Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 2

з дисципліни «Алгоритми та методи обчислення»

на тему «Обчислювальна складність алгоритмів сортування»

ВИКОНАВ:

Студент І курсу ФІОТ

групи ІО-91

Діденко Владислав Віталійович

Варіант – 9

ПЕРЕВІРИВ:

ст.вик. Порєв В. М.

Київ – 2021

**Мета:** Закріплення навичок практичної оцінки алгоритмічної складності логічних алгоритмів на прикладі алгоритмів сортування.

**Завдання:** Використовуючи відповідний до варіанту алгоритм сортування написати програму сортування масиву даних. Застосовуючи дану програму, дослідити часову складність алгоритму сортування та порівняти її з теоретичною алгоритмічною складністю.



**Текст програми:**

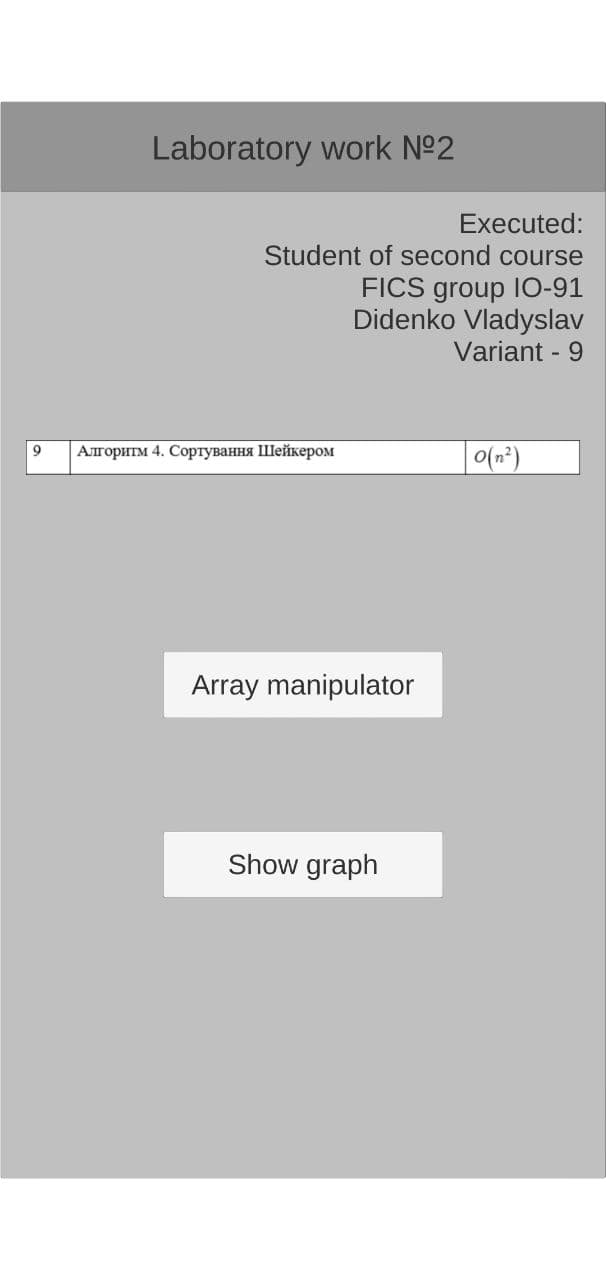
CanvasController.cs

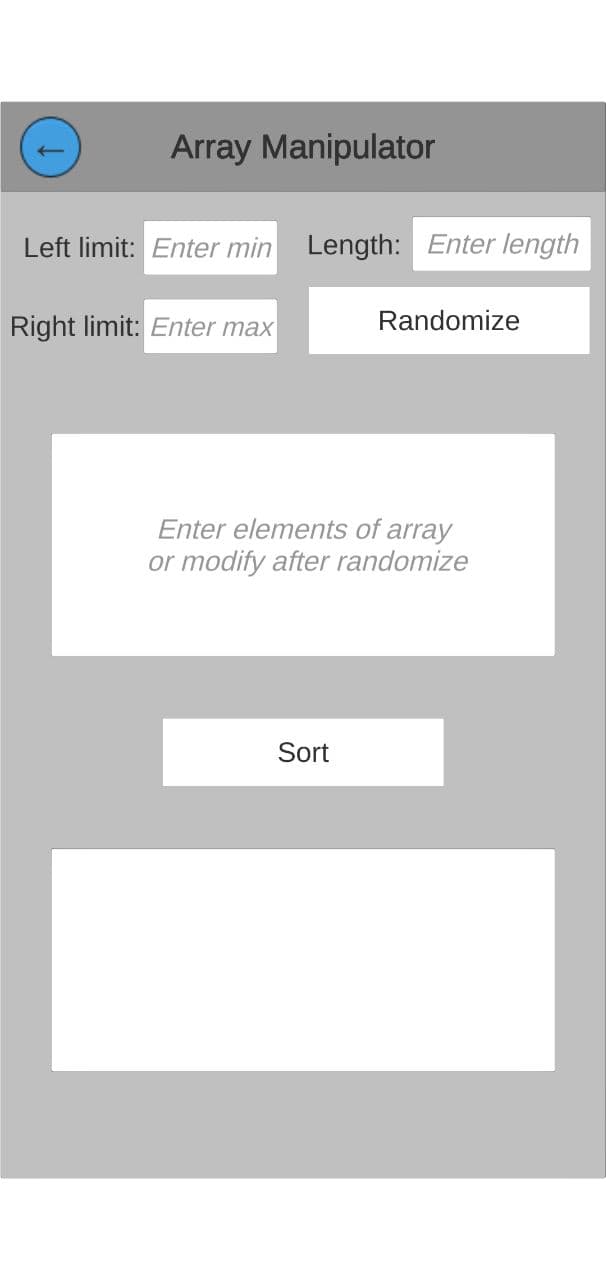
using System;  
using System.Collections;  
using System.Collections.Generic;  
using System.Globalization;  
using System.IO;  
using System.Linq;  
using Classes;  
using UnityEngine;  
using UnityEngine.UI;  
using Random = UnityEngine.Random;  
  
public class CanvasController : MonoBehaviour {  
 public static CanvasController instance;  
 private const float ERROR\_MESSAGE\_TIME = 2f;  
 private const string INPUT\_ERROR\_MESSAGE = "Wrong input";  
 private const int NUM\_OF\_DIGITS = 4;  
 private const string FILE\_NAME = "Sort analyse.txt";  
 public GameObject info;  
 public GameObject arrayManipulator;  
 public GameObject graphContainer;  
 public GameObject error;  
 public Image errorImage;  
 public Text errorText;  
 public InputField inputFieldLength;  
 public InputField inputFieldMin;  
 public InputField inputFieldMax;  
 public InputField inputFieldArray;  
 public Text sortedArray;  
 public Image graph;  
 public Sprite[] graphs;  
 private int currentGraphIndex;  
 private Coroutine errorCoroutine;  
  
  
 private void ShowErrorMessage(string message) {  
 if (errorCoroutine != null) {  
 StopCoroutine(errorCoroutine);  
 }  
  
 errorCoroutine = StartCoroutine(\_ShowErrorMessage(message));  
 }  
  
 private IEnumerator \_ShowErrorMessage(string message) {  
 error.SetActive(true);  
 errorText.text = message;  
 var time = 0f;  
 while (time < ERROR\_MESSAGE\_TIME) {  
 errorImage.color = Color.Lerp(Color.red, Color.clear, time / ERROR\_MESSAGE\_TIME);  
 time += Time.deltaTime;  
 yield return null;  
 }  
  
 error.SetActive(false);  
 errorCoroutine = null;  
 }  
  
 public void RaiseAndShowError(string message) {  
 sortedArray.text = string.Empty;  
 ShowErrorMessage(message);  
 }  
  
 private void Awake() {  
 instance = this;  
 CultureInfo.CurrentCulture = CultureInfo.GetCultureInfo("en-US");  
 }  
  
 private void Start() {  
 SetDefault();  
 // FindInfoForGraph();  
 }  
  
 public void SetDefault() {  
 info.SetActive(true);  
 arrayManipulator.SetActive(false);  
 graphContainer.SetActive(false);  
 }  
  
 private void OnClick(GameObject other) {  
 info.SetActive(false);  
 other.SetActive(true);  
 }  
  
 public void OnArrayManipulator() {  
 OnClick(arrayManipulator);  
 }  
  
 public void OnShowGraph() {  
 OnClick(graphContainer);  
 }  
  
  
 public void OnNextGraph() {  
 if (currentGraphIndex + 1 < graphs.Length) {  
 graph.sprite = graphs[++currentGraphIndex];  
 }  
 }  
  
  
 public void OnPrevGraph() {  
 if (currentGraphIndex - 1 >= 0) {  
 graph.sprite = graphs[--currentGraphIndex];  
 }  
 }  
  
 public void OnRandomize() {  
 if (!long.TryParse(inputFieldLength.text, NumberStyles.Float, CultureInfo.InvariantCulture, out var len) ||  
 !float.TryParse(inputFieldMin.text, NumberStyles.Float, CultureInfo.InvariantCulture, out var min) ||  
 !float.TryParse(inputFieldMax.text, NumberStyles.Float, CultureInfo.InvariantCulture, out var max) ||  
 min >= max ||  
 len <= 0) {  
 RaiseAndShowError(INPUT\_ERROR\_MESSAGE);  
 return;  
 }  
  
 var array = RandomizeArray(len, min, max);  
 UpdateArrayText(array);  
 }  
  
  
 private IEnumerable<float> RandomizeArray(long len, float min, float max) {  
 var arr = new float[len];  
 for (var i = 0; i < arr.Length; i++) {  
 arr[i] = (float) Math.Round(Random.Range(min, max), NUM\_OF\_DIGITS);  
 }  
  
 sortedArray.text = string.Empty;  
 return arr;  
 }  
  
  
 private void UpdateArrayText(IEnumerable<float> array) {  
 inputFieldArray.text = GetStringFromArray(array);  
 }  
  
  
 public void OnSortArray() {  
 var splited = inputFieldArray.text.Split(',').ToList();  
 var array = new float[splited.Count];  
 if (splited.Where((t, i) => !float.TryParse(t, NumberStyles.Float, CultureInfo.InvariantCulture, out array[i]))  
 .Any()) {  
 RaiseAndShowError(INPUT\_ERROR\_MESSAGE);  
 return;  
 }  
  
 array.CocktailSort(out var numOfOperations);  
  
 sortedArray.text = GetStringFromArray(array);  
 }  
  
 private string GetStringFromArray<T>(IEnumerable<T> array, string sep = ", ") {  
 return sortedArray.text = string.Join(sep, array);  
 }  
  
 private void FindInfoForGraph() {  
 var writePath = Path.Combine(Application.dataPath, FILE\_NAME);  
 for (var i = 1; i < 1000000; i++) {  
 var arr = RandomizeArray(i, 0, 1);  
 arr.ToArray().CocktailSort(out var numOfOperations);  
 try {  
 using (var sw = new StreamWriter(writePath, true, System.Text.Encoding.Default)) {  
 sw.WriteLine($"{i} {numOfOperations}");  
 }  
 }  
 catch (Exception e) {  
 Console.WriteLine(e.Message);  
 }  
 }  
 }  
}

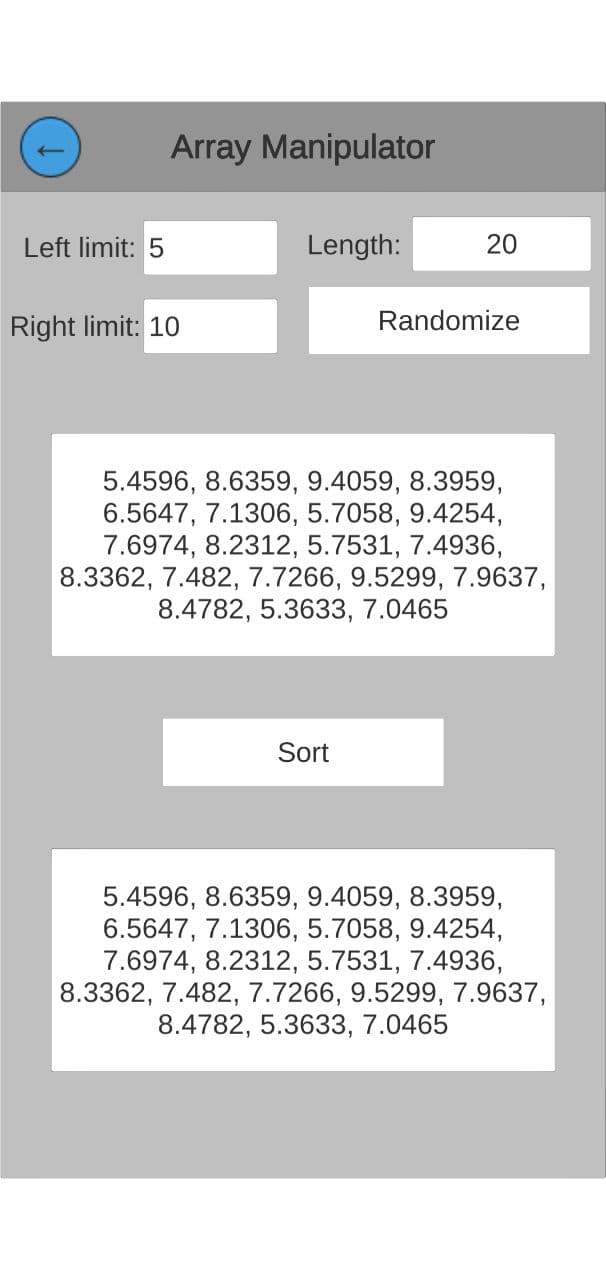
ArrayExtension.cs

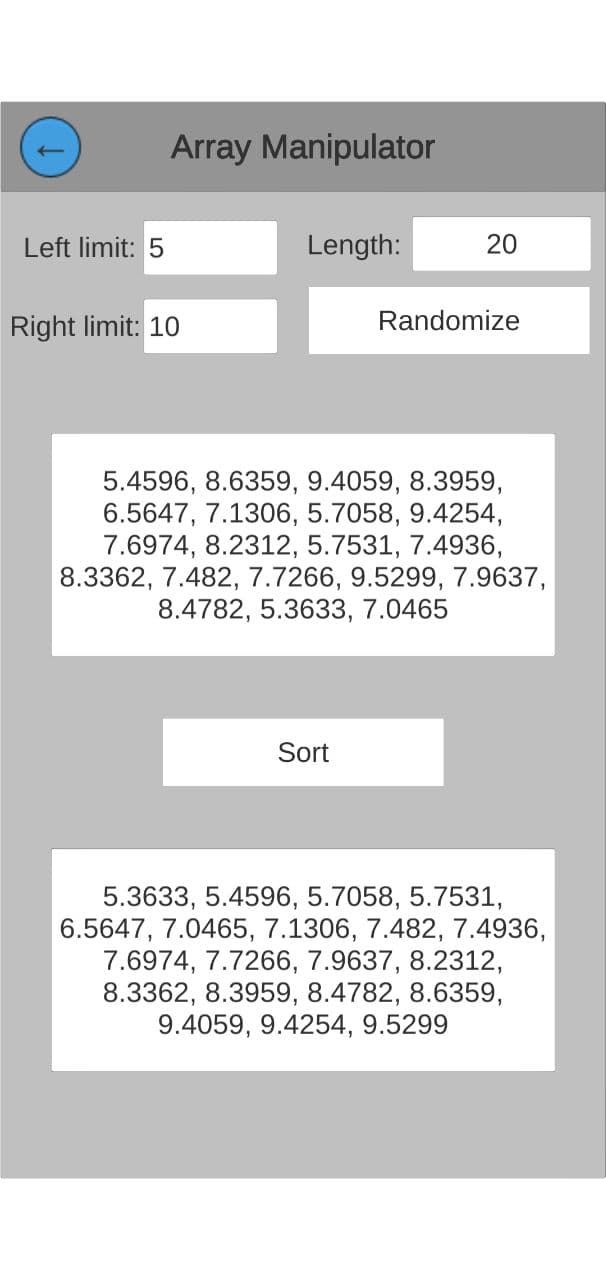
using System;  
  
namespace Classes {  
 public static class ArrayExtension {  
 public static void CocktailSort<T>(this T[] arr, out long numOfOperations) where T : IComparable<T> {  
 numOfOperations = 0;  
 for (var i = 0; i < arr.Length / 2; i++) {  
 var swapFlag = false;  
 for (var j = i; j < arr.Length - i - 1; j++) {  
 numOfOperations += 1;  
 if (arr[j].CompareTo(arr[j + 1]) > 0) {  
 Swap(ref arr[j], ref arr[j + 1]);  
 swapFlag = true;  
 }  
 }  
  
 for (var j = arr.Length - 2 - i; j > i; j--) {  
 numOfOperations += 1;  
 if (arr[j].CompareTo(arr[j - 1]) < 0) {  
 Swap(ref arr[j - 1], ref arr[j]);  
 swapFlag = true;  
 }  
 }  
  
 if (!swapFlag) {  
 break;  
 }  
 }  
 }  
  
 private static void Swap<T>(ref T e1, ref T e2) {  
 var temp = e1;  
 e1 = e2;  
 e2 = temp;  
 }  
 }  
}

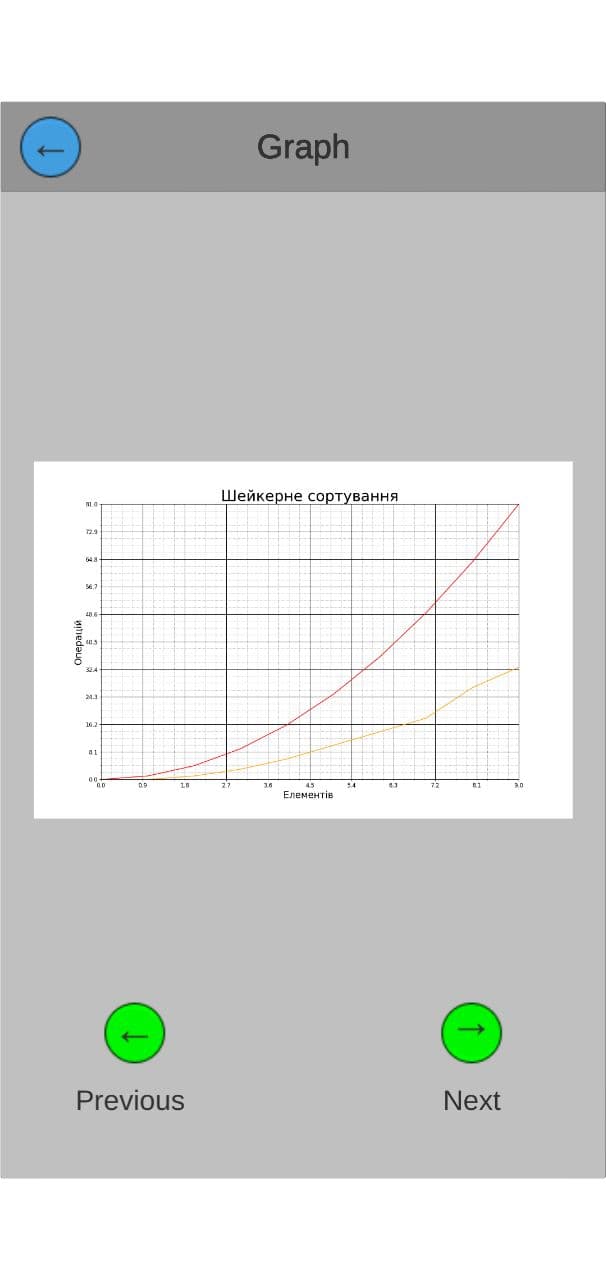
**Скріншоти виконання:**

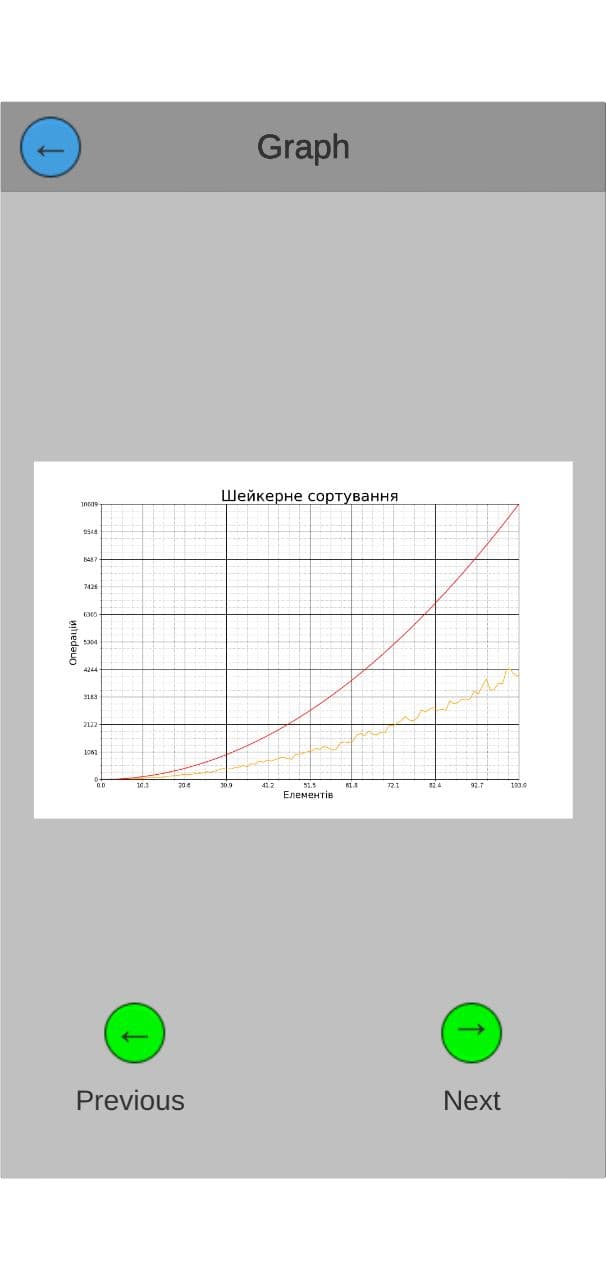


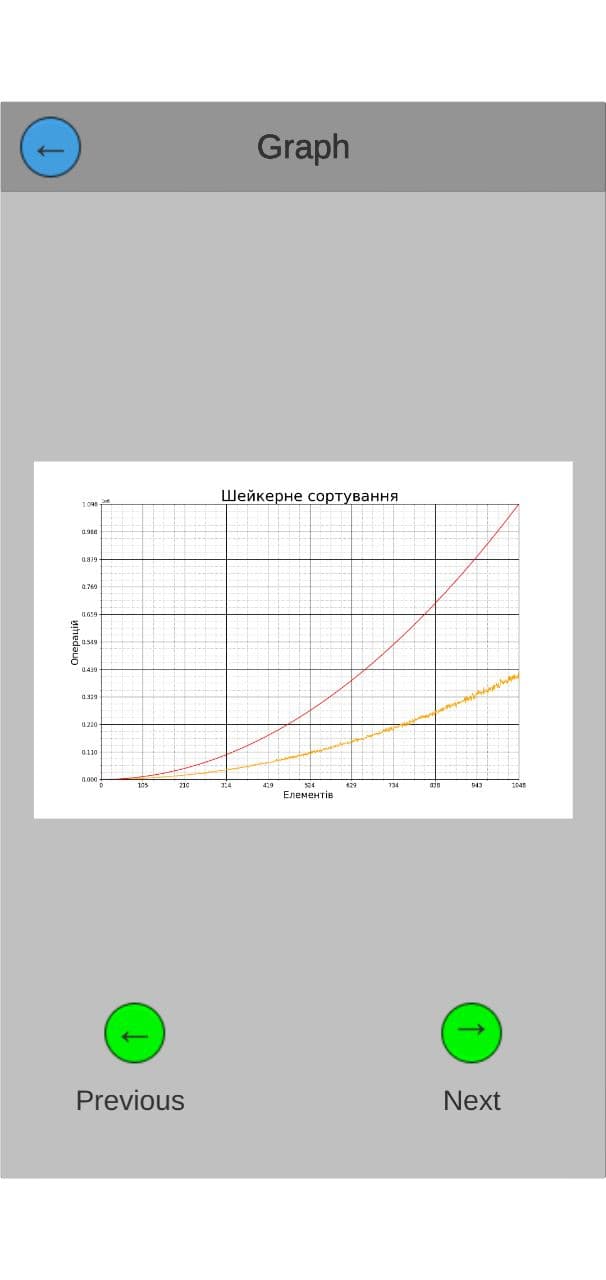


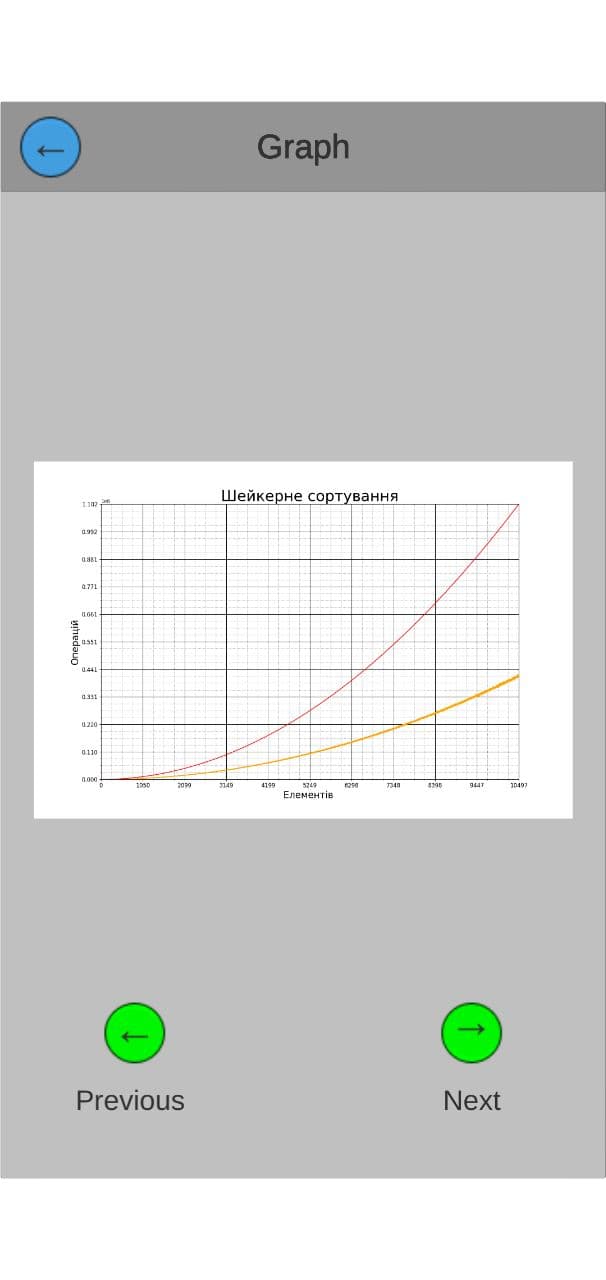












**Висновок:** У ході виконання лабораторної роботи я закріпив знання з базових понять алгоритмів, вивчив основні алгоритми сортування, покращив навички будування графіків за допомогою модуля matplotlib. Отримані результати виконання програми є вірними.