МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут **ІКНІ**

Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи №5

На тему: «Метод сортування злиттям»

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 сортування злиттям.

Мета: 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.

Під час сортування в дві допоміжні черги з основної поміщаються перші дві відсортовані підпослідовності, які потім зливаються в одну і результат записується в тимчасову чергу. Потім з основної черги беруться наступні дві відсортовані підпослідовності і так до тих пір доки основна черга не стане порожньою. Після цього послідовність з тимчасової черги переміщається в основну чергу. 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 частини масиву зменшується до 1, відбувається повернення з рекурсії. Завершуючи описання сортування злиттям, скажемо, що цей алгоритм є першим із ефективних алгоритмів сортування. У 1945 році його винайшов Джон фон Нейман, один із піонерів програмування.

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

Враховуючи, що $T(1) = O(1)$, розв'язком співвідношення є: $T(n) = O(n \cdot \log(n))$.

Крім того, алгоритм потребує для своєї роботи $E(n)$ додаткової пам'яті. Алгоритм не міняє порядок розташування однакових елементів, а отже він є стабільним. Швидкість алгоритму є асимптотично оптимальною. Але її можна пришвидшити в константну кількість разів.Можливі оптимізаці:

- Оптимізація впорядкування невеликих частин масиву — невеликі частини масиву (наприклад, при $n < 50$) впорядковувати сортуванням вставкою.
- Оптимізація кількості копіювань елементів — при злитті двох впорядкованих масивів в один, кожен елемент копіюється два рази (спочатку у тимчасовий масив, а потім знову у початковий). Кількість копіювань можна зменшити удвічі, якщо по черзі використовувати для об'єднання початковий і тимчасовий масиви. Тоді можна не виконувати зайві операції копіювання.

Покроковий опис алгоритму сортування злиттям:

Алгоритм MS:

Задано одновимірний масив цілих чисел array, left – перший елемент, right – останній елемент.

MMS1: Визначаємо middle = (left + (right -1eft)) / 2.

MMS2: Рекурсивно викликаємо MMS від left до middle.

MMS3: Рекурсивно викликаємо MMS від middle + 1 до right.

MMS4: Викликаємо MS, яка зливає підмасиви від left до middle і від middle до left

Алгоритм MS:

Задано одвимірний масив цілих чисел array, left – перший елемент, middle - середина , right – кінець.

MS1: Визначаємо size1 = middle + left -1 , size2 = right – middle. Створюємо нові масиви Left = new int[size1], Right = new int[size2].

MS2: Цикл, який триває при умові i<size1, виконуємо Left[i] = array[left + i], після кожної ітерації збільшуємо і на 1.

MS3: Цикл, який триває при умові j<size2, виконуємо Right[j] = array[middle + 1 +j], після кожної ітерації збільшуємо j на 1.

MS4: Оголошуємо i = 0, j = 0, k = left.

MS5: Цикл, який триває поки i<size1, j<size2. Перевіряємо умову, якщо

Left[i] <=Right[j], то array[k] = Left[i], i++. Якщо умова не виконується, то array[k] = Right[j], j++.

MS6: Цикл, який триває при умові i<size1 та виконує array[k] = Left[i]. Збільшуємо i на 1 та k на 1 після кожного виконання циклу.

MS7: Цикл, який триває при умові j<size2 та виконує array[k] = Right[j]. Збільшуємо j на 1 та k на 1 після кожного виконання циклу.

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 знак

вих 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 злиттям;

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.

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.

13 варіант:

Задано одномірний масив цілих чисел. Виключити з нього елементи, кратні трьом, а до всіх інших елементів масиву застосувати функцію x^2.

Код програми

Назва файлу: MyForm.cpp

```
#include "MyForm.h"
using namespace Project5;
int main() {
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);
	Application::Run(gcnew MyForm());
	return 0;
}
```

Назва файлу: MyForm.h

```
#pragma once
#include "Sort.h"
namespace Project5 {
	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::Label^ label6;
	protected:
	private: System::Windows::Forms::Label^ label5;
	private: System::Windows::Forms::Label^ label4;
	private: System::Windows::Forms::Label^ label3;
	private: System::Windows::Forms::Label^ label2;
	private: System::Windows::Forms::Label^ label1;
	private: System::Windows::Forms::TextBox^ textBox6;
	private: System::Windows::Forms::TextBox^ textBox5;
	private: System::Windows::Forms::TextBox^ textBox4;
	private: System::Windows::Forms::TextBox^ textBox3;
	private: System::Windows::Forms::RichTextBox^ richTextBox1;
	private: System::Windows::Forms::TextBox^ textBox2;
	private: System::Windows::Forms::TextBox^ textBox1;
	private: System::Windows::Forms::Button^ button1;
	private:
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated 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") + " ";
		}
		sort.delete3();
		sort.pow2();
		DateTime start = DateTime::Now;
		textBox3->Text = start.ToString("hh.mm.ss.fff tt");
		mergeSort(richTextBox1, sort.resultArray, 0, sort.length - 1 -
sort.count);
		richTextBox1->Text += "\n\nFinal result:\n";
		for (int i = 0; i < sort.length - sort.count; i++) {
			richTextBox1->Text += (sort.resultArray[i]).ToString("") + "
";
		}
		richTextBox1->Text += "\n-------- 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: Sort.h

```
#ifndef SORT_H
#define SORT_H
#pragma once
#include <iostream>
#include <random>
void mergeSort(System::Windows::Forms::RichTextBox^ richTextBox, int arr[], int
l, int r);
void merge(System::Windows::Forms::RichTextBox^ richTextBox, int arr[], int l,
int m, int r);
void printArray(System::Windows::Forms::RichTextBox^ richTextBox, int array[],
int length);
using namespace std;
class Sort {
public:
	int length;
	int* array;
	bool isChecked = true;
	int* resultArray;
	int count = 0;
public:
	Sort();
	Sort(int _length);
	~Sort();
	void randomNumbers();
	void delete3();
	void pow2();
	void isOrdered();
	bool getIsChecked();
};
#endif
```

Назва файлу: Sort.cpp

```
#include "Sort.h"
Sort::Sort() {
    length = 0;
    array = new int[length];
};
Sort::Sort(int _length) {
    length = _length;
    array = new  int[length];
}
Sort::~Sort() {
    delete[] array;
}
void Sort::randomNumbers() {
    random_device random_device;
    mt19937 generator(random_device());
    uniform_int_distribution<> distribution(1, 20);
    for (int i = 0; i < length; i++) {
        array[i] = distribution(generator);
    }
}
bool Sort::getIsChecked() {
    return isChecked;
}
void Sort::delete3() {
    ;
    for (int i = 0; i < length; i++) {
        if (array[i] % 3 == 0) {
            array[i] = -1;
```

```
            count++;
        }
    }
    resultArray = new int[length - count];
    int j = 0;
    for (int i = 0; i < length; i++) {
        if (array[i] != -1) {
            resultArray[j] = array[i];
            j++;
        }
    }
}
void Sort::pow2() {
    for (int i = 0; i < length - count; i++) {
        resultArray[i] = resultArray[i] * resultArray[i];
    }
}
void Sort::isOrdered() {
    for (int i = 1; i < length - count; i++) {
        if (resultArray[i - 1] > resultArray[i]) {
            isChecked = false;
        }
    }
}
int step = 1;
void mergeSort(System::Windows::Forms::RichTextBox^ richTextBox, int arr[], int
l, int r) {
    if (l >= r) {
        return;//returns recursively
    }
    int m = l + (r - l) / 2;
    richTextBox->Text += "Step " + step + ". Split the following array into 2
subarrays:" + "\n";
    for (int i = l; i < r + 1; i++) {
        richTextBox->Text += arr[i];
        richTextBox->Text += " ";
    }
    richTextBox->Text += "\n";
    richTextBox->Text += "The first subarray:\n";
    for (int i = l; i < m + 1; i++) {
        richTextBox->Text += arr[i];
        richTextBox->Text += " ";
    }
    richTextBox->Text += "\n" + "The second subarray:" + ":\n";
    for (int i = m + 1; i < r + 1; i++) {
        richTextBox->Text += arr[i];
        richTextBox->Text += " ";
    }
    richTextBox->Text += "\n";
    step++;
    mergeSort(richTextBox, arr, l, m);
    mergeSort(richTextBox, arr, m + 1, r);
    merge(richTextBox, arr, l, m, r);
}

void merge(System::Windows::Forms::RichTextBox^ richTextBox, int array[], int
left, int middle, int right) {
    int size1 = middle - left + 1;
    int size2 = right - middle;
    int* L = new int[size1], * R = new int[size2];
    for (int i = 0; i < size1; i++) {
        L[i] = array[left + i];
    }
```

```
    for (int j = 0; j < size2; j++) {
        R[j] = array[middle + 1 + j];
    }
    int i = 0;
    int j = 0;
    int k = left;
    while (i < size1 && j < size2) {
        if (L[i] <= R[j]) {
            array[k] = L[i];
            i++;
        }
        else {
            array[k] = R[j];
            j++;
        }
        k++;
    }
    while (i < size1) {
        array[k] = L[i];
        i++;
        k++;
    }
    while (j < size2) {
        array[k] = R[j];
        j++;
        k++;

    }
    richTextBox->Text += "Step " + step + ". Merge 2 subarrays. The first
subarray:" + ":\n";
    printArray(richTextBox, L, size1);
    richTextBox->Text += "The second subarray:" + "\n";
    printArray(richTextBox, R, size2);
    richTextBox->Text += "Merge result:\n";
    for (int i = left; i < right + 1; i++) {
        richTextBox->Text += array[i];
        richTextBox->Text += " ";
    }
    richTextBox->Text += "\n";
    step++;
}
void printArray(System::Windows::Forms::RichTextBox^ richTextBox, int array[],
int length) {
    for (int i = 0; i < length; i++) {
        richTextBox->Text += array[i];
        richTextBox->Text += " ";
    }
    richTextBox->Text += "\n";
}
```

Протокол роботи

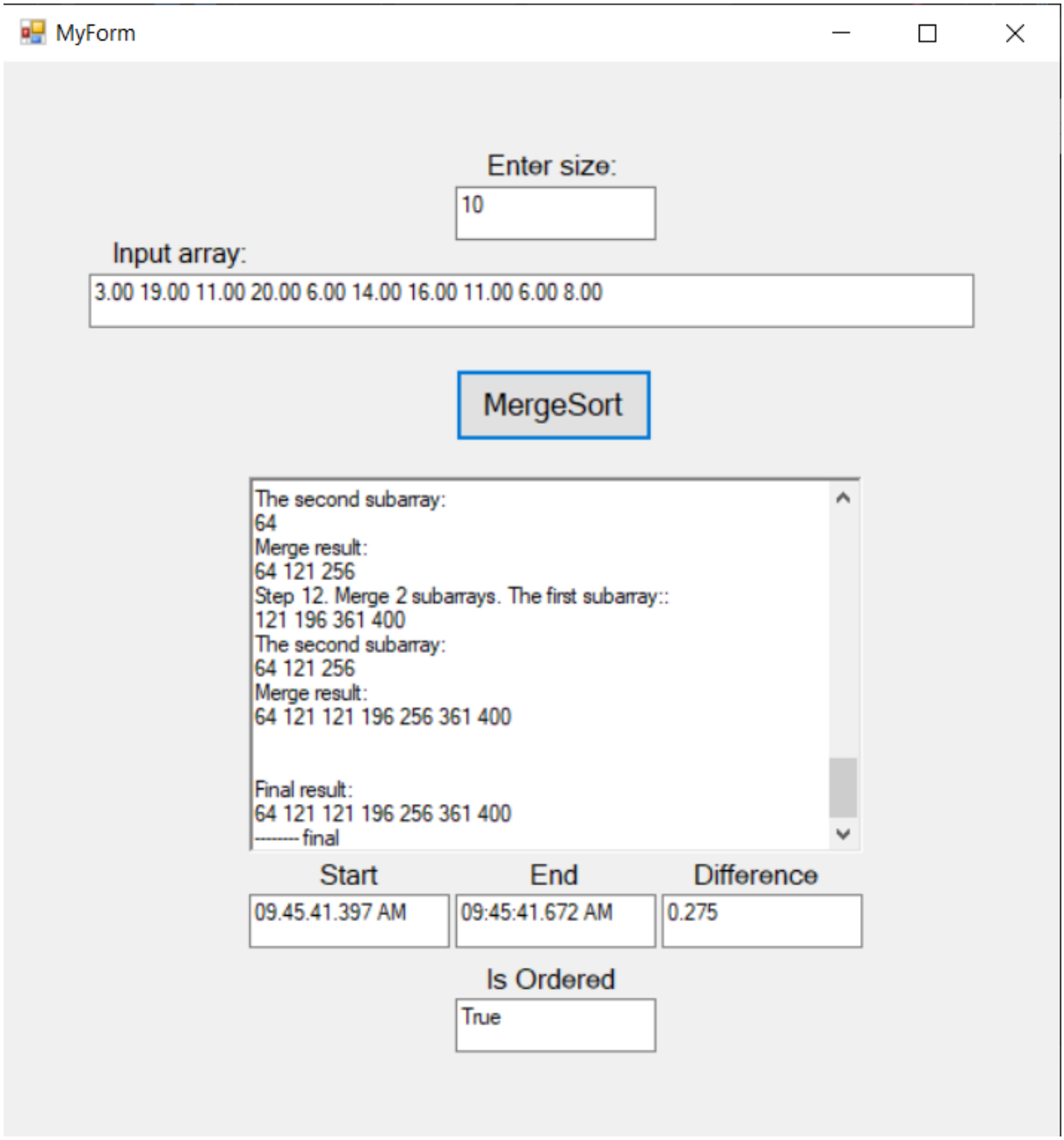

Рис. 1 Результат виконання програми у Visual Studio.

Висновок

На цій лабораторній роботі я ознайомилась із сортуванням злиттям, а також реалізувала його за допомогою c++ та продемонструвала результат роботи на формі у Visual Studio 2022. Дослідила швидкодію алгоритму, що дорівнює O (n * log n) в усіх випадках.