НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



**ЗВІТ**

**Про виконання лабораторної роботи № 4**

**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 злиттям.

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

**Теоретичні відомості**

Сортування злиттям (англійською «Merge Sort») — алгоритм сортування, в основі якого лежить принцип «розділяй та володарюй» (англійською «Divide and Conquer»). В основі цього способу сортування лежить злиття двох упорядкованих ділянок масиву в одну впорядковану ділянку іншого масиву.

Підчас сортування в дві допоміжні послідовності з основної поміщаються перші дві відсортовані підпослідовності, які потім зливаються в одну і результат записується в тимчасову послідовність. Потім з основної послідовності вибераються наступні дві відсортовані підпослідовності і так до тих пір доки основна послідовність не стане порожньою. Після цього послідовність з тимчасової переміщається в основну. І знову продовжується сортування злиттям двох відсортованих підпослідовностей. Сортування триватиме до тих пір, поки довжина відсортованої підпослідовності не стане рівною довжині самої послідовності.

Сортування злиттям можна задати рекурсивно: масив поділяється на дві приблизно рівні частини, які після сортування (тим самим способом) зливаються. Коли ж довжина масиву зменшується до 1, відбувається повернення з рекурсії.

Час роботи алгоритму *T*(*n*) по впорядкуванню *n* елементів задовільняє рекурентному співвідношенню: *T*(*n*) = 2∙*T*(½∙*n*) + *O*(*n*), де *T*(½∙*n*) — час на впорядкування половини масиву, *O*(*n*) — час на злиття цих половинок.

Враховуючи, що *T*(1) = *O*(1), розв’язком співвідношення є: *T*(*n*) = *O*(*n*∙log(*n*)).

Крім того, алгоритм потребує для своєї роботи *E*(*n*) додаткової пам’яті.

Алгоритм не міняє порядок розташування однакових елементів, а отже він є стабільним.

**Покроковий опис роботи алгоритму на етапі злиття**

***Алгоритм C***

*Дано X={ x1 ,…, xn }; Y={ y1 ,…, ym }. і - індекс для множини X , j - індекс для множини Y , k – індекс для множини Z, i=1..n; j=1..m; k=1..(n+m).*

*C1. Ініціалізація індексів i=1, j=1, k=1;*

*C2. Виконувати C3, C4 доки k<n+m.*

*С3 [Якщо xi < yi , то [ zk = xi; i=i+1], інакше [ zk=yj, j=j+1]*

*C4. k=k+1]*

*C5. Кінець. Вихід.*

**Постановка завдання**

13. Задано одномірний масив цілих чисел. Виключити з нього елементи, кратні трьом, а до всіх інших елементів масиву застосувати функцію . Отриманий масив посортувати в порядку спадання.

**Код програми**

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QMessageBox>

#include <QTextEdit>

#include <ctime>

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()));

}

void **merge**(int\* array,*const* int left,*const* int mid,*const* int right,QTextEdit\* textEdit)

{

*const* int subArrayOneCount = mid - left + 1;

*const* int subArrayTwoCount = right - mid;

int \*leftArray = *new* int[subArrayOneCount],

\*rightArray = *new* int[subArrayTwoCount];

*for* (int i = 0; i < subArrayOneCount; i++)

leftArray[i] = array[left + i];

*for* (int j = 0; j < subArrayTwoCount; j++)

rightArray[j] = array[mid + 1 + j];

int indexOfSubArrayOne = 0,

indexOfSubArrayTwo = 0;

int indexOfMergedArray = left;

*//* *Merge*

*while* (indexOfSubArrayOne < subArrayOneCount && indexOfSubArrayTwo < subArrayTwoCount) {

*if* (leftArray[indexOfSubArrayOne] <= rightArray[indexOfSubArrayTwo]) {

array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];

indexOfSubArrayOne++;

}

*else* {

array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];

indexOfSubArrayTwo++;

}

indexOfMergedArray++;

}

*while* (indexOfSubArrayOne < subArrayOneCount) {

array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];

indexOfSubArrayOne++;

indexOfMergedArray++;

}

*while* (indexOfSubArrayTwo < subArrayTwoCount) {

array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];

indexOfSubArrayTwo++;

indexOfMergedArray++;

}

textEdit->append("");

QString result = "";

*if*(array){

*for* (int i = left; i < right; i++ ) {

result += QString::number(array[i]);

*if*(i != right-1){

result += ", ";

}

}

}

textEdit->append(result);

*delete* [] leftArray;

*delete* [] rightArray;

}

void **mergeSort**(int\* array, *const* int begin,*const* int end, QTextEdit\* textEdit)

{

*if* (begin >= end)

*return*;

int mid = begin + (end - begin) / 2;

mergeSort(*array*, begin, mid, *textEdit*);

mergeSort(*array*, mid + 1, end, *textEdit*);

merge(*array*, begin, mid, end, *textEdit*);

}

void MainWindow::**MySlot**(){

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

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

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

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

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

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

}

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

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

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

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

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

srand(time(NULL));

QString str\_enter = "";

int count = (rand() % 10) + 5;

*for* (int i = 0; i < count; i++) {

int rand\_val = (rand() % 30);

str\_enter += QString::number(rand\_val);

*if*(i != count-1){

str\_enter += ", ";

}

}

ui->lineEdit\_enter->setText(str\_enter);

}

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

QString text = ui->lineEdit\_enter->text();

int count\_1 = 0;

*if*(!text.isEmpty()){

QStringList valuestr = text.split(',');

count\_1 = valuestr.size();

}

int\* array = *new* int[count\_1];

*if*(!text.isEmpty()){

QStringList valuestr = text.split(',');

*for*(int j = 0; j < count\_1; j++){

*if*(!valuestr[j].isEmpty()){

bool ok;

int value = valuestr[j].toInt(*&ok*);

*if*(ok){

array[j] = value;

}*else* QMessageBox::warning(NULL,"Warning!","Enter the numbers and comas!");

}

}

}*else* QMessageBox::warning(NULL,"Warning!","Enter the array");

*//* *task*

int count\_mod3 = 0;

*for* (int i = 0; i < count\_1; i++) {

*if*(array[i] % 3 == 0) count\_mod3++;

}

int count\_2 = count\_1 - count\_mod3;

int\* array\_2 = *new* int[count\_2];

int index = 0;

*for* (int i = 0; i < count\_1; i++) {

*if*(array[i] % 3 != 0){

array\_2[index] = array[i]\*array[i];

index++;

}

}

QString result = "";

*if*(array\_2){

*for* (int i = 0; i < count\_2; i++ ) {

result += QString::number(array\_2[i]);

*if*(i != count\_2-1){

result += ", ";

}

}

}

ui->lineEdit\_task->setText(result);

*//Merge* *sort*

mergeSort(*array\_2*,0,count\_2, *ui->textEdit\_steps*);

result = "";

*if*(array\_2){

*for* (int i = 0; i < count\_2; i++ ) {

result += QString::number(array\_2[i]);

*if*(i != count\_2-1){

result += ", ";

}

}

}

ui->lineEdit\_result->setText(result);

*delete* [] array;

*delete* [] array\_2;

}

}

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*

**main.cpp**

#include "mainwindow.h"

#include <QApplication>

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

{

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

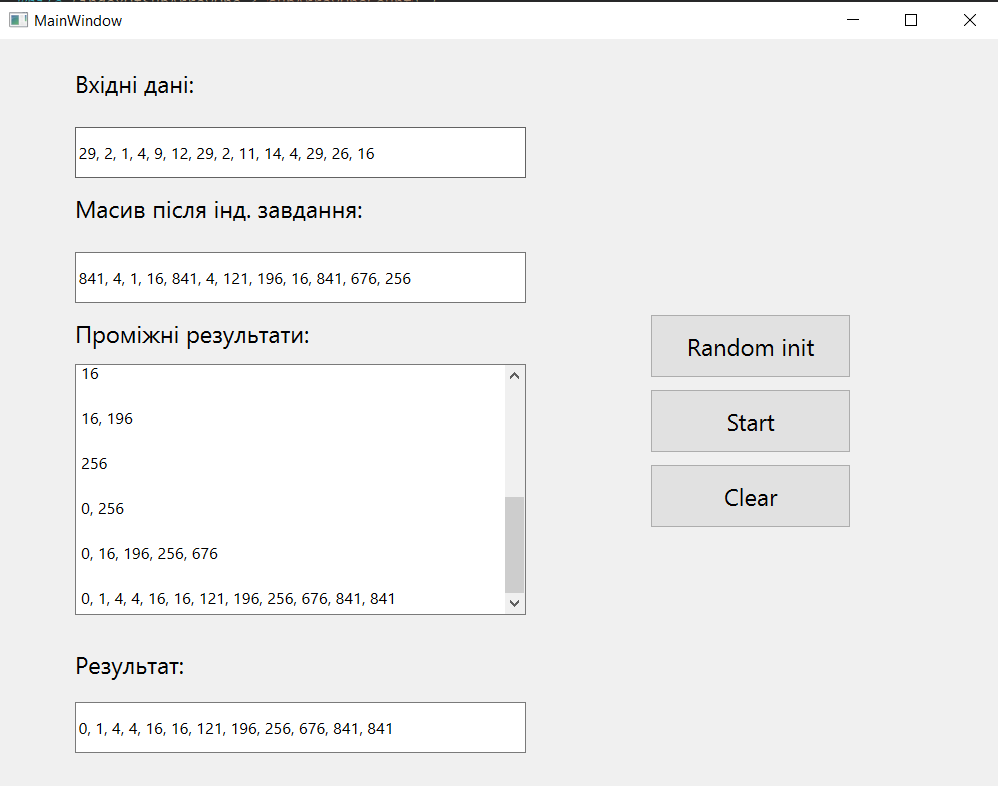
MainWindow w;

w.show();

*return* a.exec();

}

**Результат програми**



### Висновок

На даній лабораторній роботі я вивчив алгоритм сортування злиттям, здійснив програмну реалізацію алгоритму сортування злиттям, дослідив швидкодію алгоритму сортування злиттям (складність методу О(*n*∙log(*n*))).