Санкт-Петербургский Национальный Исследовательский Университет
Информационных Технологий, Механики и Оптики

Лабораторная работа №5

по дисциплине

«Алгоритмы и структуры данных»

Выполнил: Студент группы P3217

Сергачев Данила Дмитриевич

Преподаватели:

Романов Алексей Андреевич и

Волчек Дмитрий Геннадьевич

Санкт-Петербург

2018 г

Задание №1

Куча ли?

1.0 из 1.0 балла (оценивается)

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Структуру данных «куча», или, более конкретно, «неубывающая пирамида», можно реализовать на основе массива.

Для этого должно выполнятся основное свойство неубывающей пирамиды, которое заключается в том, что для каждого $1 \leq i \leq n$ выполняются условия:

- если $2i \leq n$, то $a[i] \leq a[2i]$;
- если $2i + 1 \leq n$, то $a[i] \leq a[2i + 1]$.

Дан массив целых чисел. Определите, является ли он неубывающей пирамидой.

Решение

```
package week5;

import mooc.*;
import week2.EdxIO;

public class task1 {
    static int[] arr;
    static int n;
   public static void main(String[] args){
        try (EdxIO io = EdxIO.create()) {
            n = io.nextInt();
            arr = new int[n+1];
            boolean isHeap = true;

            for(int i = 1; i <= n; i++){
                arr[i] = io.nextInt();
            }

            for(int i = 1; i <= n; i++){
                isHeap &= isHeapify(i);
            }

            io.println(isHeap ? "YES" : "NO");
        }
```

```
    }

    public static boolean isHeapify(int index){
        boolean isHeap = true;
        if(n >= index*2){
            isHeap &= arr[index] <= arr[index*2];

            if(n - 1 >= index*2){
                isHeap &= arr[index] <= arr[index*2 + 1];
            }
        }

        return isHeap;
    }
}
```

Результат работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.265	39075840	10945420	5
1	OK	0.125	21602304	14	4
2	OK	0.109	21594112	14	5
3	OK	0.109	21577728	1092	5
4	OK	0.125	21598208	889	5
5	OK	0.109	21659648	1099	4
6	OK	0.125	21594112	1100	5
7	OK	0.078	21606400	1098	5
8	OK	0.125	21569536	1093	5
9	OK	0.125	21573632	1105	4
10	OK	0.156	21581824	1095	4
11	OK	0.125	21848064	10931	5
12	OK	0.125	21835776	8837	5

Задание №2

Очередь с приоритетами

2.0 из 2.0 баллов (оценивается)

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Реализуйте очередь с приоритетами. Ваша очередь должна поддерживать следующие операции: добавить элемент, извлечь минимальный элемент, уменьшить элемент, добавленный во время одной из операций.

Решение

```
package week5;

import mooc.*;

public class task2 {
    public static void main(String[] args){
        try (EdxIO io = EdxIO.create()) {
            int n = io.nextInt();
            MinHeap heap = new MinHeap(n);
            StringBuilder st = new StringBuilder();

            for(int i = 0; i < n; i++){
                switch(io.nextChar()){
                    case 'A' :{
                        heap.addElem(io.nextInt());
                        break;
                    }
                    case 'X' :{
                        heap.deleteElem(st);
                        break;
                    }
                    case 'D' :{
                        heap.insert(io.nextInt(), io.nextInt());
                        break;
                    }
                }
            }

            io.print(st);
        }
    }

    static class MinHeap {
        private HeapItem[] innerArr;
        private int size;
```

```
private int top;
private boolean isSorted;
private int minIndex;

public MinHeap(int size){
    this.size = size;
    this.innerArr = new HeapItem[size];
    this.top = -1;
    this.minIndex = this.top;
    this.isSorted = false;
}

public void addElem(int newElem){
    this.minIndex = ++top;
    HeapItem item = new HeapItem(top, newElem);
    innerArr[top] = item;

    if(isSorted)
        isSorted = !isSorted;
}

public void deleteElem(StringBuilder st){
    if(!isEmpty()){
        if(isSorted){
            st.append(String.format("%d\n", innerArr[minIndex].value));
        } else {
            buildMaxHeap();
            isSorted = true;

            st.append(String.format("%d\n", innerArr[minIndex].value));
        }

        top--;
        minIndex = top;
    } else {
        st.append("*\n");
    }

    for(int i = 0; i < top + 1; i++){
        innerArr[i].heapIndex = i;
    }
}

public void insert(int index, int value) {
    if(top + 1 >= index){
        innerArr[index - 1].value = value;
        isSorted = false;
    }
}

private void buildMaxHeap(){
```

```
            for(int index = top/2; index >= 0; index--){
                setMaxHeapify(index);
            }
        }

        private void setMaxHeapify(int index){
            int lagestIndex, leftIndex, rightIndex;

            leftIndex = getLeftIndex(index);
            rightIndex = getRightIndex(index);

            if(isExistLeftElem(index) &&
             innerArr[innerArr[leftIndex].heapIndex].value >
              innerArr[innerArr[index].heapIndex].value) {
                lagestIndex = leftIndex;
            } else {
                lagestIndex = index;
            }

            if(isExistRightElem(index) &&
                innerArr[innerArr[rightIndex].heapIndex].value >
                innerArr[innerArr[lagestIndex].heapIndex].value) {
                    lagestIndex = rightIndex;
                }

            if(lagestIndex != index){
                swap(index, lagestIndex);
                setMaxHeapify(lagestIndex);
            }
        }

        private int getLeftIndex(int index) { return index * 2 + 1; }

        private int getRightIndex(int index) { return (index + 1) * 2; }

        private boolean isExistLeftElem(int index) { return top >= index * 2 + 1;
}

        private boolean isExistRightElem(int index) { return top >= (index + 1) *
2; }

        private void swap(int index1, int index2) {
            int tmp = innerArr[index1].heapIndex;
            innerArr[index1].heapIndex = innerArr[index2].heapIndex;
            innerArr[index2].heapIndex = tmp;
            if(innerArr[index1].value < innerArr[minIndex].value)
                minIndex = index1;
        }

        private boolean isEmpty() { return top < 0; }
```

```
        class HeapItem{
            public int index;
            public int heapIndex;
            public int value;

            public HeapItem(int index, int value){
                this.index = index;
                this.heapIndex = index;
                this.value = value;
            }
        }
    }
}
```

Результат работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.828	226906112	12083657	5694235
1	OK	0.031	10502144	37	12
2	OK	0.031	10477568	6	3
3	OK	0.031	10416128	11	3
4	OK	0.031	10444800	22	4
5	OK	0.031	10461184	19	6
6	OK	0.031	10571776	19	6
7	OK	0.031	10465280	19	6
8	OK	0.062	10498048	48	19
9	OK	0.031	10461184	58	29
10	OK	0.031	10420224	57	28