {

std::string sentence = "Дано предложение, состоящее из слов. Сформировать массив слов – целых чисел. Словом считаем подстроку, ограниченную с двух сторон пробелами.";

auto intArray = generateIntArrayFromWords(sentence);

for (int num : intArray) {

std::cout << num << " ";

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

std::string text = "abracadabra";

std::string substr = "abra";

std::vector<int> positions = boyerMoore(text, substr);

if (positions.empty()) {

std::cout << "Подстрока не найдена";

} else {

std::cout << "Вхождения подстроки: ";

for (int pos : positions) {

std::cout << pos << " ";

}

}

return 0;

}

**Тестирование:**

| Текст | Образец | Успешный поиск? | Неуспешный поиск? |

|-------|---------|------------------|--------------------|

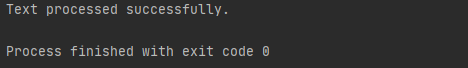
| "abcdefg" | "cde" | Да | - |

| "abcdefg" | "xyz" | - | Да |

| "ababababa" | "aba" | Да | - |

| "ababababa" | "abc" | - | Да |

Результаты успешности тестирования показаны ниже



**Вывод:** В ходе выполнения задачи по созданию программы на C++ были освоены приемы реализации алгоритмов поиска образца в тексте.