Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Кафедра ІПЗ

**Лабораторна робота № 10**

**«Прямий метод пошукуі Алгоритм Кнута, Моріса і Прата (КМП)»**

Виконав

ст. гр. ІП-22-1

Курчій С.В.

Перевірив

доцент Процюк В.Р.

Івано-Франківськ

2023

**Мета:**вивчення та практичне застосування прямого методу пошуку та алгоритм КМП , а також розвиток навичок реалізації та порівняння алгоритмів у контексті оптимізації пошукових процесів.

**Умова:**

1. Прямий метод пошуку.Нехайзадамо масив S з "n" елементів та масив Р з m елементів(Студенти вибирають самостійно). Необхідно знайти перше входження масиву Р у масиві S.

2. Алгоритм Кнута, Моріса і Прата (КМП). На вхід надходять два масиви символів: S розміром n (текст) та інший масив Р розміром m (слово), які студент вибирає самостійно. Необхідно знайти перше входження слова у тексті.

Підготувати необхідний код мовою С++ для реалізації методів(відповідних)

**Розв’язок**

**Прямий метод пошуку:**

#include<iostream>

#include<vector>

// Функція для пошуку першого входження підрядка P у тексті S

intfindSubstring(conststd::vector<int>&S, conststd::vector<int>&P)

{

    intn = S.size(); // Довжина тексту

    intm = P.size(); // Довжина підрядка

    // Проходимо весь текст

    for (inti = 0; i<= n - m; ++i)

    {

        intj;

        // Порівнюємо кожен символ підрядка P з відповідним символом тексту S

        for (j = 0; j<m; ++j)

        {

            if (S[i + j] != P[j])

            {

                break; // Якщо знайдено неспівпадіння, виходимо з циклу

            }

        }

        // Якщо весь підрядок P співпадає з частиною тексту S, повертаємо позицію знайденого входження

        if (j == m)

        {

            returni;

        }

    }

    return -1; // Якщо збігів не знайдено

}

intmain()

{

    // Приклад використання

    std::vector<int>S = {1, 2, 3, 4, 5, 2, 7, 8};

    std::vector<int>P = {2, 7};

    // Виклик функції пошуку підрядка

    intresult = findSubstring(S, P);

    // Виведення результатів

    if (result != -1)

    {

        std::cout<<"Thefirstentryisfoundattheposition "<<result<<std::endl;

    }

    else

    {

        std::cout<<"Nomatchesfound"<<std::endl;

    }

    return0;

}

**Результат**



**Метод КМП:**

#include<iostream>

#include<vector>

// Функція для побудови таблиці префіксів-суфіксів (LPS)

std::vector<int>buildLPS(conststd::string&pattern)

{

    intm = pattern.length();   // Довжина паттерна

    std::vector<int>lps(m, 0); // Ініціалізація таблиці LPS нулями

    intlen = 0;                // Довжина поточного префіксу-суфіксу

    inti = 1;                  // Індекс для перегляду паттерна, починаючи з другого символу

    while (i<m)

    {

        // Перевірка, чи символ поточного індексу в паттерні збігається з попереднім символом

        if (pattern[i] == pattern[len])

        {

            len++;        // Збільшення довжини поточного префіксу-суфіксу

            lps[i] = len; // Зберігання довжини в таблиці LPS

            i++;          // Перехід до наступного символу паттерна

        }

        else

        {

            // Якщо символи не збігаються

            if (len != 0)

            {

                len = lps[len - 1]; // "Зсув" префіксу-суфіксу назад

            }

            else

            {

                lps[i] = 0; // Якщо довжина поточного префіксу-суфіксу 0, зберігаємо 0

                i++;        // Перехід до наступного символу паттерна

            }

        }

    }

    returnlps; // Повертаємо побудовану таблицю LPS

}

// Функція для пошуку підрядка (паттерна) в тексті за допомогою алгоритму КМП

intKMPSearch(conststd::string&text, conststd::string&pattern)

{

    intn = text.length();    // Довжина тексту

    intm = pattern.length(); // Довжина паттерна

    std::vector<int>lps = buildLPS(pattern); // Побудова таблиці LPS

    inti = 0; // Індекс для тексту

    intj = 0; // Індекс для паттерна

    while (i<n)

    {

        // Порівнюємо символи тексту та паттерна

        if (pattern[j] == text[i])

        {

            i++; // Збільшуємо індекс для тексту

            j++; // Збільшуємо індекс для паттерна

        }

        // Якщо весь паттерн збіг, повертаємо позицію знайденого входження

        if (j == m)

        {

            returni - j;

        }

        // Якщо символи не збігаються, використовуємо інформацію з таблиці LPS для зсуву паттерна

        elseif (i<n&&pattern[j] != text[i])

        {

            if (j != 0)

            {

                j = lps[j - 1]; // Зсув паттерна

            }

            else

            {

                i++; // Зсув тексту

            }

        }

    }

    return -1; // Паттерн не знайдено

}

intmain()

{

    std::stringtext = "I have a great day today";

    std::stringpattern = "day";

    // Виклик функції пошуку підрядка

    intresult = KMPSearch(text, pattern);

    // Виведення результатів

    if (result != -1)

    {

        std::cout<<"Thefirstoccurrenceofthepatternisfoundatposition "<<result<<std::endl;

    }

    else

    {

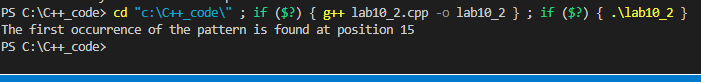
        std::cout<<"Patternnotfoundintext"<<std::endl;

    }

    return0;

}

**Результат**

****

**Висновок:** на цій лабораторній роботі я зостосував 2 методи пошуку прямий і КМП.А також розвинув навички реалізації та порівняння алгоритмів у контексті оптимізації пошукових процесів.