Министерство образования и науки Российской Федерации
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Дисциплина «**Алгоритмы и структуры данных**»

ОТЧЁТ

по лабораторным работам за 5 неделю

Студент **Пастухов К.А. группы P3218**

Санкт-Петербург
2018 г.

Куча ли?

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

Выполнение

```
from edx_io import edx_io


def main():
    with edx_io() as io:
        data = [int(t.decode()) for t in io.all_tokens()]
        N = int(data.pop(0))
        is_heap = True
        for i in range(1, N // 2 + N % 2):
            if (data[i - 1] > data[2 * i - 1] or data[i - 1] > data[2 * i]):
                is_heap = False
                break
        io.write("YES" if is_heap else "NO")


if __name__ == "__main__":
    main()
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.140	136544256	10945420	3
1	OK	0.046	10973184	14	2
2	OK	0.062	10936320	14	3
3	OK	0.062	10997760	1092	3
4	OK	0.109	10952704	889	3
5	OK	0.062	10985472	1099	2
6	OK	0.046	10973184	1100	3
7	OK	0.062	10928128	1098	3
8	OK	0.062	10993664	1093	3
9	OK	0.046	11026432	1105	2
10	OK	0.062	10964992	1095	2
11	OK	0.062	11161600	10931	3
12	OK	0.046	10977280	8837	3
13	OK	0.062	11087872	10928	2
14	OK	0.062	11096064	10934	3
15	OK	0.046	11104256	10989	3
16	OK	0.062	11087872	10934	3
17	OK	0.062	11100160	10978	2
18	OK	0.046	11108352	10960	2
19	OK	0.078	12226560	109474	3
20	OK	0.062	11968512	89095	3
21	OK	0.062	12185600	109362	2
22	OK	0.062	12345344	109479	3
23	OK	0.062	12238848	109486	3
24	OK	0.078	12288000	109443	2
25	OK	0.062	12152832	109565	2
26	OK	0.062	12189696	109493	2
27	OK	0.140	24178688	1094387	3
28	OK	0.156	21676032	886879	3
29	OK	0.140	24182784	1094726	2
30	OK	0.156	24956928	1094117	3
31	OK	0.156	24911872	1094308	3
32	OK	0.140	24965120	1094215	3
33	OK	0.156	25014272	1094084	2
34	OK	0.140	24768512	1094403	2
35	OK	1.125	136491008	10944156	3
36	OK	1.062	112066560	8876466	3
37	OK	0.906	136544256	10945179	2
38	OK	1.140	136540160	10945420	3
39	OK	1.125	136536064	10943533	3
40	OK	1.140	136503296	10944594	3
41	OK	1.078	136491008	10944330	2
42	OK	0.921	136470528	10944738	2

Очередь с приоритетами

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

Выполнение

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Openedu.Week5
{
    class week5_lab2
    {
        public static void Main(string[] args)
        {
            var app = new week5_lab2();
            app.DoWork(args);
        }

        private void DoWork(string[] args)
        {
            using (StreamWriter sw = new StreamWriter("output.txt"))
            {
                string[] stdin = File.ReadAllLines("input.txt");
                Console.SetOut(sw);
                QueueWithPriorities queue = new QueueWithPriorities();

                for (int i = 1; i < stdin.Length; i++)
                    switch (stdin[i][0]) {
                        case 'A': queue.Insert(i, int.Parse(stdin[i].Split(' ')
[1])); break;
                        case 'X': queue.Extract(); break;
                        case 'D': queue.Decrease(int.Parse(stdin[i].Split(' ')
[1]), int.Parse(stdin[i].Split(' ')[2])); break;
                    }
            }
        }


        public class QueueWithPriorities
        {
            public class Element
            {
                public int CurrentIndex { get; set; }
                public long Value { get; set; }
            }

            public Dictionary<int, Element> References = new Dictionary<int,
Element>();
            public Element[] array = new Element[6000000];
            public Element Top { get { return array[0]; } }
            public int HeapSize { get; private set; }
```

```
            public void Extract()
            {
                if (this.HeapSize == 0)
                    Console.WriteLine("*");
                else
                {
                    Console.WriteLine(this.Top.Value);
                    this.HeapSize--;
                    this.SwapElementsWithIndexes(0, this.HeapSize);
                    this.Heapify(0);
                }
            }

            public void Decrease(int lineIndex, int value)
            {
                int index = References[lineIndex].CurrentIndex;
                array[index].Value = value;
                while (index > 0 && array[this.Parent(index)].Value >
array[index].Value)
                {
                    this.SwapElementsWithIndexes(index, this.Parent(index));
                    index = this.Parent(index);
                }
            }

            public void Insert(int lineIndex, int value)
            {
                array[this.HeapSize] = new Element { Value = int.MaxValue,
CurrentIndex = HeapSize };
                References.Add(lineIndex, array[this.HeapSize]);
                this.HeapSize++;
                this.Decrease(lineIndex, value);
            }

            private int Parent(int index)
            {
                return (index + 1) / 2 - 1;
            }

            private void Heapify(int index)
            {
                int rightChildIndex = (index + 1) * 2;
                int leftChildIndex = rightChildIndex - 1;
                int lowestIndex = int.MinValue;

                if (leftChildIndex < this.HeapSize &&
array[leftChildIndex].Value < array[index].Value)
                    lowestIndex = leftChildIndex;
                else
                    lowestIndex = index;

                if (rightChildIndex < this.HeapSize &&
array[rightChildIndex].Value < array[lowestIndex].Value)
                    lowestIndex = rightChildIndex;

                if (lowestIndex != index)
                {
                    this.SwapElementsWithIndexes(lowestIndex, index);
                    this.Heapify(lowestIndex);
```

```
                }
            }

            private void SwapElementsWithIndexes(int a, int b)
            {
                array[a].CurrentIndex = b;
                array[b].CurrentIndex = a;
                Element temp = array[a];
                array[a] = array[b];
                array[b] = temp;
            }
        }
    }
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.828	226897920	12083657	5694235
1	OK	0.031	10616832	37	12
2	OK	0.031	10579968	6	3
3	OK	0.031	10625024	11	3
4	OK	0.031	10612736	22	4
5	OK	0.031	10727424	19	6
6	OK	0.031	10649600	19	6
7	OK	0.015	10596352	19	6
8	OK	0.031	10629120	48	19
9	OK	0.031	10665984	58	29
10	OK	0.031	10665984	57	28
11	OK	0.046	10706944	48	19
12	OK	0.031	10620928	58	29
13	OK	0.031	10432512	57	28
14	OK	0.031	10465280	828	573
15	OK	0.078	10436608	1037	369
16	OK	0.031	10493952	828	573
17	OK	0.015	10481664	988	404
18	OK	0.031	10493952	1082	300
19	OK	0.078	10473472	1139	240
20	OK	0.031	10461184	930	377
21	OK	0.031	10518528	1190	280
22	OK	0.031	10739712	8184	5678
23	OK	0.031	10878976	10768	3637
24	OK	0.031	10682368	8206	5700
25	OK	0.031	10858496	9903	3928
26	OK	0.031	10813440	10814	3000
27	OK	0.031	10760192	11338	2400
28	OK	0.031	10805248	11138	3582
29	OK	0.031	10801152	10904	3851
30	OK	0.062	61247488	81951	56944