

Міністерство освіти і науки України
**Національний технічний університет України «Київський
політехнічний інститут імені Ігоря Сікорського»**
Факультет інформатики та обчислювальної техніки Кафедра ІІІ
Звіт

з лабораторної роботи №5
з дисципліни «Алгоритми та структури даних 2. Структури даних»
«Піраміди»

Виконав(ла) ІІІ-22, Андреева Уляна Андріївна
(шифр, прізвище, ім'я, по батькові)

Перевірила Халус Олена Андріївна
(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

МЕТА ЛАБОРАТОРНОЇ РОБОТИ	3
ЗАВДАННЯ	4
ВИКОНАННЯ	9
1. 3.1 ПСЕВДОКОД АЛГОРИТМУ	9
2. 3.2 ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ	9
3.2.1 Вихідний код	9
ВИСНОВОК	11

Практичне завдання №5

“Піраміди”

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи -

розв'язати наступну задачу визначення послідовності медіан для заданого вхідного масиву.

2 ЗАВДАННЯ

Задача формулюється наступним чином. Нехай заданий вхідний масив $A = [x_1, \dots, x_N]$. Припустимо, що елементи масиву поступають на вхід програм и послідовно:

в кожний момент часу розглядається новий елемент x_i . Необхідно для кожного i (від 1 до N) визначити медіану підмасиву $A' = [x_1, \dots, x_i]$, тобто медіану для масиву елементів, які були отримані програмою на даний момент часу. Необхідно розв'язати цю задачу, використовуючі структури даних пірамід і так, щоб кожна медіана визначалась за час $O(\log(i))$.

3.1 Псевдокод алгоритму

```
function heapify(list, n, i):
    largest = i
    l = 2 * i + 1
    r = 2 * i + 2

    if l < n and list[l] > list[largest]:
        largest = l

    if r < n and list[r] > list[largest]:
        largest = r

    if largest != i:
        swap(list[i], list[largest])
        heapify(list, n, largest)

function heapSort(list):
    n = list.length

    for i in range(n/2 - 1, -1, -1):
        heapify(list, n, i)
```

```

    for i in range(n-1, -1, -1):
        temp = list[0]
        list[0] = list[i]
        list[i] = temp

        heapify(list, i, 0)

    return list

function readInputFile(inputFile):
    array = []

    with open(inputFile, 'r') as f:
        n = int(f.readline())
        for i in range(n):
            array.append(int(f.readline()))

    return array

function main():
    inputFile = "input.txt"
    outputFile = "output.txt"

    try:
        array = readInputFile(inputFile)

        with open(outputFile, 'w') as f:
            subArray = []
            for i in range(len(array)):
                subArray.append(array[i])
                sortedSubArray = heapSort(subArray)
                n = len(sortedSubArray)

                if n % 2 == 1:
                    f.write(str(sortedSubArray[n // 2]) + "\n")
                else:
                    f.write(str(sortedSubArray[n // 2 - 1]) + " " + str(sortedSubArray[n
// 2]) + "\n")

    except IOError as e:
        print(e)

```

3.2 Вихідний код

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class MedianSequence {
    public static void main(String[] args) {
        String inputFile = "input.txt";
        String outputFile = "output.txt";

        try {
            List<Integer> array = readInputFile(inputFile);
            List<Integer> subArray = new ArrayList<>();

            try (FileWriter writer = new FileWriter(outputFile)) {
                for (int i = 0; i < array.size(); i++) {
                    subArray.add(array.get(i));
                    List<Integer> sortedSubArray = heapSort(new
ArrayList<>(subArray));
                    int n = sortedSubArray.size();

                    if (n % 2 == 1) {
                        // Непарна кількість елементів - повертаємо одну
медіану
                        writer.write(sortedSubArray.get(n / 2) + "\n");
                    } else {
                        // Парна кількість елементів - повертаємо дві
медіани через пробіл
                        writer.write(sortedSubArray.get(n / 2 - 1) + " " +
sortedSubArray.get(n / 2) + "\n");
                    }
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static List<Integer> heapSort(List<Integer> list) {
        int n = list.size();

        for (int i = n / 2 - 1; i >= 0; i--)
            heapify(list, n, i);

        for (int i = n - 1; i >= 0; i--) {
            int temp = list.get(0);
            list.set(0, list.get(i));
            list.set(i, temp);

            heapify(list, i, 0);
        }

        return list;
    }
}
```

```

private static void heapify(List<Integer> list, int n, int i) {
    int largest = i;
    int l = 2 * i + 1;
    int r = 2 * i + 2;

    if (l < n && list.get(l) > list.get(largest))
        largest = l;

    if (r < n && list.get(r) > list.get(largest))
        largest = r;

    if (largest != i) {
        int swap = list.get(i);
        list.set(i, list.get(largest));
        list.set(largest, swap);

        heapify(list, n, largest);
    }
}

private static List<Integer> readInputFile(String inputFile) throws
IOException {
    List<Integer> array = new ArrayList<>();

    try (BufferedReader reader = new BufferedReader(new
FileReader(inputFile))) {
        int n = Integer.parseInt(reader.readLine());
        for (int i = 0; i < n; i++) {
            array.add(Integer.parseInt(reader.readLine()));
        }
    }

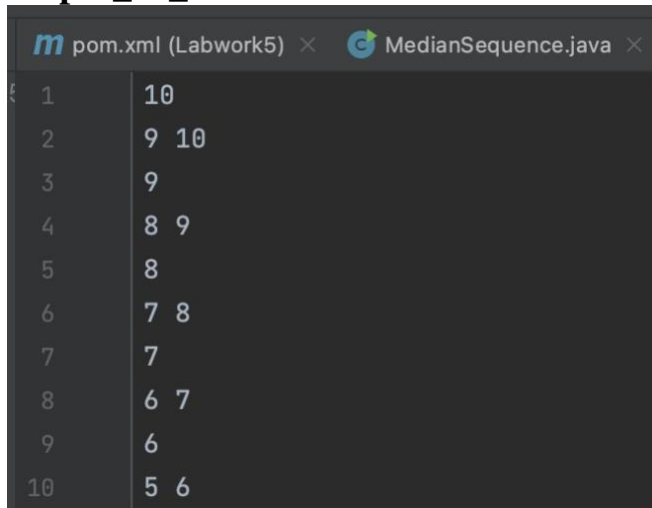
    return array;
}
}

```

3.2.1 Програмна реалізація input_02_10.txt

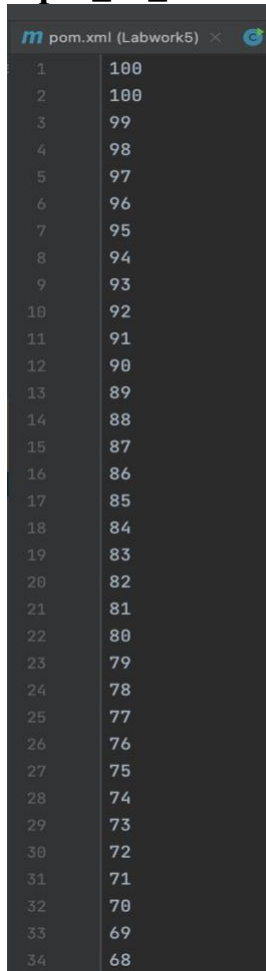
pom.xml (Labwork5) × MedianSequence.java ×	
1	10
2	10
3	9
4	8
5	7
6	6
7	5
8	4
9	3
10	2
11	1

output_02_10.txt



1	10
2	9 10
3	9
4	8 9
5	8
6	7 8
7	7
8	6 7
9	6
10	5 6

input_07_100.txt



1	100
2	100
3	99
4	98
5	97
6	96
7	95
8	94
9	93
10	92
11	91
12	90
13	89
14	88
15	87
16	86
17	85
18	84
19	83
20	82
21	81
22	80
23	79
24	78
25	77
26	76
27	75
28	74
29	73
30	72
31	71
32	70
33	69
34	68

output_07_100.txt

```

1 100
2 99 100
3 99
4 98 99
5 98
6 97 98
7 97
8 96 97
9 96
10 95 96
11 95
12 94 95
13 94
14 93 94
15 93
16 92 93
17 92
18 91 92
19 91
20 90 91
21 90
22 89 90
23 89
24 88 89
25 88
26 87 88
27 87
28 86 87
29 86
30 85 86
31 85
32 84 85
33 84
34 83 84

```

Опис алгоритму

$[10, 5, 6, 4, 7, 3, 8, 2, 9, 1, 10]$
 array size = 10
 subarray = 0
 subarray + 1 = 1
 $n \% 2 == 1 - \text{true} \leftarrow \text{uneven List}$
 return $10 : 2 = 5$
 subArray - size = 2
 else - an even quantity
 return 5, 6
 $\text{size} - \text{subarray} = 3$
 $n \% 2 == 1 - \text{true}$
 5 - return
 subarray size = 4
 an even quantity.
 continuation the same ...

5
5 6
5
5 6
5
5 6
5
5 6
5
5 6

Сортування підпоследовності чисел потрібно для знаходження медіани цієї підпоследовності. Медіана - це середнє значення відсортованої последовності чисел.

У даному коді використовується алгоритм Heap Sort для сортування підпоследовностей чисел. Він є одним з найшвидших алгоритмів сортування на практиці та має часову складність $O(n \log n)$. Сортування потрібно для того, щоб знайти медіану підпоследовності. Якщо підпоследовність містить непарну кількість елементів, то медіаною є центральний елемент підпоследовності. Якщо кількість елементів парна, то медіаною є середнє значення двох центральних елементів.

ВИСНОВОК

Виконавши цю лабораторну роботу, я навчилася програмувати алгоритм для знаходження медіани масиву за час $O(\log(n))$, а також детально дослідила таку структуру даних як піраміди.