НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



**ЗВІТ**

**Про виконання лабораторної роботи № 11**

**з дисципліни «Алгоритми і структури даних»**

**Лектор:**

Доцент кафедри ПЗ

Коротєєва Т.О.

**Виконав:**

студ. групи ПЗ-23

Михалевич П.-І.В.

**Прийняла:**

Доцент кафедри ПЗ

Мельник Н.Б.

«\_\_\_» \_\_\_\_\_\_ 2021 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2021

**Тема роботи**: Алгоритм Кнута, Моріса і Прата.

**Мета роботи**: Навчитися застосовувати алгоритм Кнута, Моріса і Прата при пошуку слова в тексті.

**Теоретичні відомості**

На вхід надходять два масиви символів: S розміром n (текст) та P розміром m (словом). Необхідно знайти перше входження слова у тексті. Схема алгоритму полягає у поступовому порівнянні слова з текстом та у разі знайденого незбігу зсуву по тексту на крок, що дорівнює різницю кількості збіжних символів та значення відповідної префікс-функції. Алгоритм використовує просте спостереження, що коли є незбіг тексту і слова містить достатньо інформації для того, щоб визначити, де наступне входження може початися, пропускаючи кількаразову перевірку, попередньо порівняних символів. Попередньо досліджуєтеся слово та визначається префікс-функція.

Префікс-функція рядка – це довжина найбільшого префікс рядка S[1...i], який не збігається з цим рядком і одночасно є її суфіксом. Простіше кажучи, це довжина найдовшого початку рядка, що одночасно є і її кінцем. Позначимо таку функцію D(S,i). А крок зсуву P за S визначається за префікс функцією h-d-D(d), де d- кількість символів тексту, які збіглися, і слова при знайденому незбігу.

Час обчислення префікс-функції оцінюється як . Доведемо це.

Всі *i* діляться на:

1. Збільшують *k* на одиницю. Цикл проходить одну ітерацію.
2. Не змінюють нульове *k*. Цикл також проходить одну ітерацію. Випадків 1 і 2 в сумі не більше S - 1 штук.
3. Не змінюють або зменшують *k*. Оскільки всередині циклу значення *k* може тільки зменшуватися, а збільшення *k* можливо лише на одиницю, то сумарно значення *k* не може зменшитися більш, ніж на S - 2 рази, що і обмежує кількість спрацьовувань внутрішнього циклу.

Разом алгоритм вимагає не більше 2\*S ітерацій, що доводить порядок швидкості . «Найгіршим» для алгоритму є випадок обробки такі рядки 'aa ... ab'.

Час роботи даного алгоритму лінійно залежить від об’єму вхідних даних, тобто створити асимптотично ефективніший алгоритм неможливо. Він був розроблений у 1997 році Кнутом і Пратом, та незалежно від них Морісом.

Через те, що дві складові алгоритму мають складності, відповідно, і , складність всього алгоритму становить .

**Постановка завдання**

11. Задано два тексти. В першому тексті знайти слово, довжина якого дорівнює середньому значенню довжин всіх слів (в першому тексті). Знайдене слово знайти в другому тексті відповідним алгоритмом пошуку.

**Покроковий опис алгоритму лінійного пошуку:**

Покроковий опис префікс-функції:

ПФ 1. Ініціалізація: i = 0,j=1 D[0] = 0.

ПФ 2. Цикл: поки M < S, повторювати ПФ 3 - ПФ 5.

ПФ 3. Якщо S[i] = S[j], тоді i+=1, j+=1, добавити в D, i+1. Інакше: перейти до наступного кроку.

ПФ 4. Якщо i > 0, добавити в масив D, 0. Інакше: перейти до ПФ 5.

ПФ 5. Якщо i < 0, i = D[i-1].

ПФ 6. Кінець.

Покроковий опис алгоритму КМП:

КМП 1. Для заданого слова визначити префікс-функцію ПФ.

КМП 2. Встановити i=0,j=0.

КМП 3. Поки j<m,i<n. Перевірка: якщо S[i]=P[j],то j++ поки j !=m.

КМП 4. Інакше зсуваємо слово i=i + j - ПФ(j-1). Перейти на крок КМП 3.

КМП 5. Кінець.

**Код програми**

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QMessageBox>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

connect(ui->pushButton\_clear,*SIGNAL*(clicked()),*this*,*SLOT*(MySlot()));

connect(ui->pushButton\_start,*SIGNAL*(clicked()),*this*,*SLOT*(MySlot()));

ui->textEdit->setText("These modules was hard");

ui->textEdit\_text\_2->setText("I was in Paris these days and the day was hard");

}

*//* *Removes* *unnecessary* *spaces.*

std::string RemoveSpaces(std::string myString){

*for* (size\_t i = 1; i <= myString.size(); ++i) {

*if* (myString[i - 1] == ' ' && (myString[i] == ' ' || myString[i] == '\0')) {

myString.erase(myString.begin() + (i - 1));

myString.shrink\_to\_fit();

i--;

}

}

*return* myString;

}

void MainWindow::MySlot()

{

QPushButton\* btn = (QPushButton \*) sender();

*if*(btn->text() == "Start"){

*auto* begin = std::chrono::high\_resolution\_clock::now();

*if*(ui->textEdit->toPlainText() == "" || ui->textEdit\_text\_2->toPlainText() == ""){

QMessageBox::information(*nullptr*,"No input data","Enter text 1 and text 2!");

*return*;

}

QString line = ui->textEdit->toPlainText();

QString lowerline = line.toLower();

QStringList list = QString::fromStdString(RemoveSpaces(lowerline.toStdString())).split(' ');

int sizeSum = 0;

*for* (int i = 0;i < list.size(); i++) {

sizeSum += list[i].size();

}

*auto* wordSize = sizeSum/list.size();

ui->lineEdit\_size->setText(QString::number(wordSize));

QString word;

int index;

*for* (int i = 0;i < list.size(); i++) {

*if*(list[i].size() == wordSize){

word = list[i];

index = i;

ui->lineEdit\_word->setText(word);

ui->lineEdit\_word\_idx->setText(QString::number(index+1));

*break*;

}

}

word = " " + word + " ";

QString line\_2 = " " + ui->textEdit\_text\_2->toPlainText() + " ";

QString lowerline\_2 = line\_2.toLower();

QString strlowerline\_2 = QString::fromStdString(RemoveSpaces(lowerline\_2.toStdString())) + " ";

std::vector<int> table;

int i = 0;

int j = 1;

table.push\_back(0);

bool flag = *false*;

*while*(table.size() != word.size())

{

*if*(word[i] == word[j])

{

table.push\_back(i+1);

i++;

j++;

*continue*;

}

*else*

{

*if*(!i) {table.push\_back(0); j++;}

*else* i = table[i-1];

}

}

int d = 1;

int spaceCount = 0;

*for* (i = 0, j = 0;j < word.size() && i < strlowerline\_2.size(); ) {

*if*(strlowerline\_2[i] == ' '){

spaceCount++;

}

*if*(d != word.size()){

*if*(word[j] == strlowerline\_2[i]){

i++;

j++;

d++;

}*else*{

i += d-table[d-1];

j = 0;

d = 1;

}

}*else* *if*(d == word.size()){

flag = *true*;

*break*;

}

}

*if*(!flag){

ui->lineEdit\_word\_2\_idx->setText("Element not found");

}*else*{

ui->lineEdit\_word\_2\_idx->setText(QString::number(spaceCount));

}

*auto* end = std::chrono::high\_resolution\_clock::now();

*auto* elapsed = std::chrono::duration\_cast<std::chrono::nanoseconds>(end - begin);

ui->lineEdit\_task\_time->setText(QString::number(elapsed.count()));

}*else* *if*(btn->text() == "Clear"){

ui->textEdit->clear();

ui->textEdit\_text\_2->clear();

ui->lineEdit\_size->clear();

ui->lineEdit\_word\_2\_idx->clear();

ui->lineEdit\_word->clear();

ui->lineEdit\_word\_idx->clear();

ui->lineEdit\_find\_time->clear();

ui->lineEdit\_task\_time->clear();

}

}

MainWindow::~MainWindow()

{

*delete* ui;

}

**mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

*namespace* **Ui** { *class* **MainWindow**; }

QT\_END\_NAMESPACE

*class* **MainWindow** : *public* QMainWindow

{

Q\_OBJECT

*public*:

**MainWindow**(QWidget \*parent = *nullptr*);

~***MainWindow***();

*private* slots:

void **MySlot**();

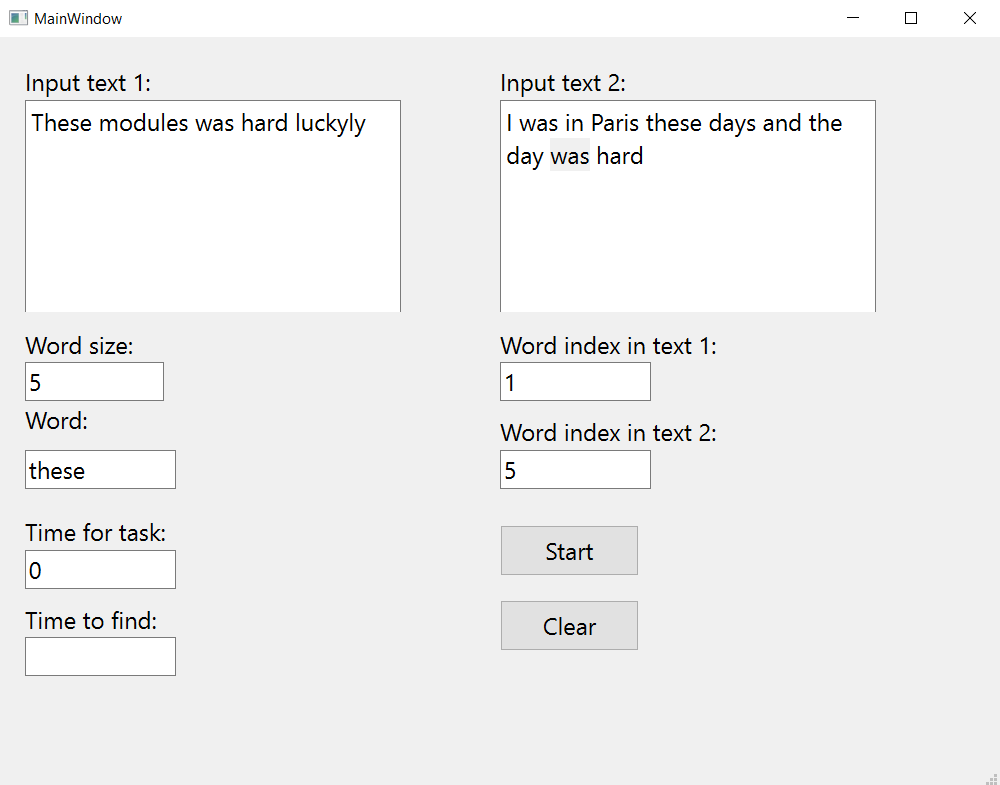
*private*:

Ui::MainWindow \*ui;

};

#endif *//* *MAINWINDOW\_H*

**Результат програми**

****

### Висновок

На цій лабораторній роботі я ознайомився з алгоритмом Кнута, Моріса та Прата. Час роботи даного алгоритму лінійно залежить від об’єму вхідних даних, тобто створити асимптотично ефективніший алгоритм неможливо. Через те, що дві складові агоритму мають складності, відповідно, і , де m – довжина слова, n – довжина тексту, складність всього алгоритму становить .