НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



**ЗВІТ**

**Про виконання лабораторної роботи № 6**

**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**: Метод сортування підрахунком.

**Мета роботи**: Вивчити алгоритм сортування підрахунком. Здійснити програмну реалізацію алгоритму сортування підрахунком. Дослідити швидкодію алгоритму.

**Теоретичні відомості**

Сортування підрахунком (англійською «Counting Sort») — алгоритм впорядкування, що застосовується при малій кількості різних елементів (ключів) у масиві даних. Час його роботи лінійно залежить як від загальної кількості елементів у масиві так і від кількості різних елементів.

Ідея алгоритму полягає в наступному: спочатку підрахувати скільки разів кожен елемент (ключ) зустрічається в вихідному масиві. Спираючись на ці дані можна одразу вирахувати на якому місці має стояти кожен елемент, а потім за один прохід поставити всі елементи на свої місця.

В алгоритмі присутні тільки прості цикли: довжини N (довжина масиву) та цикл довжини K (величина діапазону). Отже, обчислювальна складність роботи алгоритму становить *O*(*N* + *K*).

В алгоритмі використовується додатковий масив. Тому алгоритм потребує *E*(*K*) додаткової пам’яті. В такій реалізації алгоритм є стабільним. Саме ця його властивість дозволяє використовувати його як частину інших алгоритмів сортування. Використання даного алгоритму є доцільним тільки у випадку малих *K*.

**Покроковий опис роботи алгоритму**

***Алгоритм CS.***

Задано масив елементів *R1 ,R2 ,…,Rn.* Даний алгоритм реорганізує масив у висхідному порядку, тобто для його елементів буде мати місце співвідношення *Ri < Rj -* для всіх *i,j=1..n. ; Створюєм додатковий масив T1 ,T2 ,…,Tk , де k – к-сть різних значень в нашому діапазоні*.

CS1. *Записуєм в масив Т к-сть різних значень масиву R.*

CS2. Переписуєм R відповідно до Т.

CS3. Кінець. Вихід.

**Постановка завдання**

15. Задано матрицю дійсних чисел. Впорядкувати (переставити) рядки матриці за зростанням суми їх елементів.

**Код програми**

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#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\_init,SIGNAL(clicked()),*this*,SLOT(MySlot()));

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

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

void MainWindow::**MySlot**()

{

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

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

*for*(int i = 0; i < ui->tableWidget\_enter->rowCount(); i++)

{

*for*(int j = 0; j < ui->tableWidget\_enter->columnCount(); j++)

{

ui->tableWidget\_enter->item(i,j)->setText("");

*delete* ui->tableWidget\_enter->item(i,j);

}

}

*for*(int i = 0; i < ui->tableWidget\_result->rowCount(); i++)

{

*for*(int j = 0; j < ui->tableWidget\_result->columnCount(); j++)

{

ui->tableWidget\_result->item(i,j)->setText("");

*delete* ui->tableWidget\_enter->item(i,j);

}

}

ui->textEdit->clear();

ui->spinBox\_column->*clear*();

ui->spinBox\_column->setValue(ui->spinBox\_column->minimum());

ui->spinBox\_row->*clear*();

ui->spinBox\_row->setValue(ui->spinBox\_row->minimum());

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

ui->textEdit->clear();

int row\_count = ui->tableWidget\_enter->rowCount();

int column\_count = ui->tableWidget\_enter->columnCount();

double\*\* mas = *new* double\*[row\_count];

*for*(int i = 0; i < row\_count; i++)

{

mas[i] = *new* double[column\_count];

}

double\*\* sort\_mas = *new* double\*[row\_count];

*for*(int i = 0; i < row\_count; i++)

{

sort\_mas[i] = *new* double[column\_count];

}

*//Task*

double \* row\_index = *new* double[row\_count]{0};

double \* sort\_row\_index = *new* double[row\_count]{0};

*for*(int i = 0; i < row\_count; i++)

{

*for*(int j = 0; j < column\_count; j++)

{

mas[i][j] = ui->tableWidget\_enter->item(i,j)->text().toDouble();

row\_index[i] += mas[i][j];

}

}

*for*(int i = 0; i < row\_count; i++)

{

sort\_row\_index[i] = row\_index[i];

}

*//* *Count* *sort*

std::map<double,int> keys\_values;

*for*(int i = 0; i < row\_count; i++)

{

keys\_values[sort\_row\_index[i]]++;

}

*for*(*auto* it : keys\_values){

QString strRow = "";

strRow += QString::number(it.second) + " - " +QString::number(it.first,'f',1);

ui->textEdit->append(strRow);

}

int ind = 0;

*for*(*auto* it : keys\_values){

double val = it.first;

int freq = it.second;

*for* (int j = 0;j < freq ;j++ ) {

sort\_row\_index[ind] = val;

ind++;

}

}

ui->textEdit->append("");

*for*(int i = 0; i < row\_count; i++)

{

QString strRow = "";

strRow += QString::number(row\_index[i],'f',1) + " \\ " +QString::number(sort\_row\_index[i],'f',1);

ui->textEdit->append(strRow);

}

*for*(int i = 0; i < row\_count; i++){

*for* (int j = 0; j < row\_count ; j++ ) {

*if*(*static\_cast*<int>(row\_index[i]\*10) == *static\_cast*<int>(sort\_row\_index[j]\*10)){

*for* (int k = 0; k < column\_count ; k++ ) {

sort\_mas[j][k] = mas[i][k];

}

sort\_row\_index[j] = 0;

*break*;

}

}

}

ui->textEdit->append("");

*for*(int i = 0; i < row\_count; i++)

{

QString strRow = "";

strRow += QString::number(row\_index[i],'f',1) + " \\ " +QString::number(sort\_row\_index[i],'f',1);

ui->textEdit->append(strRow);

}

*for*(int i = 0; i < row\_count; i++)

{

*for*(int j = 0; j < column\_count; j++)

{

ui->tableWidget\_result->item(i,j)->setText(QString::number(sort\_mas[i][j],'f',1));

}

}

*for*(int i = 0; i < row\_count; i++)

{

*delete* [] mas[i];

}

*delete* [] mas;

*for*(int i = 0; i < row\_count; i++)

{

*delete* [] sort\_mas[i];

}

*delete* [] sort\_mas;

*delete* [] row\_index;

*delete* [] sort\_row\_index;

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

srand(time(NULL));

*for*(int i = 0; i < ui->tableWidget\_enter->rowCount(); i++)

{

*for*(int j = 0; j < ui->tableWidget\_enter->columnCount(); j++)

{

QTableWidgetItem\* item = *new* QTableWidgetItem;

double rand\_val = *static\_cast*<double>(rand() % 200) / 2.5;

item->setText(QString::number(rand\_val,'f',1));

item->setTextAlignment(Qt::*AlignCenter*);

ui->tableWidget\_enter->setItem(i,j,*item*);

}

}

*for*(int i = 0; i < ui->tableWidget\_result->rowCount(); i++)

{

*for*(int j = 0; j < ui->tableWidget\_result->columnCount(); j++)

{

QTableWidgetItem\* item = *new* QTableWidgetItem;

item->setText("");

item->setTextAlignment(Qt::*AlignCenter*);

ui->tableWidget\_result->setItem(i,j,*item*);

}

}

}

}

void MainWindow::**on\_spinBox\_row\_valueChanged**(int arg1)

{

ui->tableWidget\_enter->setRowCount(arg1);

ui->tableWidget\_result->setRowCount(arg1);

}

void MainWindow::**on\_spinBox\_column\_valueChanged**(int arg1)

{

ui->tableWidget\_enter->setColumnCount(arg1);

ui->tableWidget\_result->setColumnCount(arg1);

}

**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**();

void **on\_spinBox\_row\_valueChanged**(int arg1);

void **on\_spinBox\_column\_valueChanged**(int arg1);

*private*:

Ui::MainWindow \*ui;

};

#endif *//* *MAINWINDOW\_H*

**main.cpp**

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(*argc*, *argv*);

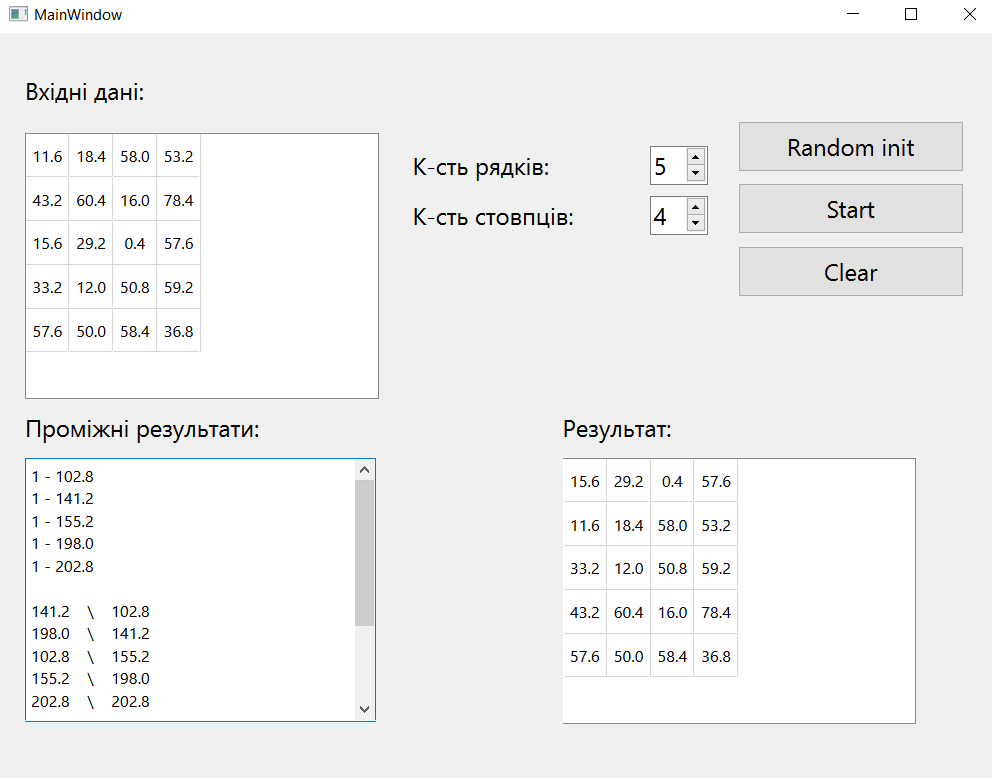
MainWindow w;

w.show();

*return* a.exec();

}

**Результат програми**



### Висновок

На даній лабораторній роботі я вивчив алгоритм сортування підрахунком, здійснив програмну реалізацію алгоритму сортування підрахунком, дослідив швидкодію алгоритму сортування підрахунком (складність методу О**.**