МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут **ІКНІ**

Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи №4

На тему: «Метод швидкого сортування»

З дисципліни: «Алгоритми та структури даних»

Лектор : доцент каф.ПЗ

Коротєєва Т.О.

Виконала: ст.гр.ПЗ-23

Кохман О.В.

Прийняв: асистент каф.ПЗ

Франко А.В.

«____»_____2022 р.

∑____ _ _ _ _ _ _ _ _ .

Львів – 2022

Тема: Метод швидкого сортування.

Мета: Вивчити алгоритм швидкого сортування. Здійснити програмну реалізацію алгоритму швидкого сортування. Долідити швидкодію алгоритму швидкого сортування.

Теоретичні відомості

Швидке сортування (англійською «Quick Sort») — алгоритм сортування, добре відомий, як алгоритм розроблений Чарльзом Хоаром, який не потребує додаткової пам'яті і виконує у середньому $O(n \cdot \log(n))$ операцій. Оскільки алгоритм використовує дуже прості цикли і операції, він працює швидше інших алгоритмів, що мають таку ж асимптотичну оцінку складності.

В основі алгоритму лежить принцип «розділяй та володарюй» (англійською «Divide and Conquer»). Ідея алгоритму полягає в переставлянні елементів масиву таким чином, щоб його можна було розділити на дві частини і кожний елемент з першої частини був не більший за будь-який елемент з другої. Впорядкування кожної з частин відбувається рекурсивно. Алгоритм швидкого сортування може бути реалізований як на масиві, так і на двобічному списку.

Швидке сортування є алгоритмом на основі порівнянь, і не є стабільним.

Алгоритм швидкого сортування було розроблено Чарльзом Хоаром у 1962 році під час роботи у маленькій британській компанії Elliott Brothers.

В класичному варіанті, запропонованому Хоаром, з масиву обирався один елемент, і весь масив розбивався на дві частини по принципу: в першій частині — ті що не більші даного елементу, в другій частині — ті що не менші даного елемента.

Час роботи алгоритму сортування залежить від збалансованості, що характеризує розбиття. Збалансованість, у свою чергу залежить від того, який елемент обрано як опорний (відносно якого елемента виконується розбиття). Якщо розбиття збалансоване, то асимптотично алгоритм працює так само швидко як і алгоритм сортування злиттям.

- Найгірше розбиття. Найгірша поведінка має місце у тому випадку, коли процедура, що виконує розбиття, породжує одну підзадачу з (n – 1) елементом, а другу — з 0 елементами. Нехай таке незбалансоване розбиття виникає при кожному рекурсивному виклику. Для самого розбиття потрібен час $\Theta(n)$. Тоді рекурентне співвідношення для часу

роботи, можна записати наступним чином: $T(n) = T(n - 1) + T(0) + \Theta(n) = T(n - 1) + \Theta(n)$.

- Найкраще розбиття. В найкращому випадку процедура поділу ділить задачу на дві підзадачі, розмір кожної з яких не перевищує $(n / 2)$. Час роботи, описується нерівністю: $T(n) \leq 2 \cdot T(n / 2) + \Theta(n)$. Тоді: $T(n) = O(n \cdot \log(n))$ — асимптотично найкращий час.

Математичне очікування часу роботи алгоритм на всіх можливих вхідних масивах є $O(n \cdot \log(n))$, тобто середній випадок ближчий до найкращого.

В середньому алгоритм працює дуже швидко, але на практиці, не всі можливі вхідні масив мають однакову імовірність. Тоді, шляхом додання рандомізації вдається отримати середній час роботи в будь-якому випадку. В рандомізованому алгоритмі, при кожному розбитті в якості опорного обирається випадковий елемент.

Покроковий опис роботи алгоритму швидкого сортування:

Алгоритм QS:

Задано одновимірний масив array, індекс елемента, з якого починається сортування – left, індекс елемента, на якому закінчується сортування – right. Індекс i=left, індекс j = right, pivot = (left + right) /2 – ідекс середнього елемента масиву, з яким будуть порівнюватись array[i], array[j].

QS1: цикл, який триває доки i<=j, виконуютьс кроки QS2, QS3, QS4.

QS2: цикл, у якому шукається елемент, де array[i] > array[pivot].

QS3: цикл, у якому шукається елемент, де array[j] < array[pivot].

QS4: умова, де перевіряється чи i<=j, якщо так , то свапаємо елементи під індексами i,j, i+, j--.

QS5: умова , де перевіряється чи j > left, якщо так , то викликаємо функцію рекурсивно з новими межами – quickSort(array,left,j).

QS6: умова, де перевіряється чи i< right, якщо так, то викликаємо функцію рекурсивно з новими межами – quickSort(array,i,right).

QS7: вихід.

Індивідуальне завдання

1. Відвідати лекцію, вислухати та зрозуміти пояснення лектора. Прочитати та зрозуміти методичні вказівки, рекомендовані джерела та будь-які інші матеріали, що можуть допомогти при виконанні лабораторної роботи. Відвідати лабораторне заняття, вислухати та зрозуміти рекомендації викладача.

2. Встановити та налаштувати середовище розробки.

3. Написати віконний додаток на мові програмування C або C++. Реалізована програма повинна виконувати наступну послідовність дій:

1) запитуватиме в користувача кількість цілих чотирьохбайтових знакових чисел — елементів масиву, сортування якого буде пізніше здійснено;

2) виділятиме для масиву стільки пам’яті, скільки необхідно для зберігання вказаної кількості елементів, але не більше;

3) ініціалізовуватиме значення елементів масиву за допомогою стандартної послідовності псевдовипадкових чисел;

4) засікатиме час початку сортування масиву з максимально можливою точністю;

5) сортуватиме елементи масиву в неспадному порядку за допомогою алгоритму сортування Шелла;

6) засікатиме час закінчення сортування масиву з максимально можливою точністю;

7) здійснюватиме перевірку упорядкованості масиву;

8) повідомлятиме користувачу результат перевірки упорядкованості масиву та загальний час виконання сортування з максимально можливою точністю;

9) звільнятиме усю виділену раніше пам’ять.

Варіант 12: Задано одномірний масив дійсних чисел. Впорядкувати елементи, розташовані після максимального елемента в порядку спадання.

4. Оформити звіт про виконання лабораторної роботи. Звіт повинен бути надрукований з однієї сторони аркушів формату А4 шрифтом 12 кеглю з

одинарним інтерліньяжем та скріплений за допомогою степлера. Правильно оформлений звіт обов’язково повинен містити такі складові частини:

5. Захистити звіт про виконання лабораторної роботи. Процедура захисту передбачає демонстрацію роботи програми, перевірку оформлення звіту та відповіді на будь-яку кількість будь-яких запитань викладача, що так чи інакше стосуються теми лабораторної роботи.

Код програми

Назва файлу: MyForm.h

```
#pragma once
#include "Sort.h"
namespace Project4 {
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
		}

	protected:
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^ button1;
	protected:
	private: System::Windows::Forms::TextBox^ textBox1;
	private: System::Windows::Forms::TextBox^ textBox2;
	private: System::Windows::Forms::RichTextBox^ richTextBox1;
	private: System::Windows::Forms::TextBox^ textBox3;
	private: System::Windows::Forms::TextBox^ textBox4;
	private: System::Windows::Forms::TextBox^ textBox5;
	private: System::Windows::Forms::TextBox^ textBox6;
	private: System::Windows::Forms::Label^ label1;
	private: System::Windows::Forms::Label^ label2;
	private: System::Windows::Forms::Label^ label3;
	private: System::Windows::Forms::Label^ label4;
	private: System::Windows::Forms::Label^ label5;
	private: System::Windows::Forms::Label^ label6;
	private:
		System::ComponentModel::Container ^components;
#pragma region Windows Form Designer generated code
		// another code //

#pragma endregion
```

```
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
	int size = System::Convert::ToInt64(textBox1->Text);
	Sort sort = Sort(size);
	sort.randomNumbers();
	for (int i = 0; i < sort.length; i++) {
		textBox2->Text += (sort.array[i]).ToString("#,0.00") + " ";
	}
	DateTime start = DateTime::Now;
	textBox3->Text = start.ToString("hh.mm.ss.fff tt");
	quickSort(richTextBox1, sort.array, sort.findIndexMax(), size - 1);
	for (int i = 0; i < sort.length; i++) {
		richTextBox1->Text += (sort.array[i]).ToString("#,0.00") + " ";
	}
	richTextBox1->Text += " --------final";
	DateTime end = DateTime::Now;
	textBox4->Text = end.ToString("hh:mm:ss.fff tt");
	TimeSpan inverval = end - start;
	textBox5->Text = inverval.Seconds.ToString() + "." +
inverval.Milliseconds.ToString();
	sort.isOrdered();
	textBox6->Text = sort.getIsChecked().ToString();
}
};
}
```

Назва файлу: MyForm.cpp

```
#include "MyForm.h"
using namespace Project4;
int main() {
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);
	Application::Run(gcnew MyForm());
	return 0;
}
```

Назва файлу: Sort.h

```
#ifndef SORT_H
#define SORT_H
#pragma once
#include <random>
using namespace std;
class Sort {
public:
	int length;
	double* array;
	bool isChecked = true;
public:
	Sort();
	Sort(int _length);
	~Sort();
	void randomNumbers();
	int findIndexMax();
	friend void quickSort(System::Windows::Forms::RichTextBox^ richTextBox,
double* array, int left, int right);
	void isOrdered();
	bool getIsChecked();;
};
#endif
```

Назва файлу: Sort.cpp

```
#include "Sort.h"
Sort::Sort() {
	length = 0;
	array = new double[length];
};
Sort::Sort(int _length) {
	length = _length;
	array = new  double[length];
}
Sort::~Sort() {
	delete[] array;
}
void Sort::randomNumbers() {
	random_device random_device;
	mt19937 generator(random_device());
	uniform_real_distribution<double> distribution(0, 200);
	for (int i = 0; i < length; i++) {
		array[i] = distribution(generator);
	}
}
void Sort::isOrdered() {
    for (int i = this->findIndexMax() + 1 ; i < length; i++) {
        if (array[i - 1] < array[i]) {
            isChecked = false;
            break;
        }
    }
}
bool Sort::getIsChecked() {
	return isChecked;
}
int Sort::findIndexMax() {
    int count = 0;
    int max = array[0];
    for (int i = 0; i < length; i++) {
        if (array[i] > max) {
            max = array[i];
            count = i;
        }
    }
    return count;
}
int counter = 0;
void quickSort(System::Windows::Forms::RichTextBox^ richTextBox, double* array,
int left, int right) {
    int pivot = (left + right) / 2;
    int i = left, j = right;
    while (i <= j) {
        while (array[i] > array[pivot]) {
            i++;
        }
        while (array[j] < array[pivot]) {
            j--;
        }
        if (i <= j) {
            double temp = array[i];
            array[i] = array[j];
            array[j] = temp;
            i++, j--;
            for (int k = 0; k <= right; k++) {
```

```
                richTextBox->Text += array[k].ToString("#,0.00") + " ";
            }
            richTextBox->Text += System::Convert::ToString("-------step " +
counter + " \n");
            counter++;
        }
        if (j > left) {
            quickSort(richTextBox, array, left, j);
        }
        if (i < right) {
            quickSort(richTextBox, array, i, right);
        }
    }
}
```

Протокол роботи

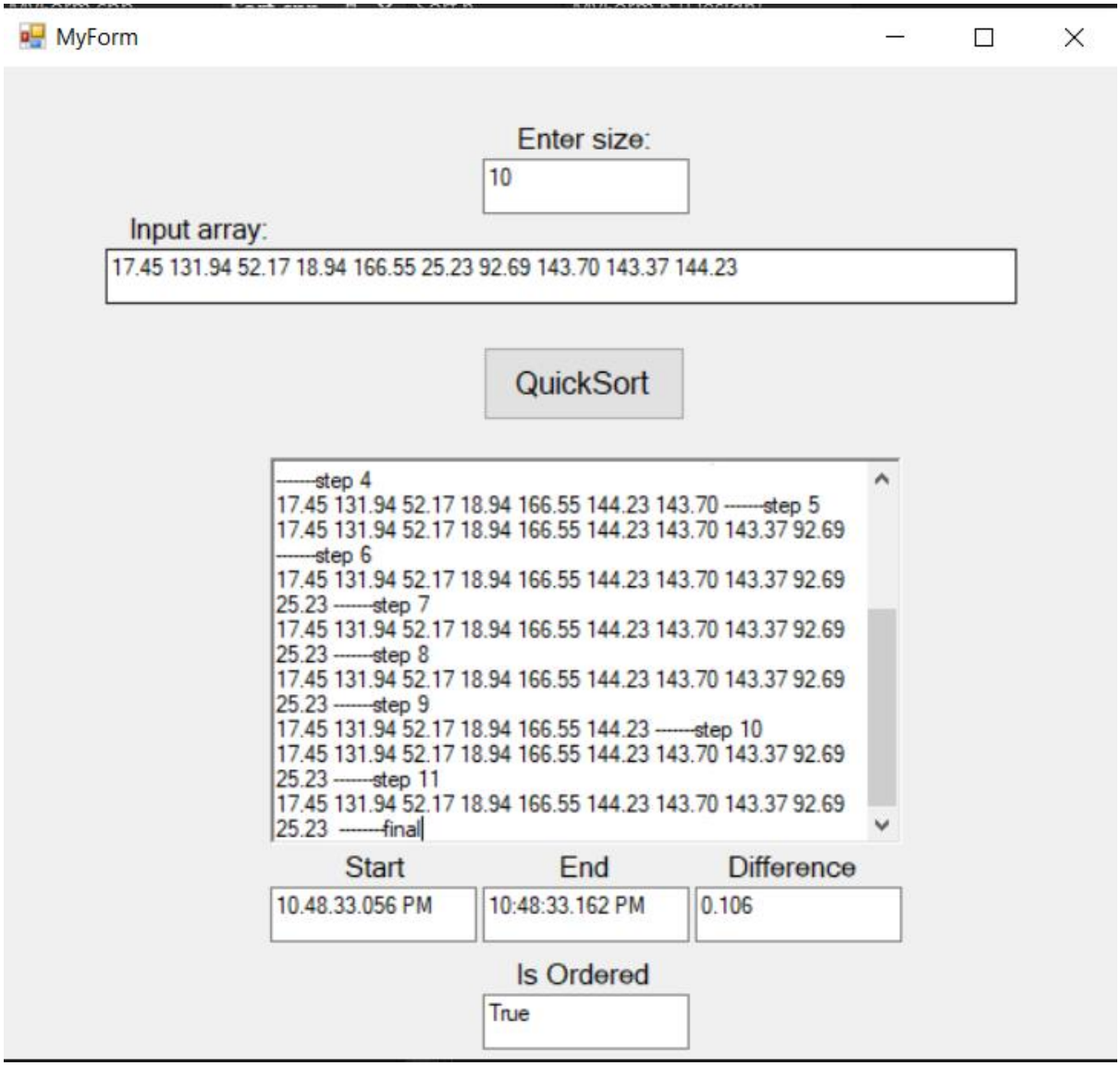

Рис. 1 Форма програми з результатами

Висновок

На цій лабораторній роботі я дізналась про алгоритм швидкого сортування, реалізувала його у Visual Studio 2022 та продемонструвала результати за допомогою форми. Також дізналась про швидкодію алгориму: у найкращому випадку – O(n*log n), у середньому – O(n* log n) , у найгіршому – O(n^2);