**МІНІСТЕРСТВО ОСВІТИ І НАУКИ  УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **ІКНІ**

Кафедра **ПЗ**

**ЗВІТ**

До лабораторної роботи № 4

**З дисципліни:** *“Алгоритми та структури даних”*

**На тему:** *“ Метод швидкого сортування.”*

**Лектор:**

доц. каф. ПЗ

Коротєєва Т.О.

**Виконав:**

ст. гр. ПЗ – 22

Солтисюк Д.А.

**Прийняв:**

асист. каф. ПЗ

Франко А.В.

« \_\_\_\_ » \_\_\_\_\_\_\_\_  2022 р.

 ∑= \_\_\_\_\_                               .

Львів – 2022

**Тема роботи:** Метод сортування вибором.

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

**TЕОРЕТИЧНІ ВІДОМОСТІ**

 Швидке сортування (англійською «Quick Sort») — алгоритм сортування, який не потребує додаткової пам’яті і виконує *O*(*n*∙log(*n*)) операцій порівнянь.

Ідея алгоритму полягає в перестановці 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 двонапрямленому списку.

Час роботи алгоритму швидкого сортування залежить від збалансованості, що характеризує розбиття. Збалансованість, у свою чергу залежить від того, який елемент обрано як базовий. Якщо розбиття збалансоване, то асимптотично алгоритм працює так само швидко як і алгоритм сортування злиттям. У найгіршому випадку асимптотична поведінка алгоритму настільки ж погана, як і в алгоритму сортування включенням.

Отже, для реалізації алгоритму швидкого сортування вводимо два вказівники і та j , та проводимо їх ініціалізацію i = 1, j = n. , де n кількість елементів  вхідного масиву. Довільним чином вибирається за базовий будь-який елемент з масиву (перший, середній або останній).

Алгоритм працює за принципом "спалювання свічки з обох кінців". Рекурсія завершується, коли всі підмножини будуть складатися з одного елементу, або весь масив буде впорядкований.

***Покроковий опис роботи алгоритму сортування вибором.***

**Алгоритм B.**

Задано масив елементів Rn, n – розмір масиву*.* Даний алгоритм реорганізує масив у зростаючому порядку, тобто для його елементів буде мати місце співвідношення: Ri<Ri+1, для всіх i=1..n-1

1. Якщо n < 2, то перейти до кроку 7.
2. Визначити базовий елемент P.
3. Присвоїти змінні i=1, j=n.
4. Цикл. Повторювати крок 5  при ij.
5. Якщо Ri<P, то i=i+1, якщо Rj>P, то j=j-1, інакше переставити місцями елементи Ri та Rj, та присвоїти значення змінним i=i+1, j=j-1.
6. Рекурсивне сортування частин. Повторити заданий алгоритм для лівої частини R1,…,Ri-1 та правої частини Ri,…, Rn.
7. Вихід.

**ЗАВДАННЯ**

Задано перелік студентів і їх середній бал. Упорядкувати за алфавітом список тих студентів, середній бал яких вищий за 4.

**ХІД РОБОТИ**

**Код функції сортування:**

**def** swap(A, i, j):

A[i], A[j] = A[j], A[i]

**def** quick\_sort(arr, comparator):

**return** \_quick\_sort(arr, 0, len(arr) - 1, comparator)

**def** \_quick\_sort(arr, p, q, comparator):

**if** p >= q:

**return**

piv = arr[q]

pivindx = p

**for** i **in** range(p, q):

**if** comparator(arr[i], piv):

swap(arr, i, pivindx)

pivindx += 1

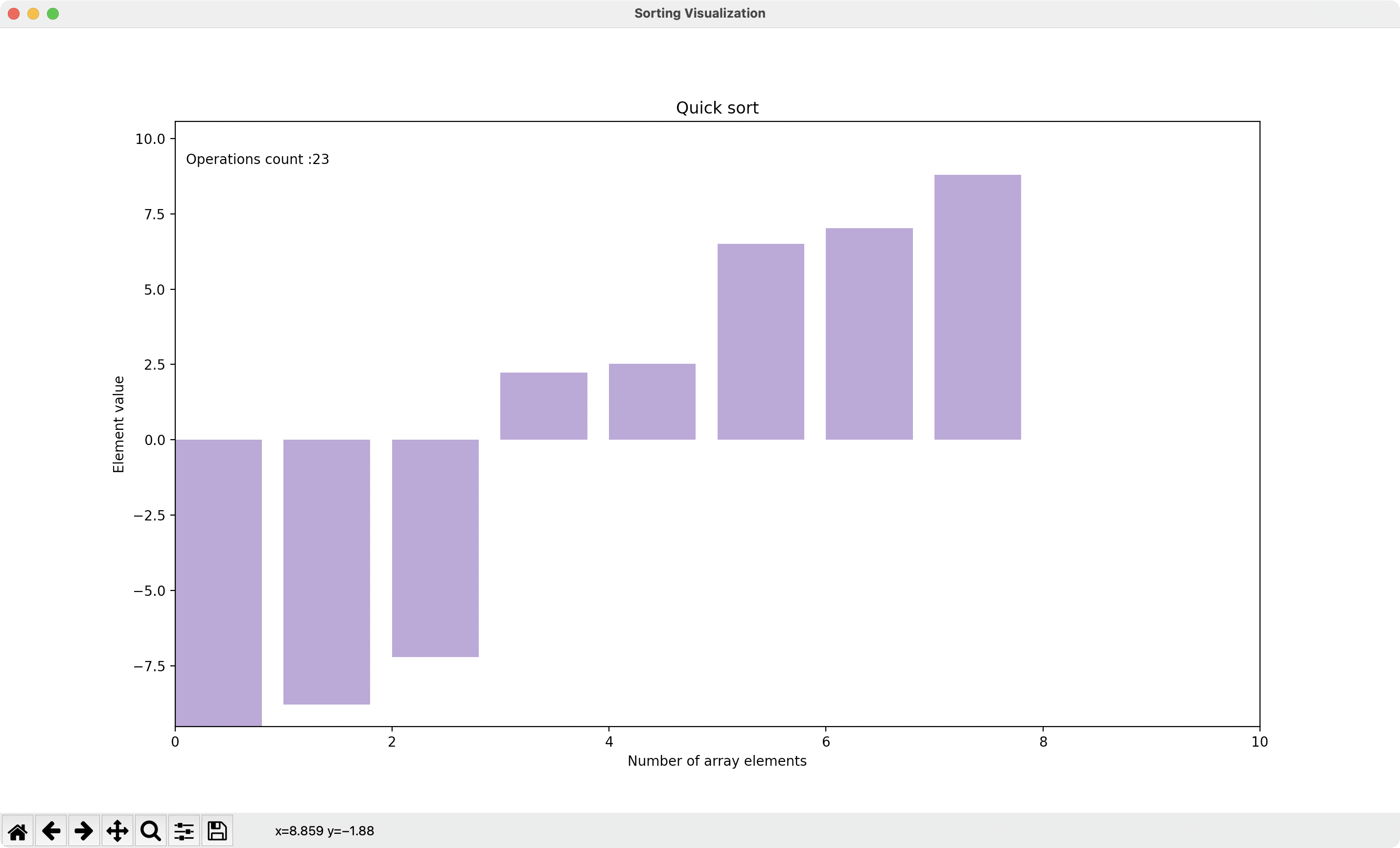
**yield** arr

swap(arr, q, pivindx)

**yield** arr

**yield from** \_quick\_sort(arr, p, pivindx - 1, comparator)

**yield from** \_quick\_sort(arr, pivindx + 1, q, comparator)

**РЕЗУЛЬТАТИ**

**ВИСНОВКИ**

На даній лабораторній роботі я розглянув реалізацію алгоритму швидкого сортування. В результаті виконання лабораторної роботи я навчився використовувати даний алгоритм для сортування масивів. Варто зазначити, що даний алгоритми є ефективним, але не стабільним. Його складність складає O(nlog(n)) в найкращому та середньому випадку. Проте, при найгіршому випадку складність алгоритму складає O(n2). Даний алгоритм застосовується в більшості реальних системах, оскільки він не вимагає додаткової пам’яті для реалізації і працює досить швидко в більшості випадках.