

**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
Ясногородський Н.В.

Прийняв:

асист. каф. ПЗ
Франко А.В.

« ____ » _____ 2022 р.

Σ = ____ .

Тема роботи: Метод сортування вибором.

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

ТЕОРЕТИЧНІ ВІДОМОСТІ

Швидке сортування (англійською «Quick Sort») — алгоритм сортування, який не потребує додаткової пам'яті і виконує $O(n \cdot \log(n))$ 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 i 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 елемента, або весь масив буде впорядкований.

Покроковий опис роботи алгоритму сортування вибором.

Алгоритм В.

Задано масив елементів R_n , n – розмір масиву. Даний алгоритм реорганізує масив у зростаючому порядку, тобто для його елементів буде мати місце співвідношення: $R_i < R_{i+1}$, 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 $i < j$.
5. Якщо $R_i < P$, то $i=i+1$, якщо $R_j > P$, то $j=j-1$, інакше переставити місцями елементи R_i та R_j , та присвоїти значення змінним $i=i+1$, $j=j-1$.
6. Рекурсивне сортування частин. Повторити заданий алгоритм для лівої частини R_1, \dots, R_{i-1} та правої частини R_i, \dots, R_n .
7. Вихід.

ЗАВДАННЯ

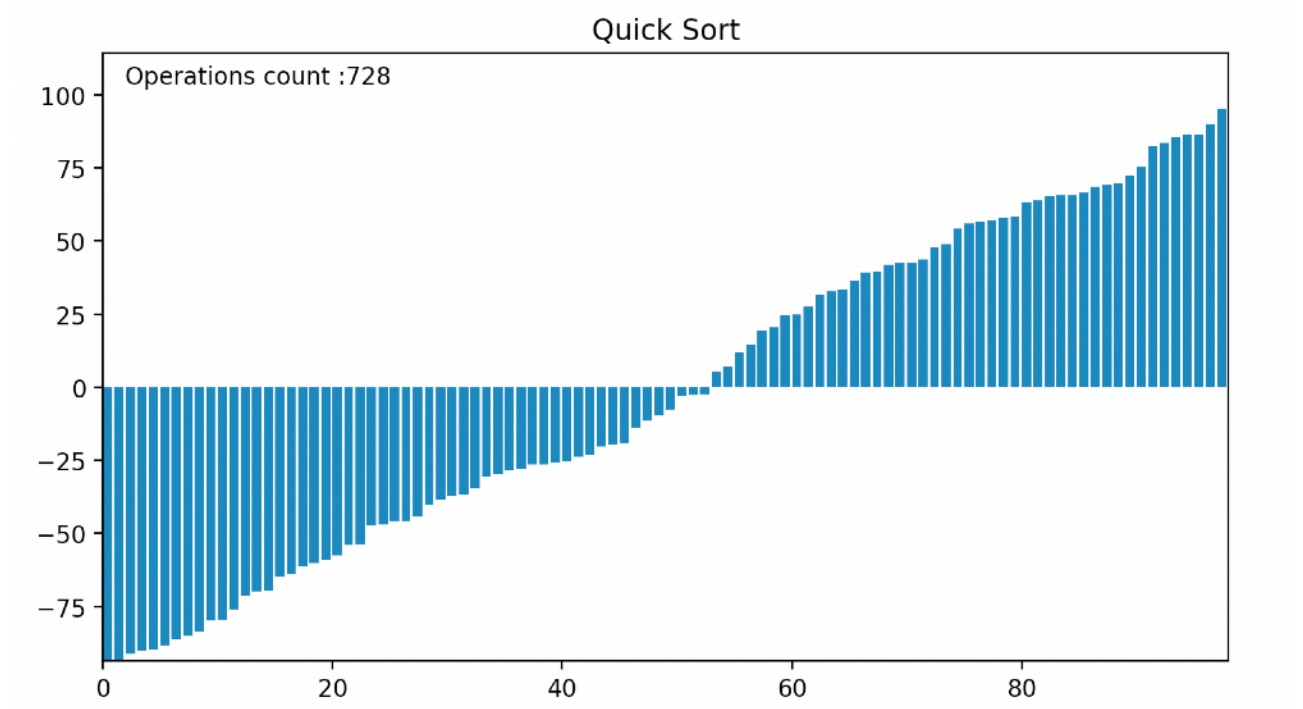
Задано перелік студентів і їх середній бал. Упорядкувати за алфавітом список тих студентів, середній бал яких вищий за 4.

ХІД РОБОТИ

Код функції сортування:

```
def quick_sort(arr, comparator):  
    return _quick_sort(arr, 0, len(arr) - 1, comparator)  
  
def swap(A, i, j):  
    A[i], A[j] = A[j], A[i]  
  
def _quick_sort(arr, a, b, comparator):  
    if a >= b:  
        return  
    piv = arr[b]  
    piv_idx = a  
    for i in range(a, b):  
        if comparator(arr[i], piv):  
            swap(arr, i, piv_idx)  
            piv_idx += 1  
    yield arr  
    swap(arr, b, piv_idx)  
    yield arr  
  
    yield from _quick_sort(arr, a, piv_idx - 1, comparator)  
    yield from _quick_sort(arr, piv_idx + 1, b, comparator)
```

РЕЗУЛЬТАТИ



ВИСНОВКИ

Я розглянув реалізацію алгоритму швидкого сортування. Варто зазначити, що даний алгоритм є ефективним, але не стабільним. Його складність складає $O(n \log(n))$ в найкращому та середньому випадку. Проте, при найгіршому випадку складність алгоритму складає $O(n^2)$. Даний алгоритм застосовується в більшості реальних системах, оскільки він не вимагає додаткової пам'яті для реалізації і працює досить швидко в більшості випадках.