НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



**ЗВІТ**

**Про виконання лабораторної роботи № 12**

**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 Бойера Мура.

**Мета роботи**: Навчитися застосовувати алгоритм Бойера Мура при пошуку слова в тексті.

**Теоретичні відомості**

Aлгоритм – Бойера-Мура – оснований на схемі: порівняння символів позначається з кінця слова, а не з початку. Нехай для кожного символу  слова  – відстань від першого у слові входження до кінця слова. Припустімо, знайдено незбіг між словом та текстом на символі у тексті. Тоді слово можна посунути праворуч на позицій, що є більшою або дорівнює 1. Якщо у слові взагалі не зустрічається, то посунути слово можна зразу на всю довжину .

У цьому алгоритму символ розглядається як стоп-символ – це є символ у тексті, який є першим незбігом тексту і слова під час порівняння праворуч(з кінця слова). Розглянемо три можливі ситуації:

1. Стоп-символа у слові взагалі немає, тоді зсув є довжиною слова .
2. Крайня права позиція входження стоп-символа у слові є меншою від його позиції у тексті. Тоді слово можна зсунути праворуч на позицій так, щоб стоп-символ у слові і тексті опинились один під одним.
3. Крайня права позиція входження стоп-символа у слові є більшою від його позиції у тексті. Тоді якщо у слова є ще один такий символ, то необхідно зсунути слово до збігу цього символу з символом у тексті, інакше зсув дорівнює 1.

Для оптимізації пошуку пропонують попередньо сформувати таблицю стоп-символів, розмір якою відповідає кількості літер алфавіту, що використовуєтся в вхідному тексті. Початково всім елементів в таблиці надається значення  – кількість символів шуканого слова, для реалізації першої з можливих ситуацій зі стоп-символом.

У найгіршому випадку алгоритм Бойера-Мура потребує  порівнянь, де – кількість символів у тексті. У найкращих обставинах, коли останній символ слова завжди не збігається з символом тексту, кількість порівнянь дорівнює , де – кількість символів слова. Отже складнійсть алгоритму Бойера-Мура .

**Постановка завдання**

12. Дано текст що складається з n стрічок. Поміняти місцями i-ту та  n-1 –шу стрічки. Вивести всі індекси входження підстрічки T що зустрічається в тексті S в порядку зростання.

**Покроковий опис алгоритму лінійного пошуку:**

Покроковий опис для алгоритму побудови таблиці стоп-символів:

Алгоритм ST

Дано S[1..n] – стрічка, для якої будується таблиця стоп-символів; T[1..m] – словник для таблиці стоп-сиволів, де m – кількість символів алфавіту використаних у стрічці S, ключем є символ, а значенням індекс; i – індекс по S.

ST1. Повторювати крок 2, поки 1≤i≤n.

ST2. Присвоїти для T[S[i]] = i.

Алгоритм Бойера – Мура:

Алгоритм BM

Дано S[1..n] – стрічка, в якій відбувається пошук; P[1..m] – стрічка, входження якої у S необхідно найти; T[1..m] – таблиця зсувів; i – індекс по S; j – індекс по P; answer – масив входжень стрічки P у S.

BM1. Повторювати крок 2, поки 1≤i≤n.

BM2. Повторювати крок 3, 4, поки m≥j≥1.

BM3. Якщо S\_i дорівнює P\_j, то присвоїти j=j-1, інакше i=i+T[i+m].

BM4. Якщо j=0, то додати i до масиву answer. Присвоїти j=m.

BM5. Кінець. Вихід.

**Код програми**

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <vector>

#include <QMessageBox>

#include <map>

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()));

}

*//* *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;

}

int FindIndex(*const* QString line, int position){

int count = 0;

*for*(int i = position; i >= 0; --i){

*if*(line[i] == ' '){

count++;

}

}

*return* count;

}

void MainWindow::MySlot()

{

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

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

QStringList lineList = ui->textEdit\_text->toPlainText().toLower().split('.',Qt::SkipEmptyParts);

*for* (int i = 1; i < lineList.size() ; ++i) {

lineList[i].remove('\n');

}

*//Task*

*if*(ui->spinBox\_line\_idx->value() > lineList.size() || lineList.size() == 1){

QMessageBox::warning(*nullptr*,"Incorrect line index!","Text has only "

+ QString::number(lineList.size()) + " lines!\nPlease, change the line index.");

*return*;

}

QString tmp = lineList[lineList.size()-2];

lineList[lineList.size()-2] = lineList[ui->spinBox\_line\_idx->value()-1];

lineList[ui->spinBox\_line\_idx->value()-1] = tmp;

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

ui->textEdit\_change\_text->append(lineList[i]);

}

*//Search*

*if*(ui->lineEdit\_word->text() == ""){

QMessageBox::warning(*nullptr*,"Incorrect input!","Please, enter the word to find.");

*return*;

}

QString word = QString::fromStdString(RemoveSpaces((" " + ui->lineEdit\_word->text()).toLower().toStdString())) + " ";

*for*(*auto*&& line : lineList){

line = QString::fromStdString(RemoveSpaces((" " + line).toStdString())) + " ";

}

std::map<QChar,int> map;

map[word[word.size() - 1]] = word.size();

*for* (int i = 0; i < word.size() - 1 ; i++) {

map[word[i]] = word.size()-1-i;

}

*for*(int m = 0; m < lineList.size(); ++m){

*const* *auto* line = lineList[m];

bool l\_flag = *false*;

*for* (int i = word.size()-1; i < line.size();) {

int iCur = i;

bool flag = *false*;

*for*(int j = word.size()-1; j >= 0;){

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

*if*(j == 0){

flag = *true*;

*break*;

}

--j;

--i;

}*else*{

*auto* it = map.find(line[i]);

*if*(it == map.end()){

i = iCur + j;

*break*;

}

*if*(it->second < i){

i = iCur + it->second;

}*else*{

i = iCur + 1;

}

*break*;

}

}

*if*(flag){

ui->textEdit\_result->append("Found in line " + QString::number(m + 1) + " : " + QString::number(FindIndex(line, i)) + " word.");

i = i + word.size();

l\_flag = *true*;

}

}

*if*(!l\_flag){

ui->textEdit\_result->append("Word not found in line " + QString::number(m + 1) + ".");

}

}

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

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

ui->textEdit\_result->clear();

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

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

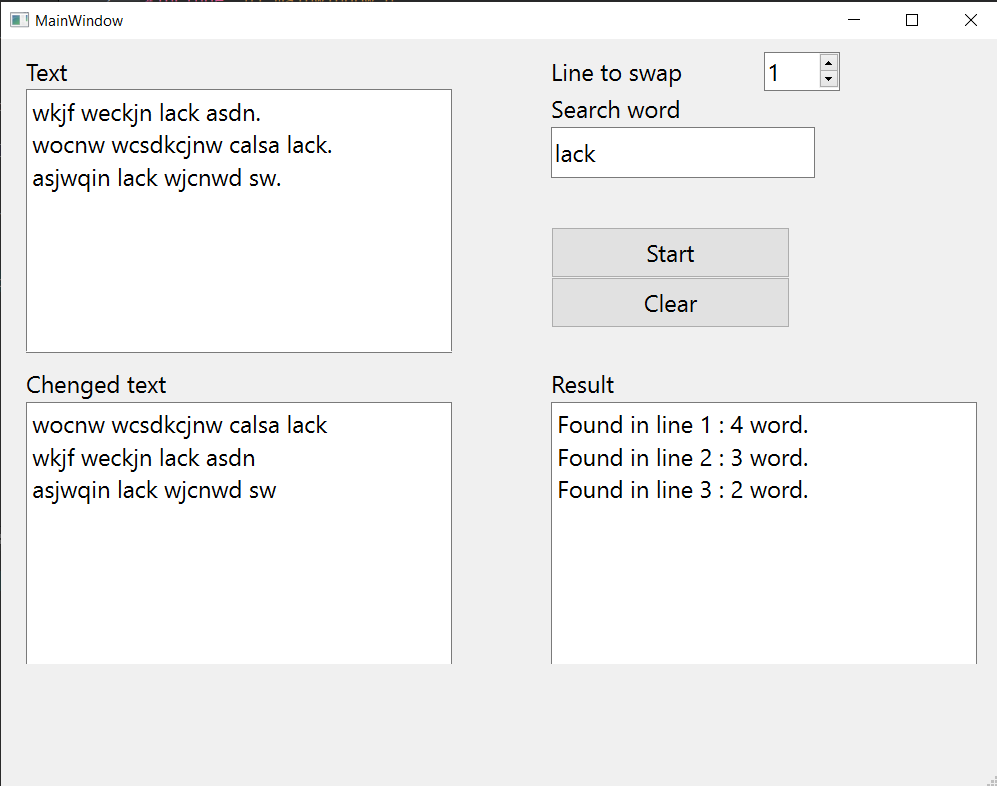
*private*:

Ui::MainWindow \*ui;

};

#endif *//* *MAINWINDOW\_H*

**Результат програми**

****

### Висновок

На цій лабораторній роботі яі ознайомився з алгоритмом Бойера-Мура. Ефективний алгоритм пошуку рядка, який є еталоном при практичних дослідженнях алгоритмів пошуку рядка. У найгіршому випадку алгоритм Бойера-Мура потребує порівнянь, де – кількість символів у тексті. У найкращих обставинах, коли останній символ слова завжди не збігається з символом тексту, кількість порівнянь дорівнює , де – кількість символів слова. Отже складнійсть алгоритму Бойера-Мура .