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.503.525а.03О.123-Комп'ютерна інженерія, ПЗ №9629619

Виконав студент гр. 525а *Литвиненко А.В.*
(№ групи) (П.І.Б.)
26.10.22
(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)

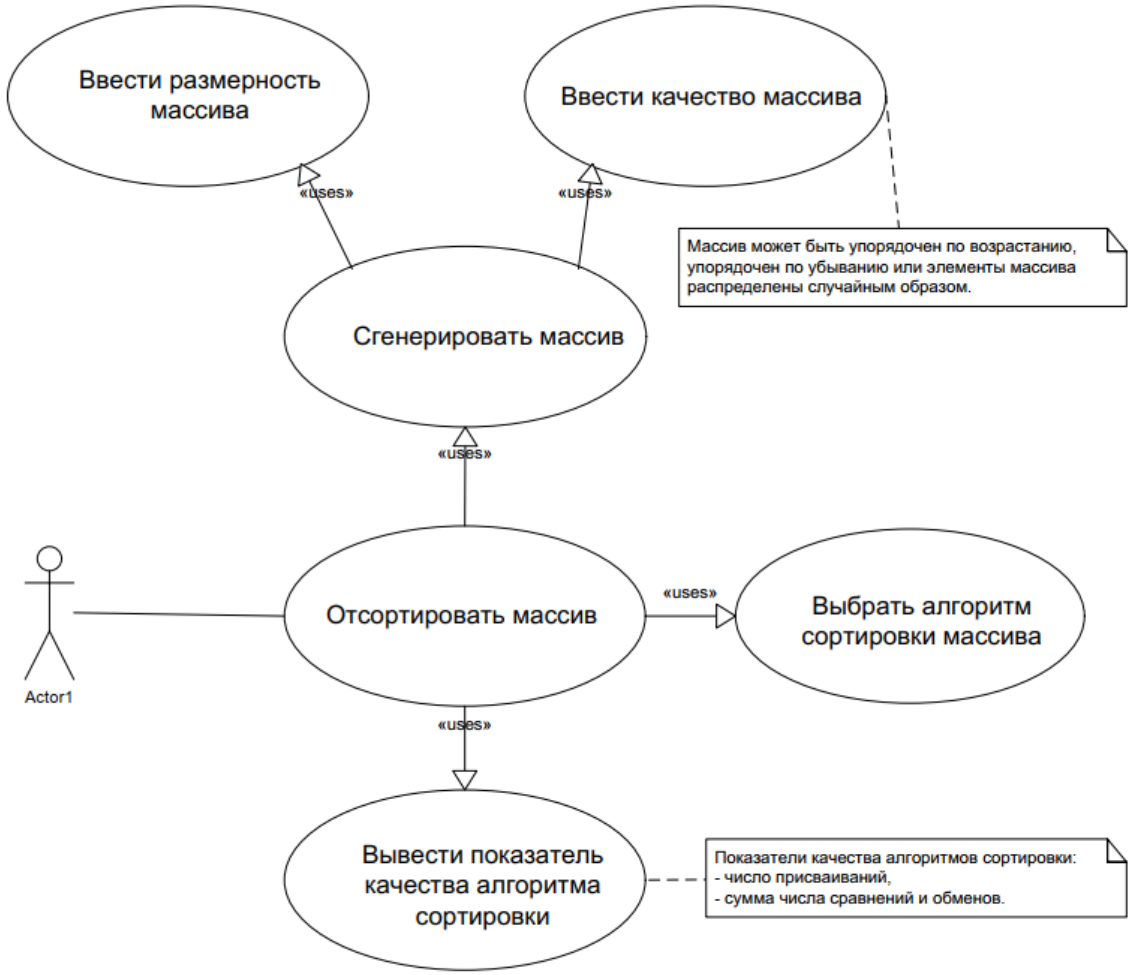


Рис. 1. Диаграмма вариантов использования проекта лабораторной работы № 2.

Умова з додатка:

Варианты задач по лабораторной работе:

1. Сортировка подсчетом.
2. Поразрядная сортировка – *LSD*.

5.	[0, 500]	1, 2

1. Сортировка подсчетом.

Пусть исходная последовательность записана в массиве $A[1..n]$, $C[1..k]$ – вспомогательный массив (k – мощность диапазона чисел, используемых в исходном массиве $A[1..n]$), отсортированная последовательность записывается в массив $B[1..n]$.

Counting_Sort (A, B, k)

```
1.  for i:=1 to k
2.              do C[i]:=0
3. for j:=1 to length(A)
4.              do C[A[j]]:=C[A[j]]+1
5. // C[i]  равно количеству элементов, равных i
6. for i:=2 to k
7.              do C[i]:=C[i]+C[i-1]
8. // C[i]  равно количеству элементов, не превосходящих i
9. for j:= length(A) downto 1
10.             do B[C[A[j]]] :=A[j]
11.                C[A[j]]:= C[A[j]]-1
```

После инициализации (строки 1 - 2) сначала помещают в $C[i]$ количество элементов массива A, равных i (строки 3 - 4), а затем, находя частичные суммы последовательности $C[1]$,$C[2]$, ... , $C[k]$ – количество элементов, не превосходящих i (строки 6 - 7). В строках 9-10 каждый из элементов массива A помещается на нужное место в массиве B. В самом деле, если все n элементов различны, то в отсортированном массиве B число $A[j]$ должно стоять на месте $C[A[j]]$, ибо именно столько элементов массива A не превосходят числа $A[j]$, если в массиве A встречаются повторения, то после каждой записи числа $A[j]$ в массив B число $C[A[j]]$ уменьшается на единицу (строка 11), так что при следующей встрече с числом, равным $A[j]$, оно будет записано на одну позицию левее.

Пример: исходная последовательность A = | 3 | 4 | 6 | 3 | 1 | 4 | 4 | 1 | 2 |, k=6 – мощность чисел в массиве A, размерность массива A – n=9.

C = | 2 | 1 | 2 | 3 | 0 | 1 | , число присваиваний Пр=9, новое значение $C[i]$ равно количеству элементов, непревосходящих i ($C[i]$:=$C[i]$+$C[i$-1]),

C = | 2 | 3 | 5 | 8 | 8 | 9 |. Пр=Пр+(k -1)=9+(6-1)=14.

1. элемент отсортированного массива $B[C[A[9]]] = A[9]$, то есть $B[C[2]] = B[3]=A[9] = 2$, значит $B[3] = 2$ (Пр=Пр+1=15), далее $C[A[9]]=C[2]=C[2]$-1=3-1=2 (Пр=Пр+1=16).

C = | 2 | 2 | 5 | 8 | 8 | 9 |

2. $B[C[A[8]]]=B[C[1]]= B[2]=1$, то есть $B[2]=1$, $C[A[8]]=C[1]=C[1]$-1=2-1=1, (Пр=Пр+2=18)

C = | 1 | 2 | 5 | 8 | 8 | 9 |

3. $B[C[A[7]]]=B[C[4]]=B[8]=A[7]=4$, то есть $B[8]=4, C[A[7]]=C[4]=C[4]-1=7$, (Пр=Пр+2=20)

$C =$

1	2	5	7	8	9

4. $B[C[A[6]]] = B[C[4]] = B[7]=A[6]=4$, то есть $B[7]=4$, $C[4]=C[4]-1=6$, (Пр=Пр+2=22)

$C =$

1	2	5	6	8	9

5. $B[C[A[5]]] = B[C[1]] = B[1]=A[5]=1$, то есть $B[1]=1$, $C[1]=C[1]-1=0$, (Пр=Пр+2=24)

$C =$

0	2	5	6	8	9

6. $B[C[A[4]]] = B[C[3]] = B[5]=A[4]=3$, то есть $B[5]=3$, $C[3]=C[3]-1=5-1=4$, (Пр=Пр+2=26)

$C =$

0	2	4	6	8	9

И так далее.

Количество присваиваний для реализации сортировки подсчетом массива A размерности n =9 и при k =6 – Пр=n+(k-1)+n*2=9+(6-1)+2*9=32. Количество сравнений для поиска величины k равно (n - 1)=8. Общее количество операций Оп= n+(k-1)+n*2+(n-1)=4*n+k-2=36+6-2=40.

Таким образом, результирующий массив будет иметь вид

B=

1	2	3	4	5	6	7	8	9
1	1	2	3	3	4	4	4	6

Приклад покрокового виконання:

Input Data

0	4	2	2	0	0	1	1	0	1	0	2	4	2

Count Array

0	1	2	3	4
5	3	4	0	2

Sorted Data

0	0	0	0	0	1	1	1	2	2	2	2	4	4

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

2. Поразрядная сортировка (*LSD*).

LSD (least significant digit radix sort) - поразрядная сортировка сначала по младшей цифре.

Пусть в общем случае сортируемые числа $A=(a_1, a_2, \ldots, a_n)$ являются целыми и состоят из p цифр (более короткие числа дополняются нулями) с основанием системы счисления r. Число a_i имеет следующее представление: $a_i=(a_{i,p}, a_{i,p-1}, \ldots, a_{i,1})$, где $a_{i,p}$ – цифра старшего разряда, $a_{i,1}$ – цифра младшего разряда числа a_i. Поразрядная сортировка основана на том свойстве, что числа можно полностью отсортировать, выполняя сортировку по отдельным разрядам, начиная с самого младшего.

Алгоритм поразрядной сортировки состоит в следующем:

1) $k=1$. Выбираем младшую цифру числа.
2) Берем каждое число из массива A и помещаем его в конец одной из r очередей в зависимости от значений цифры в позиции k.
3) Восстанавливаем каждую очередь в массив A, начиная с очереди чисел с цифрой 0 и кончая очередью чисел с (r-1)-й цифрой. (После распределения массива A по очередям (по "карманам") выполняется операция конкатенации всех списков в один список.)
4) $k=k+1$. Выполняем пункты 2, 3 пока ($k \le p$), то есть до старшей значащей цифры числа.

(При $k=1$ помещаем каждое сортируемое число a_i в очередь с номером (a_i *mod* r). При $k=2$ число помещается в очередь с номером $\lfloor a_i / r \rfloor$, то есть номер очереди

равен наибольшему целому числу, равному или меньшему $\lfloor a_i / r \rfloor$, (другими словами, скобки здесь обозначают операцию взятия целой части числа).)

Процесс поразрядной сортировки для последовательности A=(25, 57, 48, 37, 12, 92, 86, 33) показан в таблице 1 (здесь r =10, p =2, n =8).

Сложность поразрядной сортировки составляет примерно $O(p \cdot n)$.

Для экономии памяти эффективно использовать линейные связанные списки из указателей для представления сортируемых элементов и очередей.

Таблица 1 Пример поразрядной сортировки (*LSD*)

Очереди для k=1		*Очереди для k=2*	
Разряд	Содержимое разряда	Разряд	Содержимое разряда
0		0	
1		1	12
2	12 92	2	25
3	33	3	33 37
4		4	48
5	25	5	57
6	86	6	
7	57 37	7	
8	48	8	86
9		9	92
Восстанавливаем массив A {12, 92, 33, 25, 86, 57, 37, 48}		Восстанавливаем массив A {12, 25, 33, 37, 48, 57, 86, 92}	

Чтобы увидеть, почему этот алгоритм работает правильно, достаточно заметить, что когда числа помещаются в одну очередь (в один “карманам”), например числа 33 и 37 в карман 3, то они будут располагаться в возрастающем порядке, поскольку в списке {12, 92, 33, 25, 86, 57, 37, 48} они упорядочены по самой правой цифре (k=1). Следовательно, в любом “кармане” числа также будут упорядочены по самой правой цифре. И, конечно, распределение чисел на втором этапе по “карманам” в соответствии с первой цифрой (k=2 в примере) гарантирует, что в объединенном списке все числа будут расставлены в возрастающем порядке.

1. Количество присваиваний при формировании очередей для k=1 равно Оп=n=8.

2. Количество операций при восстановлении массива равно r=10, то есть Оп=n+r=18.

3. Количество присваиваний при формировании очередей для k=2 равно n=8, то есть Оп=n+r+n=26.

4. Количество операций при восстановлении массива равно r=10, то есть Оп = n + r + n + r =36.

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

```
Enter operation: 3
How many elements?: 7
Min number: 0
Max number: 10

Result:  [5, 8, 7, 9, 1, 4, 3]

Enter operation: 4
Counting sort
[INFO] Counting sort starts...
[Count]  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[Initialized count]  [0, 1, 0, 1, 1, 1, 0, 1, 1, 1]
[Comulative count]  [0, 1, 1, 2, 3, 4, 4, 5, 6, 7]
[PROCESSING ARRAY]  [0, 3, 0, 0, 0, 0, 0] [0, 1, 1, 1, 3, 4, 4, 5, 6, 7]
[PROCESSING ARRAY]  [0, 3, 4, 0, 0, 0, 0] [0, 1, 1, 1, 2, 4, 4, 5, 6, 7]
[PROCESSING ARRAY]  [1, 3, 4, 0, 0, 0, 0] [0, 0, 1, 1, 2, 4, 4, 5, 6, 7]
[PROCESSING ARRAY]  [1, 3, 4, 0, 0, 0, 9] [0, 0, 1, 1, 2, 4, 4, 5, 6, 6]
[PROCESSING ARRAY]  [1, 3, 4, 0, 7, 0, 9] [0, 0, 1, 1, 2, 4, 4, 4, 6, 6]
[PROCESSING ARRAY]  [1, 3, 4, 0, 7, 8, 9] [0, 0, 1, 1, 2, 4, 4, 4, 5, 6]
[PROCESSING ARRAY]  [1, 3, 4, 5, 7, 8, 9] [0, 0, 1, 1, 2, 3, 4, 4, 5, 6]
[INFO] Counting sort finished! The result:  [1, 3, 4, 5, 7, 8, 9]

[INFO] Operation count:  23
Time:  0.0038793087005615234
```

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

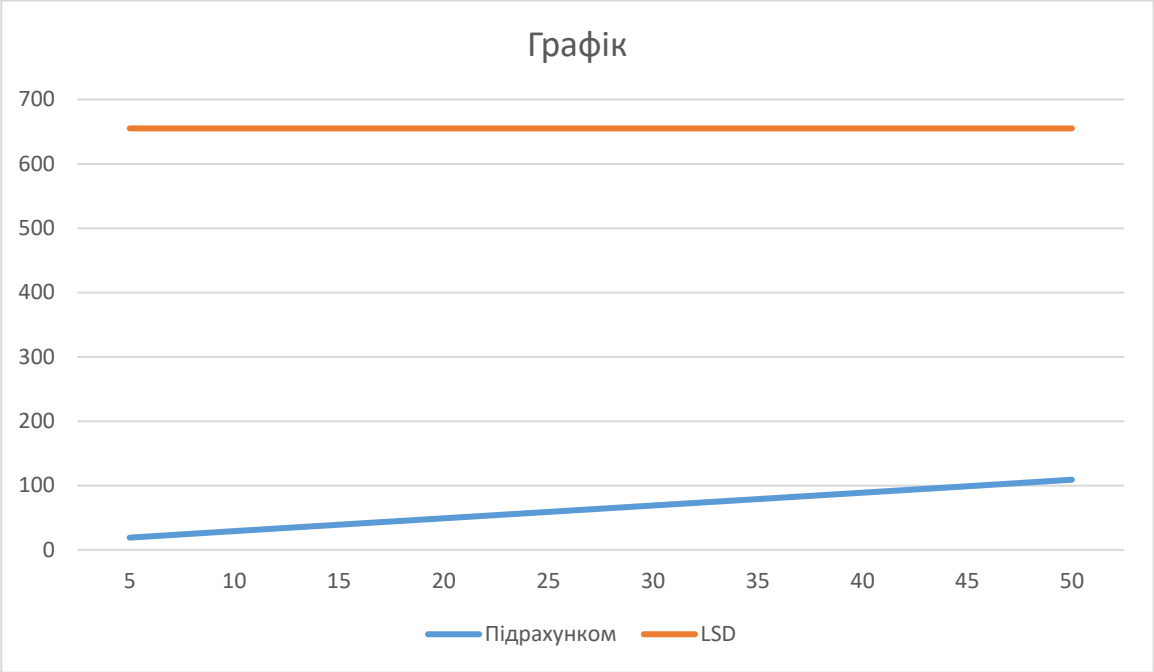
Частина 3. Порівняння алгоритмів

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