МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій
Кафедра програмного забезпечення

ЗВІТ

до лабораторної роботи №6

на тему: «Метод сортування підрахунком»

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 сортування підрахунком.

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

Теоретичні відомості

Сортування підрахунком (англійською «Counting Sort») — алгоритм впорядкування, що застосовується при малій кількості різних елементів (ключів) у масиві даних. Час його роботи лінійно залежить як від загальної кількості елементів у масиві так і від кількості різних елементів.

Ідея алгоритму полягає в наступному: спочатку підрахувати скільки разів кожен елемент (ключ) зустрічається в вихідному масиві. Спираючись на ці дані можна одразу вирахувати на якому місці має стояти кожен елемент, а потім за один прохід поставити всі елементи на свої місця.

В алгоритмі присутні тільки прості цикли довжини N (довжина масиву), та один цикл довжини K (величина діапазону). Отже, обчислювальна складність роботи алгоритму становить O(N + K).

В алгоритмі використовується додатковий масив. Тому алгоритм потребує E(K) додаткової пам'яті.

В такій реалізації алгоритм є стабільним. Саме ця його властивість дозволяє використовувати його як частину інших алгоритмів сортування (наприклад, сортування за розрядами).

Використання даного алгоритму є доцільним тільки у випадку малих K.

Покрокове зображення алгоритму

АЛГОРИТМ CS. Сортування масиву алгоритмом підрахунку в порядку зростання.
array – вхідний масив, **i**, – індекс елементів масиву, **out** – вихідний масив, **count** – масив, що містить кількість кожного елемента в **array**.

CS1. Ініціалізація out = [0 * розмір **array**]. Інціалізація count = [0 * (максимальне значення **array** – мінімальне значення **array** + 1)]. Ініціалізація **i** = 0

CS2. Якщо **i** < розмір **array**, перехід на **CS3**. Інакше перехід на CS4

CS3. Інкрементація count[**array[i]** – мінімальне значення array] += 1. **i** += 1, перехід на **CS2**.

CS4. Ініціалізація **i** = 1.

CS5. Якщо **i** < розмір **count**, перехід на **CS6**, інакше перехід на **CS7**.

CS6. **count**[**i**] += **count**[**i** – 1]. **i** += 1. Перехід на **CS5**.

CS7. Ініціалізація **i** = розмір **array** – 1.

CS8. Якщо **i** > 0, перехід на **CS9**, інакше перехід на **CS10**.

CS9. **out**[**count**[**array**[**i**] – мінімальне значення **array**] - 1] = **array**[**i**]. Декрементація **count**[**array**[**i**] – мінімальне значення **array**] -= 1. Інкрементація **i** += 1. Перехід на **CS8**.

CS10. Вихід з алгоритму.

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.

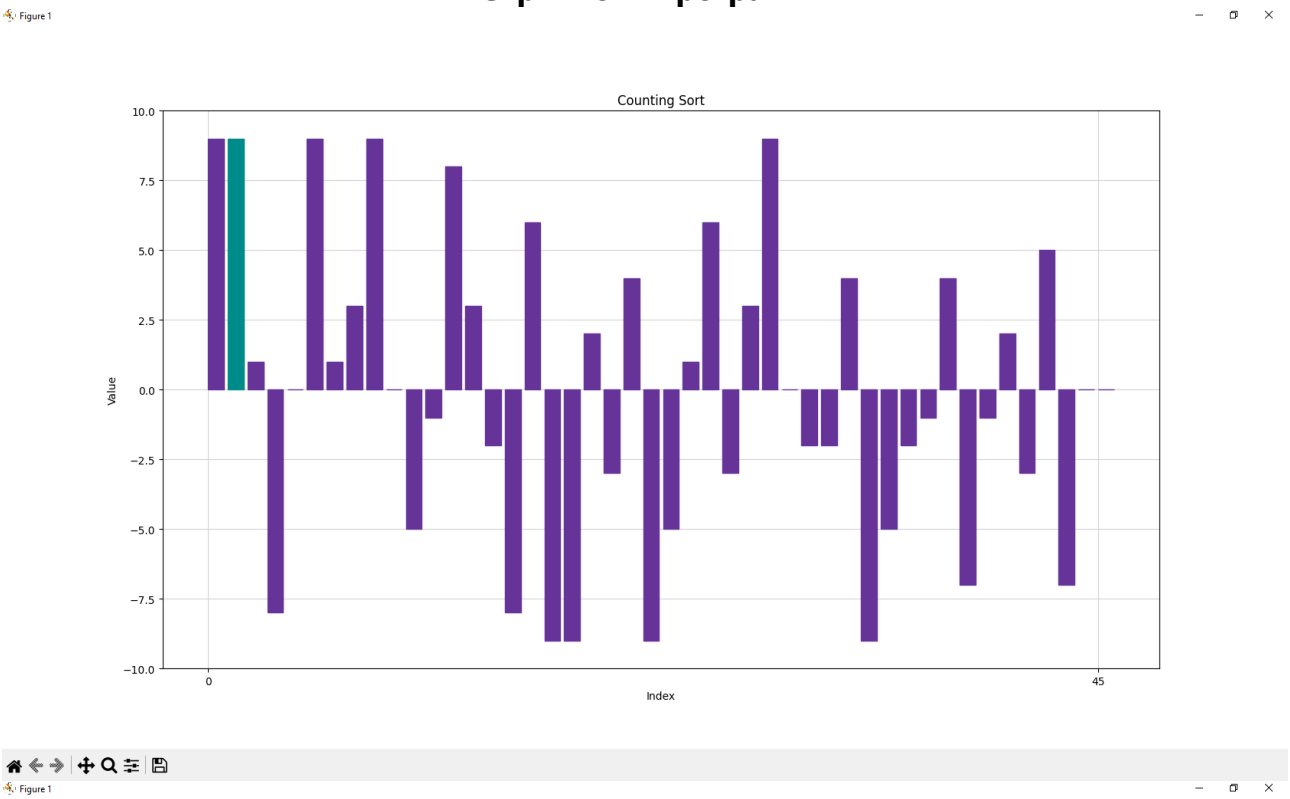
Варіант 1:

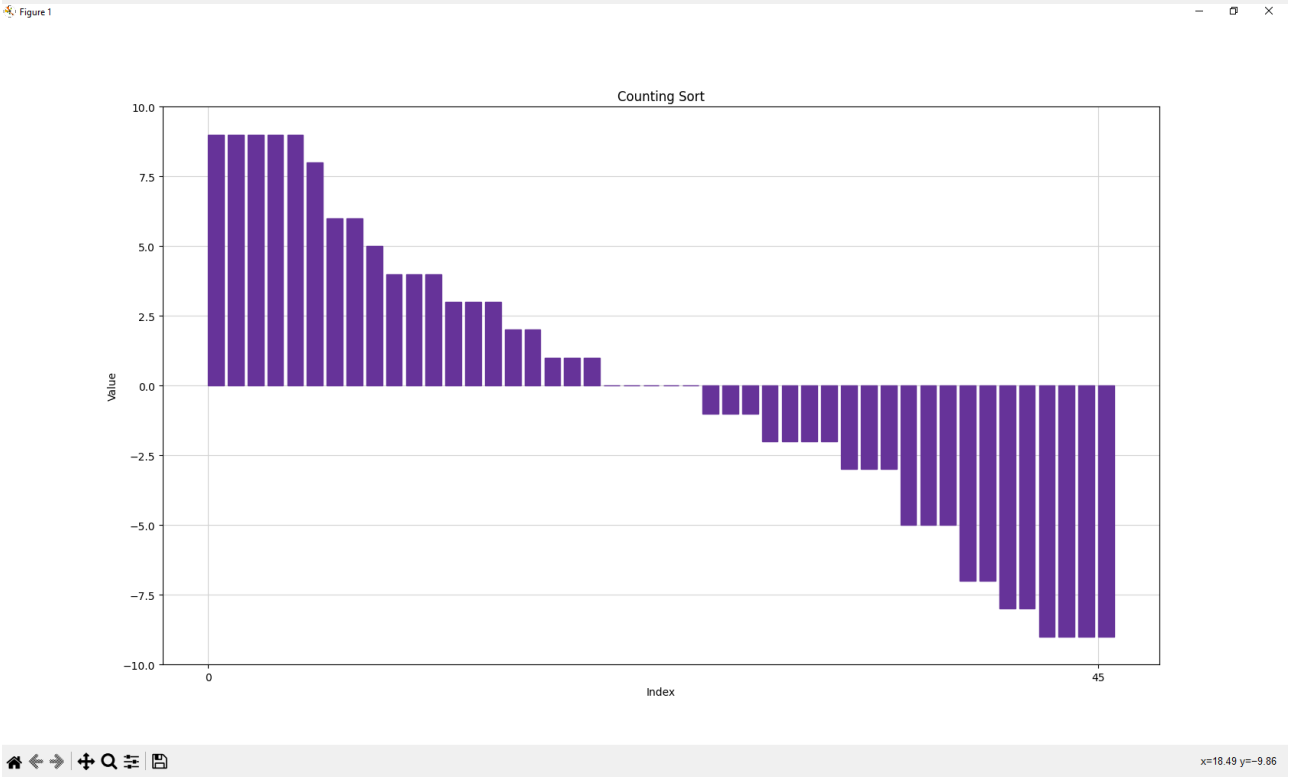
Задано одномірний масив дійсних чисел. Виключити з нього мінімальний та максимальний елементи. Отриманий масив посортувати в порядку спадання.

Код програми

```
import numpy as np

def counting_sort(array):
    array = np.delete(array, np.where(array == np.min(array)))
    array = np.delete(array, np.where(array == np.max(array)))
    min = np.min(array)
    max = np.max(array)
    output = np.zeros(array.size, dtype=int)
    count = np.zeros(max - min + 1, dtype=int)

    for elem in array:
        count_idx = elem - min
        count[count_idx] += 1

    for i in range(1, count.size):
        count[i] += count[i - 1]

    for i in range(array.size - 1, -1, -1):
        count_idx = array[i] - min
        output[output.size - count[count_idx]] = array[i]
        count[count_idx] -= 1
```

Скріншоти програми





Висновок:

під час виконання лабораторної роботи вивчив алгоритм сортування підрахунком та здійснив програмну реалізацію.

Швидкодія алгоритму дорівнює $O(n + k)$