МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій
Кафедра програмного забезпечення

ЗВІТ
до лабораторної роботи №4
на тему: «Метод швидкого сортування»
з дисципліни: «Алгоритми і структури даних»

Лектор:
доц. кафедри ПЗ
Коротєєва Т. О.

Виконав:
ст. гр. ПЗ-22
Чаус О. М.

Прийняв:
асист. кафедри ПЗ
Франко А. В.

« ____ » _____ 2022 р.

∑= _____ .....:

Львів-2022

Тема роботи: Метод швидкого сортування.

Мета роботи: Вивчити алгоритм швидкого сортування. Здійснити програмну реалізацію алгоритму швидкого сортування. Дослідити 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·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.

Покрокове зображення алгоритму

АЛГОРИТМ QS. Сортування масиву алгоритмом Швидкого сортування в порядку зростання. **array** – вхідний масив, **i**, **j** – індекси елементів масиву, **low, high** – межі підмасиву, **pivot** – опорний елемент

QS1. Виклик ф-ції з межами **low** = 0, **high** = розмір **array** - 1

QS2. Якщо **low** > **high**, перехід на **QS9**. Ініціалізація **i** = **low**, **j** =**high** – 1.

QS3. Ініціалізація **pivot** = випадковий елемент. Swap(array[pivot], array[high])

QS4. Якщо i > j, перехід на **QS8**, інакше на **QS5**.

QS5. Якщо **array[i]** < **pivot**, інкрементація **i**++, перехід на **QS4**. Інакше, перехід на **QS6**.

QS6. Якщо **array[j]** > **pivot**, декрементація **j**--, перехід на **QS4**. Інакше, перехід на **QS7**.

QS7. Swap(**array[i]**, **array[j]**). Інкрементація **i**++, декрементація **j**--. Перехід на **QS4**.

QS8. Swap(**array[i]**, array[high]). Перехід на **QS2** з межами **low** = **low**, **high** = **i** – 1. Перехід на **QS2** з межами **low** = **i** + 1, **high** = **high**.

QS9. Вихід з функції

Індивідуальне завдання

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 вибором;
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 16:

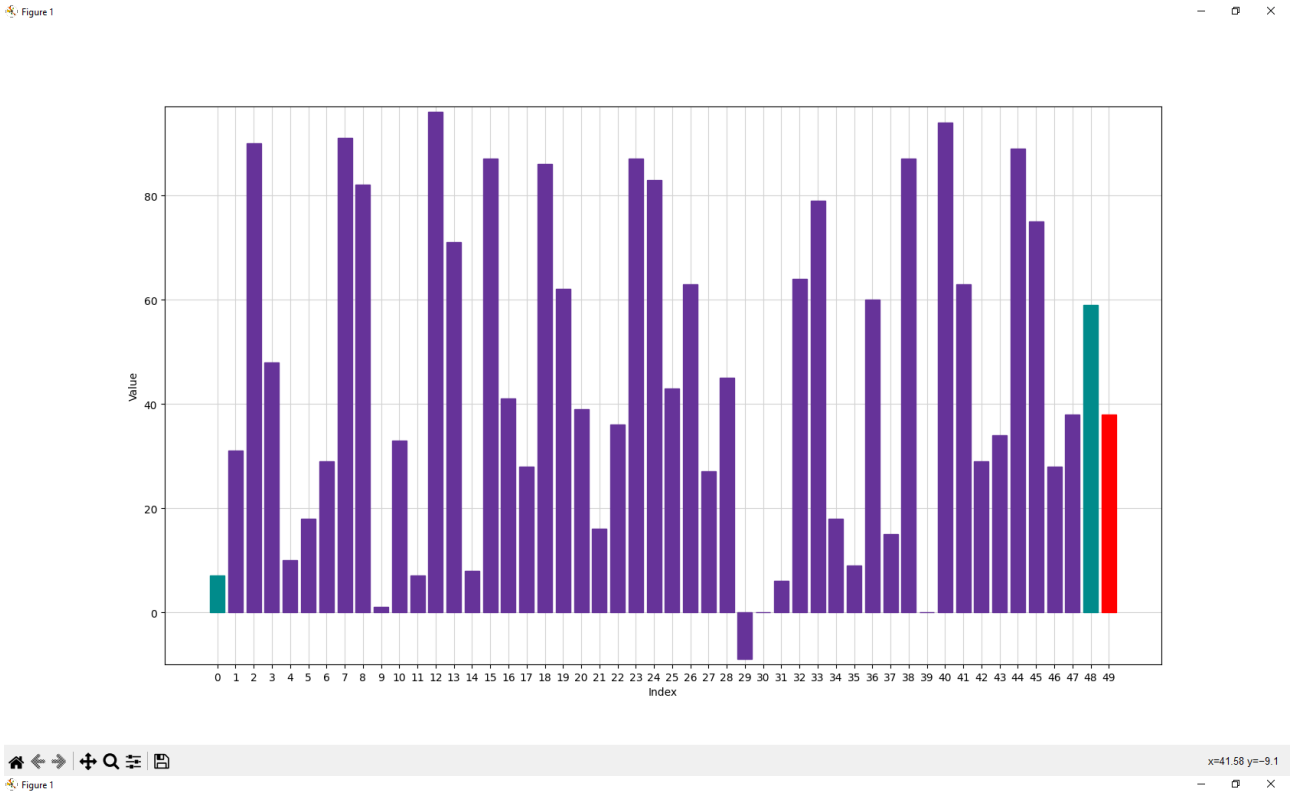
Задано одномірний масив дійсних чисел. Від’ємні елементи масиву домножити на мінімальний елемент. Отриманий масив посортувати в порядку спадання.

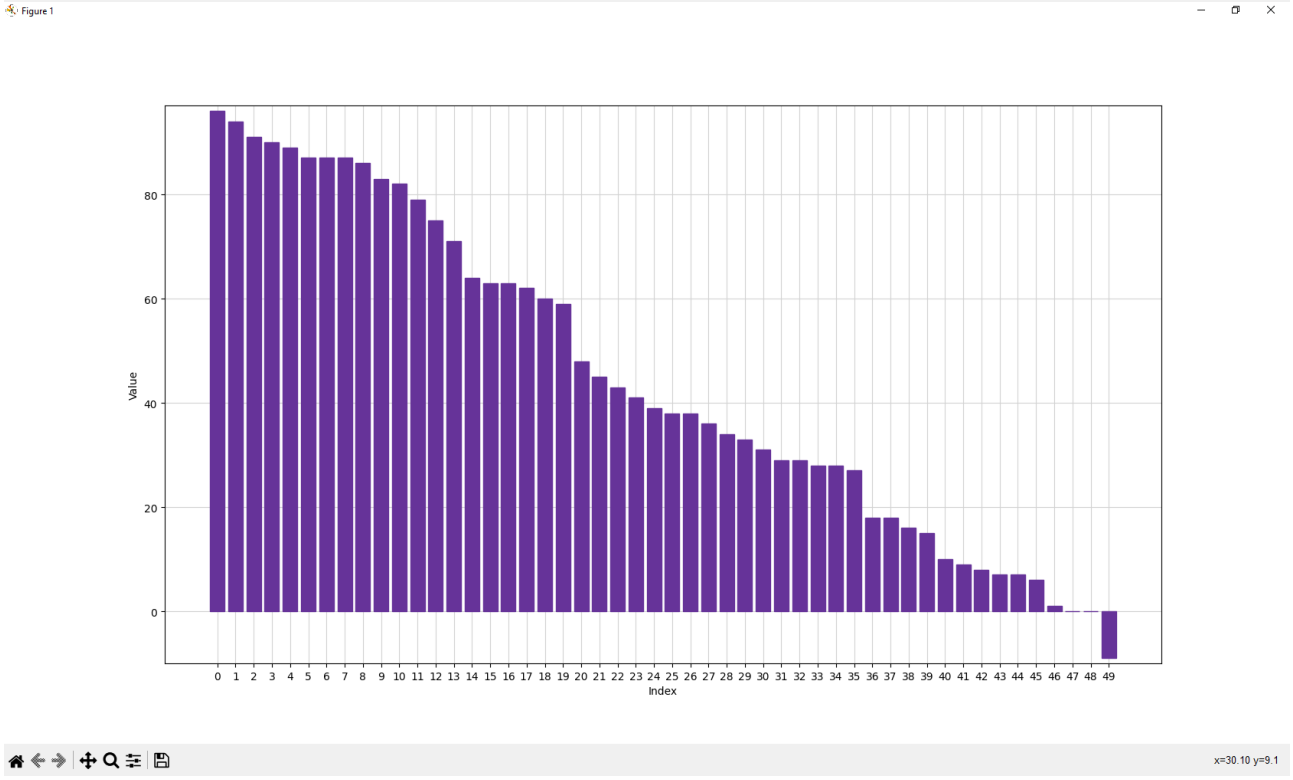
Код програми

```
import random
import numpy as np

def quick_sort(array):
    min_idx = np.argmin(array)
    for idx in range(array.size):
        if array[idx] < 0 and idx != min_idx:
            array[idx] *= array[min_idx]
    def qs(arr, low, high):
        if(low < high):
            from_left = low
            from_right = high - 1
            pivot_index = random.randint(low, high)
            pivot = array[pivot_index]
            arr[pivot_index], arr[high] = arr[high], arr[pivot_index]
            while(from_left <= from_right):
                if arr[from_left] > pivot:
                    from_left += 1
                elif arr[from_right] < pivot:
                    from_right -= 1
                else:
                    arr[from_left], arr[from_right] = arr[from_right],
arr[from_left]
                    from_left += 1
                    from_right -= 1
            arr[high], arr[from_left] = arr[from_left], arr[high]
            qs(arr, low, from_left - 1)
            qs(arr, from_left + 1, high)
    qs(array, 0, array.size - 1)
```

Скріншоти програми





Висновок: під час виконання лабораторної роботи вивчив алгоритм швидкого сортування та здійснив програмну реалізацію. Швидкодія алгоритму дорівнює $O(n \log n)$