МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського

«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп’ютерних систем та інфокомунікацій

Кафедра комп’ютерних систем, мереж і кібербезпеки (503)

Лабораторна робота № *2*

|  |  |
| --- | --- |
|  | *Дослідження покращених алгоритмів сортування* |
|  | (назва лабораторної роботи) |
| з дисципліни | *Моделі та структури даних* |
|  | (шифр)  ХАІ**.**503**.**525a**.**03О**.**123-Комп'ютерна інженерія**,** ПЗ №9629619 |

|  |  |  |
| --- | --- | --- |
| Виконав студент гр. | 525а | *Литвиненко А.В.* |
| 26.10.22 | (№ групи) | (П.І.Б.) |
| (підпис, дата) |  |  |
| Перевірив | канд. техн. наук, доцент | |
|  |  | *А. В. Шостак* |
| (підпис, дата) |  | (П.І.Б.) |

Харків – 2022

**Тема роботи:** Дослідження покращених алгоритмів сортування

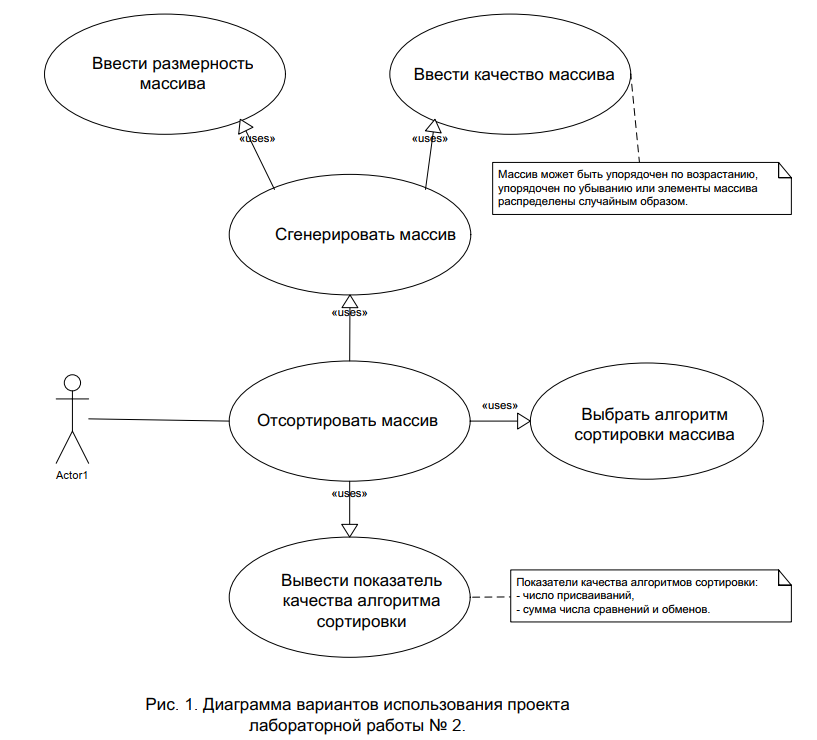
**Мета роботи**: Розробити проект для дослідження алгоритмів сортування

**Варіант 5**

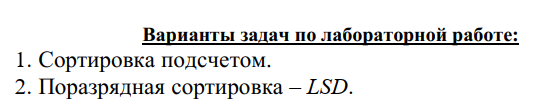
**Задача 1**

**Частина 1**. Постановка завдання

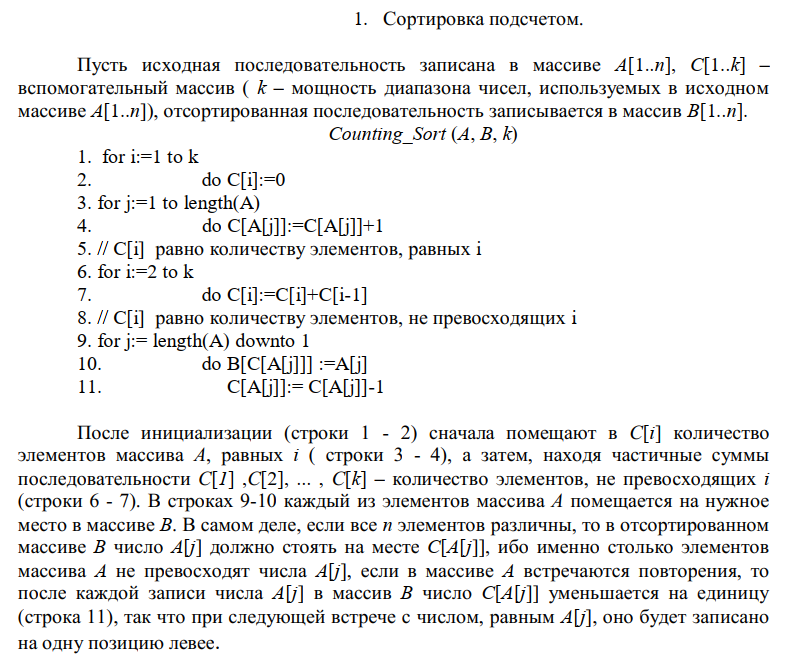
**Умова:** Розробити проект для дослідження алгоритмів сортування відповідно до варіантом (діаграма варіантів використання проекту представлена на рис. 1)

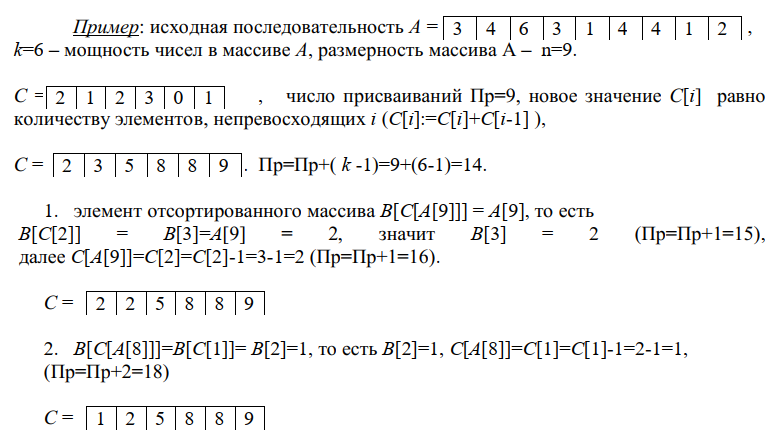


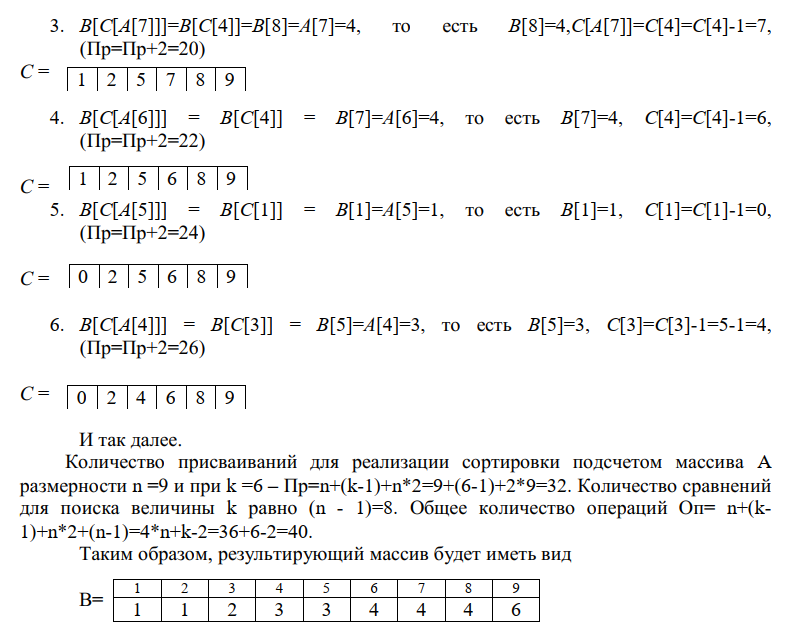
**Умова з додатка:**



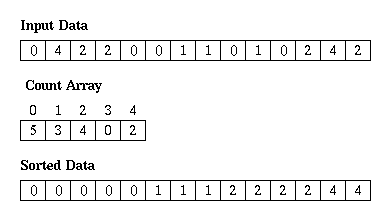




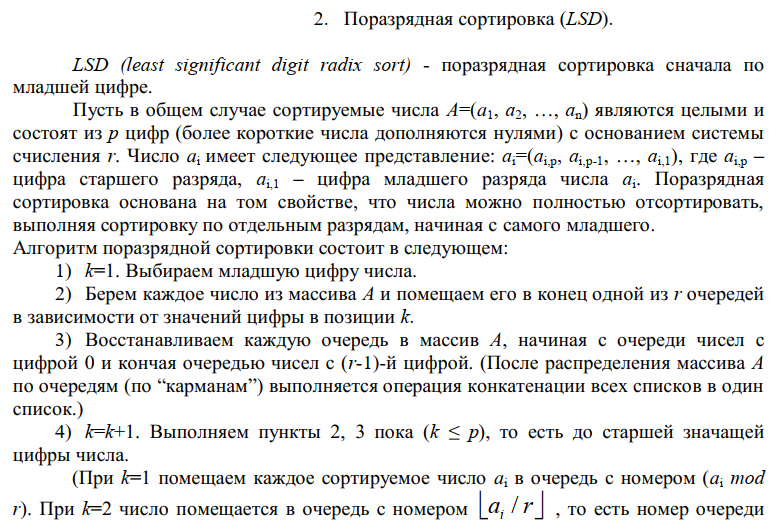


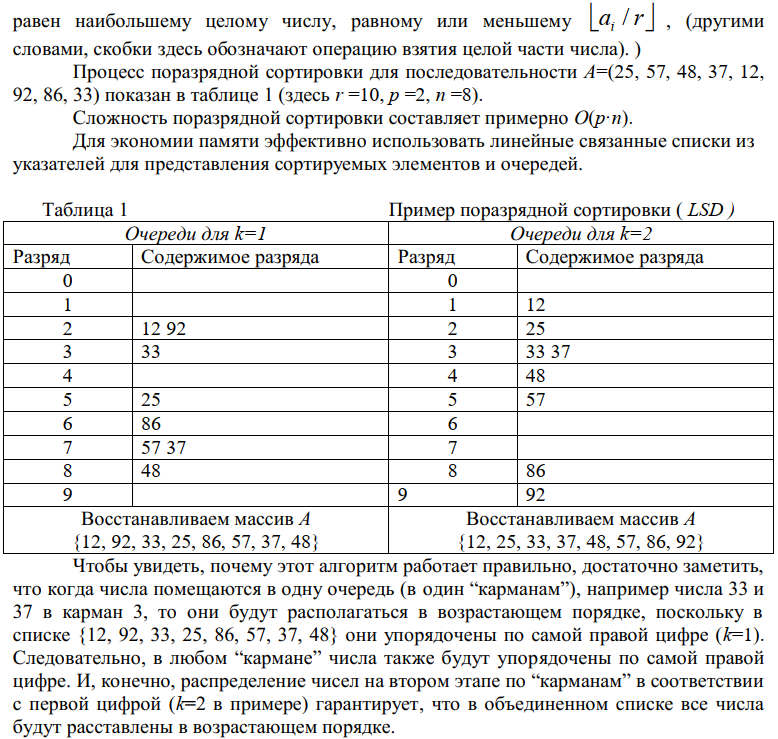


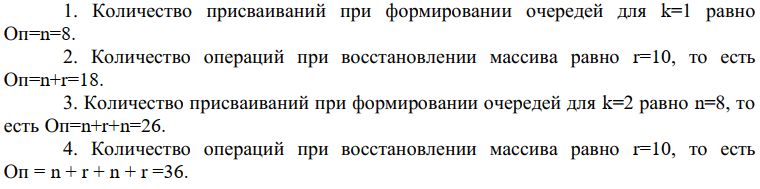
Приклад покрокового виконання:



Спочатку підраховується кожний елемент масиву і заноситься в Count array, після чого виконується послідовне вставлення елементів.









LSD-сортування виконується по бітово починаючи з найбільшого розряду, тобто: якщо перший біт нуль, а він ТОЧНО буде менший аніж число з найстаршим бітом 1 (не беремо до уваги типи даних signed, де найстарший біт = мінусу), тоді виконується обмін чисел. Таким чином пробігається по усіх розрядах.

**Частина 2**. Код програми

Main.py

# VARIANT: 5

# Сортування підрахунком

# Порозрядне сортування - LSD

# [0, 500] 1,2

from random import randint

from time import time

def genArray(array: list,

        start: int = 0, end: int = 10, step:int = 1,

        startNum:int = 0, endNum:int = 500) -> list:

    array = []

    for i in range(start, end, step):

        array.append(randint(startNum, endNum))

    return array

def countingSort(arr) -> list:

    """

    Версія сортування вибором для звичайного використання

    """

    print("[INFO] Counting sort starts...")

    op\_count = 0

    # Визначення розмірності масиву і створення пустої копії

    size = len(arr)

    output = [0] \* size

    # Ініціалізувати пустий массив для підрахунків

    count = [0] \* 501

    print("[Count] ", count)

    # Додавання кількості кожного елементу

    for m in range(0, size):

        op\_count += 1

        count[arr[m]] += 1

    print("[Initialized count] ", count)

    # Встановлення комулятивної кількості

    for m in range(1, 10):

        op\_count += 1

        count[m] += count[m - 1]

    print("[Comulative count] ", count)

    # Встановлення елементів в вихідний масив після пошуку індексу до кожного елемента оригінального масива в підрахунках

    m = size - 1

    while m >= 0:

        op\_count += 1

        output[count[arr[m]] - 1] = arr[m]

        count[arr[m]] -= 1

        m -= 1

        print("[PROCESSING ARRAY] ", output, count)

    # Перезапис існуючого масиву на відсортований

    for m in range(0, size):

        arr[m] = output[m]

    print("[INFO] Counting sort finished! The result: ", arr, end="\n"\*2)

    print("[INFO] Operation count: ", op\_count)

    return arr

def countingSortForRadix(inputArray, placeValue):

    """

    Сортування вибором для сортування LSD

    """

    countArray = [0] \* 10

    inputSize = len(inputArray)

    for i in range(inputSize):

        placeElement = (inputArray[i] // placeValue) % 10

        countArray[placeElement] += 1

    for i in range(1, 10):

        countArray[i] += countArray[i-1]

    outputArray = [0] \* inputSize

    i = inputSize - 1

    while i >= 0:

        currentEl = inputArray[i]

        placeElement = (inputArray[i] // placeValue) % 10

        countArray[placeElement] -= 1

        newPosition = countArray[placeElement]

        outputArray[newPosition] = currentEl

        i -= 1

    return outputArray

def radixSort(inputArray: list) -> list:

    op\_count = 0

    # Знаходження максимального елементу у введенному масиві

    maxEl = max(inputArray)

    print("[INFO] Max element: ", maxEl)

    # Знайти число серед цифр яке буде найбільшим

    D = 1

    while maxEl > 0:

        op\_count += 1

        maxEl /= 10

        D += 1

    print("[INFO] Max number: ", maxEl)

    # Ініціалізувати місце для значення

    placeVal = 1

    # Виконання сортування

    outputArray = inputArray

    print("D\tArray")

    while D > 0:

        op\_count += 1

        print(D, outputArray)

        outputArray = countingSortForRadix(outputArray, placeVal)

        placeVal \*= 10

        D -= 1

    print("[INFO] The result is ", outputArray)

    print("[INFO] Operation count: ", op\_count)

    return outputArray

def Menu() -> None:

    print("""

    - 0. Show menu

    - 1. Set  array

    - 2. Show array

    - 3. Gen  array

    - 4. Counting sort

    - 5. LSD sort

    - 6. Exit

    """)

def main():

    op = 0

    arr = []

    while True:

        if not arr and op in [4,5]:

            print("You forget to enter the array")

            op = 0

        match op:

            case 0:

                Menu()

            case 1:

                arr = list(map(int, input("Enter array by space: ").split(" ")))

            case 2:

                print(arr)

            case 3:

                start = 0

                end = int(input("How many elements?: "))

                step = 1

                startNum = int(input("Min number: "))

                endNum = int(input("Max number: "))

                arr = genArray(arr,start,end,step,startNum,endNum)

                print("\nResult: ", arr)

            case 4:

                start\_time = time()

                print("Counting sort")

                countingSort(arr)

                print("Time: ", time() - start\_time)

            case 5:

                start\_time = time()

                print("LSD sort")

                radixSort(arr)

                print("Time: ", time() - start\_time)

            case 6:

                print("Exiting...")

                break

            case \_:

                print("Invalid operation")

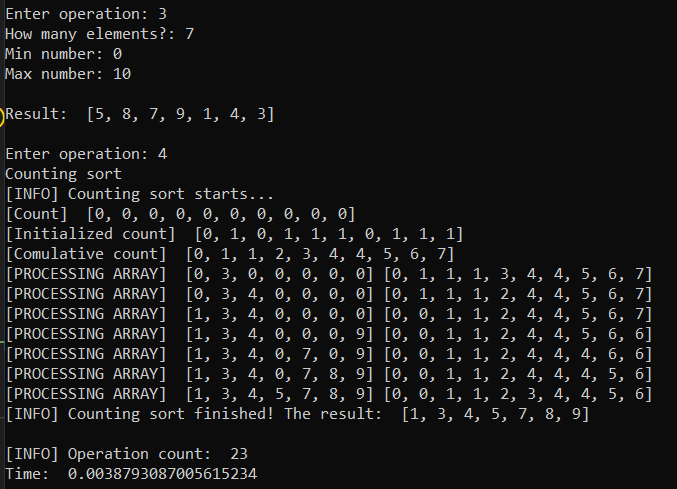
        op = int(input("\nEnter operation: "))

if \_\_name\_\_ == "\_\_main\_\_":

    main()

    print("Program finished")

Тест 1



Тест 2

nter operation: 3

How many elements?: 7

Min number: 0

Max number: 100

Result: [3, 38, 37, 68, 15, 100, 89]

Enter operation: 5

LSD sort

[INFO] Max element: 100

[INFO] Max number: 0.0

D Array

327 [3, 38, 37, 68, 15, 100, 89]

326 [100, 3, 15, 37, 38, 68, 89]

325 [100, 3, 15, 37, 38, 68, 89]

324 [3, 15, 37, 38, 68, 89, 100]

[INFO] The result is [3, 15, 37, 38, 68, 89, 100]

[INFO] Operation count: 653

Time: 0.25589561462402344

**Частина 3.** Порівняння алгоритмів

Сортування підрахунком

|  |  |  |  |
| --- | --- | --- | --- |
| Кількість елементів | Кількість порівнянь | Кількість обмінів | Сума |
| 5 | 0 | 19 | 19 |
| 10 | 0 | 29 | 29 |
| 15 | 0 | 39 | 39 |
| 20 | 0 | 49 | 49 |
| 25 | 0 | 59 | 59 |
| 30 | 0 | 69 | 69 |
| 35 | 0 | 79 | 79 |
| 40 | 0 | 89 | 89 |
| 45 | 0 | 99 | 99 |
| 50 | 0 | 109 | 109 |

LSD-сортування

|  |  |  |  |
| --- | --- | --- | --- |
| Кількість елементів | Кількість порівнянь | Кількість обмінів | Сума |
| 5 | 0 | 655 | 655 |
| 10 | 0 | 655 | 655 |
| 15 | 0 | 655 | 655 |
| 20 | 0 | 655 | 655 |
| 25 | 0 | 655 | 655 |
| 30 | 0 | 655 | 655 |
| 35 | 0 | 655 | 655 |
| 40 | 0 | 655 | 655 |
| 45 | 0 | 655 | 655 |
| 50 | 0 | 655 | 655 |

**Висновок.** Спираючись на розроблену програму і самостійне дослідження графіку.

* Обидві частини сортування не мають порівнянь
* Сортування LSD не залежить від якості масиву, розміру масиву
* Сортування підрахунком має кращі результати порівнюючи з LSD сортування, але при дуже великих масивах LSD буде мати перевагу.

**Використані джерела:**

* Лекції
* Пошукова система Google