Львівський національний університет імені Івана Франка
Факультет електроніки та комп’ютерних технологій

Звіт про виконання лабораторної роботи №6

Швидке сортування. Порядкові статистики.

Виконав:
Студент групи ФЕП-22
Серафим Д.В.

Львів-2022

Теоретичні відомості (Лекція_8 Швидке сортування. Порядкові статистики.pdf)

Хід роботи:

Частина 1. Швидке сортування.

1. У бібліотеці Sort (створеній раніше), згідно описаних в теоретичній частині алгоритмів, створити функції QuickSort(…), Partition(…)та RandomizedPartition(…) для реалізації швидкого сортування одномірного масиву даних.

```
def quick_sort(main_array):
    if len(main_array) <= 1:  # Умова виходу з рекурсії
        return main_array

    supporting_element = main_array[len(main_array) - 1]  # Опорний елемент
(Останній в масиві)

    # supporting_element = randint(0, int(len(main_array) - 1))  # Опорний
елемент (Рандомно)

    left_part = list(filter(lambda x: x < supporting_element, main_array))  #
Ліва частина

    # Центральна частина
    # На випадок якщо є багато елементів які == опорному
    center_part = []
    for i in main_array:
        if i == supporting_element:
            center_part.append(i)

    right_part = list(filter(lambda x: x > supporting_element, main_array))  #
Права частина

    return quick_sort(left_part) + center_part + quick_sort(right_part)
```

2. Створити новий проект Lab_6_1 та підключити до нього бібліотеку Sort. У функції main() проекту реалізувати можливість введення одномірного масиву даних та відображення результатів сортування. Відкомпілювати проект та продемонструвати його роботу для одномірного масиву даних, отриманого від викладача.

```
from Sort import quick_sort, select_statistic
from random import randint

print(randint(0, 0))
"""
Автоматичне введення в масив
"""


def auto_input():
    main_array = []
    number_of_elements = (input(f"Вкажіть кількість елементів в масиві: "))
    frame_min_number = 0
    frame_max_number = int(input(f"Вкажіть максимальне число яке може входити в
```

```
масив: "))
    for i in range(int(number_of_elements)):
        main_array.append(randint(frame_min_number, frame_max_number))

    return main_array


main_array = auto_input()

"""
PART 1
"""

print(f"Вхідний масив                                    : {main_array}")

print(f"Після сортування методом Quick Sort              :
{quick_sort(main_array)}")
```

```
Вкажіть кількість елементів в масиві: 12
Вкажіть максимальне число яке може входити в масив: 32
Вхідний масив                                    : [22, 1, 26, 4, 12, 6, 0, 8, 21, 28, 7, 21]
Після сортування методом Quick Sort              : [0, 1, 4, 6, 7, 8, 12, 21, 21, 22, 26, 28]
```

```
Вкажіть кількість елементів в масиві: 17
Вкажіть максимальне число яке може входити в масив: 23
Вхідний масив                                  : [5, 0, 18, 3, 3, 4, 7, 13, 8, 5, 0, 4, 11, 11, 10, 1, 9]
Після сортування методом Quick Sort            : [0, 0, 1, 3, 3, 4, 4, 5, 5, 7, 8, 9, 10, 11, 11, 13, 18]
```

Частина 2. Порядкові статистики.

1. Створити новий проект Lab_6_2. Згідно описаного в теоретичній частині алгоритму, створити функцію RandomizedSelect(…) дляпошуку порядкових статистик. У функції main() проектуреалізувати можливість введення одномірного масиву даних та відображення результатів пошуку порядкових статистик у ньому.

```
from random import randint

"""
PART 1
"""


def quick_sort(main_array):
    if len(main_array) <= 1:  # Умова виходу з рекурсії
        return main_array

    supporting_element = main_array[len(main_array) - 1]  # Опорний елемент
(Останній в масиві)

    # supporting_element = randint(0, int(len(main_array) - 1))  # Опорний
елемент (Рандомно)
```

```
    left_part = list(filter(lambda x: x < supporting_element, main_array))  #
Ліва частина

    # Центральна частина
    # На випадок якщо є багато елементів які == опорному
    center_part = []
    for i in main_array:
        if i == supporting_element:
            center_part.append(i)

    right_part = list(filter(lambda x: x > supporting_element, main_array))  #
Права частина

    return quick_sort(left_part) + center_part + quick_sort(right_part)


"""
PART 2
"""


def select_statistic(main_array, i):  # Функція пошуку порядкової статистики
    buffer_array = quick_sort(main_array)
    task_2_3(buffer_array)  # Виклик завдання Частина_2_2
    return buffer_array[int(i) - 1]


def task_2_3(main_array) -> None:  # Завдання Частина_2_2
    print(f"Максимальне значення: {max(main_array)}")
    print(f"Мінімальне значення: {min(main_array)}")

    buffer_median = []

    if len(main_array) % 2 == 1:
        buffer_median.append(main_array[len(main_array) // 2])
        print(f"Медіана: {buffer_median}")
    else:
        buffer_median.append(main_array[len(main_array) // 2])
        buffer_median.append(main_array[(len(main_array) // 2) - 1])
        print(f"Медіана: {buffer_median}")
```

2. Підключити до проекту Lab_6_2 бібліотеку Sort з метою використання функції RandomizedPartition(…).

```
from Sort import quick_sort, select_statistic
from random import randint

print(randint(0, 0))
"""
Автоматичне введення в масив
"""


def auto_input():
    main_array = []
    number_of_elements = (input(f"Вкажіть кількість елементів в масиві: "))
    frame_min_number = 0
    frame_max_number = int(input(f"Вкажіть максимальне число яке може входити в
масив: "))
    for i in range(int(number_of_elements)):
        main_array.append(randint(frame_min_number, frame_max_number))

    return main_array
```

```python
main_array = auto_input()

"""
PART 1
"""

print(f"Вхідний масив                                    : {main_array}")

print(f"Після сортування методом Quick Sort              :
{quick_sort(main_array)}")


"""
PART 2
"""


print("-----------------------------------------------")

print("Зауваження!! Номер порядкової статистики не має перевищувати кількість
елементів в масиві!!")
i_index = input(f"Введіть номер порядкової статистики яку хочете знайти: ")
if int(i_index) > len(main_array) or int(i_index) < 0:
    print("Введено дані котрі не підлягають умові ):")
    print("Скоріше всього номер порядкової статистики перевищує кількість
елементів в масиві!!")
else:
    print(f"Вхідний масив: {main_array}")

    print(f"Результат пошуку порядкової статистики:
{select_statistic(main_array, i_index)}")

    print("-----------------------------------------------")
```

3. Відкомпілювати проект та продемонструвати його роботу для одномірного масиву даних, отриманого від викладача, зокрема, знайти максимальне, мінімальне значення та медіану.

```
Зауваження!! Номер порядкової статистики не має перевищувати кількість елементів в масиві!!
Введіть номер порядкової статистики яку хочете знайти: 8
Вхідний масив: [22, 1, 26, 4, 12, 6, 0, 8, 21, 28, 7, 21]
Максимальне значення: 28
Мінімальне значення: 0
Медіана: [12, 8]
Результат пошуку порядкової статистики: 21
```

```
Зауваження!! Номер порядкової статистики не має перевищувати кількість елементів в масиві!!
Введіть номер порядкової статистики яку хочете знайти: 9
Вхідний масив: [5, 0, 18, 3, 3, 4, 7, 13, 8, 5, 0, 4, 11, 11, 10, 1, 9]
Максимальне значення: 18
Мінімальне значення: 0
Медіана: [5]
Результат пошуку порядкової статистики: 5
```

Висновок: На цій лабораторній роботі я ознайомився з такими методами сортування: Швидке сортування. Порядкові статистики.