**Министерство образования и науки Российской Федерации**

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,   
МЕХАНИКИ И ОПТИКИ**

Факультет программной инженерии и компьютерной техники  
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Aлгоритмы и структуры данных»

**ОТЧЁТ**

по лабораторной работе №5 (Week 5 Openedu)

Студент Луговских Савелий Михайлович Р3218

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

[Задача 1 Куча ли? 3](#_Toc4451028)

[Исходный код к задаче 1 3](#_Toc4451029)

[Бенчмарк к задаче 1 4](#_Toc4451030)

[Задача 2 Очередь с приоритетами 5](#_Toc4451031)

[Исходный код к задаче 2 6](#_Toc4451032)

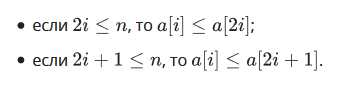
[Бенчмарк к задаче 2 8](#_Toc4451033)

# Задача 1 Куча ли?

|  |  |
| --- | --- |
| 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 выполняются условия:



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

#### Формат входного файла

Первая строка входного файла содержит целое число . Вторая строка содержит целых чисел, по модулю не превосходящих .

#### Формат выходного файла

Выведите «YES», если массив является неубывающей пирамидой, и «NO» в противном случае.

#### Примеры

|  |  |
| --- | --- |
| input.txt | output.txt |
| 5 1 0 1 2 0 | NO |
| 5 1 3 2 5 4 | YES |

## Исходный код к задаче 1

class Lab5\_1

{

public static void Main(string[] args)

{

var app = new Lab5\_1();

app.DoWork(args);

}

private void DoWork(string[] args)

{

using (StreamWriter sw = new StreamWriter("output.txt"))

{

int[] stdin = File.ReadAllLines("input.txt")[1].Split(' ')

.Select(x => int.Parse(x)).ToArray();

if (IsHeap(stdin))

sw.WriteLine("YES");

else

sw.WriteLine("NO");

}

}

static bool IsHeap(int[] array)

{

bool isHeap = true;

for (int i = array.Length / 2 - 1; i >= 0 && isHeap; i--)

{

var index = i + 1;

int rigthChildIndex = index \* 2;

int leftChildIndex = index \* 2 - 1;

if (leftChildIndex < array.Length &&

array[leftChildIndex] < array[index - 1])

isHeap = false;

if (rigthChildIndex < array.Length &&

array[rigthChildIndex] < array[index - 1])

isHeap = false;

}

return isHeap;

}

}

## Бенчмарк к задаче 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.515 | 109912064 | 10945420 | 5 |
| 1 | OK | 0.031 | 10887168 | 14 | 4 |
| 2 | OK | 0.015 | 10870784 | 14 | 5 |
| 3 | OK | 0.031 | 10895360 | 1092 | 5 |
| 4 | OK | 0.031 | 10883072 | 889 | 5 |
| 5 | OK | 0.046 | 10907648 | 1099 | 4 |
| 6 | OK | 0.031 | 10924032 | 1100 | 5 |
| 7 | OK | 0.031 | 10924032 | 1098 | 5 |
| 8 | OK | 0.031 | 10928128 | 1093 | 5 |
| 9 | OK | 0.015 | 10960896 | 1105 | 4 |
| 10 | OK | 0.031 | 10919936 | 1095 | 4 |
| 11 | OK | 0.031 | 11112448 | 10931 | 5 |
| 12 | OK | 0.031 | 11091968 | 8837 | 5 |
| 13 | OK | 0.031 | 11087872 | 10928 | 4 |
| 14 | OK | 0.031 | 11067392 | 10934 | 5 |
| 15 | OK | 0.031 | 11145216 | 10989 | 5 |
| 16 | OK | 0.046 | 11075584 | 10934 | 5 |
| 17 | OK | 0.031 | 11051008 | 10978 | 4 |
| 18 | OK | 0.031 | 11046912 | 10960 | 4 |
| 19 | OK | 0.031 | 12124160 | 109474 | 5 |
| 20 | OK | 0.031 | 12021760 | 89095 | 5 |
| 21 | OK | 0.031 | 12144640 | 109362 | 4 |
| 22 | OK | 0.031 | 12120064 | 109479 | 5 |
| 23 | OK | 0.031 | 12103680 | 109486 | 5 |
| 24 | OK | 0.046 | 12103680 | 109443 | 4 |
| 25 | OK | 0.031 | 12132352 | 109565 | 4 |
| 26 | OK | 0.031 | 12161024 | 109493 | 4 |
| 27 | OK | 0.062 | 21729280 | 1094387 | 5 |
| 28 | OK | 0.078 | 20934656 | 886879 | 5 |
| 29 | OK | 0.062 | 21766144 | 1094726 | 4 |
| 30 | OK | 0.062 | 21786624 | 1094117 | 5 |
| 31 | OK | 0.093 | 21798912 | 1094308 | 5 |
| 32 | OK | 0.062 | 21737472 | 1094215 | 5 |
| 33 | OK | 0.062 | 21766144 | 1094084 | 4 |
| 34 | OK | 0.078 | 21774336 | 1094403 | 4 |
| 35 | OK | 0.468 | 109912064 | 10944156 | 5 |
| 36 | OK | 0.421 | 99569664 | 8876466 | 5 |
| 37 | OK | 0.515 | 109883392 | 10945179 | 4 |
| 38 | OK | 0.453 | 109862912 | 10945420 | 5 |
| 39 | OK | 0.500 | 106758144 | 10943533 | 5 |
| 40 | OK | 0.500 | 109903872 | 10944594 | 5 |
| 41 | OK | 0.500 | 109891584 | 10944330 | 4 |
| 42 | OK | 0.468 | 109834240 | 10944738 | 4 |

# Задача 2 Очередь с приоритетами

|  |  |
| --- | --- |
| 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.

#### Формат входного файла

В первой строке входного файла содержится число - число операций с очередью.

Следующие n cтрок содержат описание операций с очередью, по одному описанию в строке. Операции могут быть следующими:

* A x — требуется добавить элемент x в очередь.
* X— требуется удалить из очереди минимальный элемент и вывести его в выходной файл. Если очередь пуста, в выходной файл требуется вывести звездочку «∗».
* D x y — требуется заменить значение элемента, добавленного в очередь операцией A в строке входного файла номер x+1, на y. Гарантируется, что в строке действительно находится операция A, что этот элемент не был ранее удален операцией X, и что y меньше, чем предыдущее значение этого элемента.

В очередь помещаются и извлекаются только целые числа, не превышающие по модулю .

#### Формат выходного файла

Выведите последовательно результат выполнения всех операций X, по одному в каждой строке выходного файла. Если перед очередной операцией X очередь пуста, выведите вместо числа звездочку «∗».

#### Пример

|  |  |
| --- | --- |
| input.txt | output.txt |
| 8 A 3 A 4 A 2 X D 2 1 X X X | 2 1 3 \* |

## Исходный код к задаче 2

class Lab5\_2

{

public static void Main(string[] args)

{

var app = new Lab5\_2();

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;

}

}

}

## Бенчмарк к задаче 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 1.859 | 226975744 | 12083657 | 5694235 |
| 1 | OK | 0.000 | 10493952 | 37 | 12 |
| 2 | OK | 0.031 | 10412032 | 6 | 3 |
| 3 | OK | 0.015 | 10407936 | 11 | 3 |
| 4 | OK | 0.031 | 10403840 | 22 | 4 |
| 5 | OK | 0.031 | 10477568 | 19 | 6 |
| 6 | OK | 0.031 | 10407936 | 19 | 6 |
| 7 | OK | 0.031 | 10473472 | 19 | 6 |
| 8 | OK | 0.031 | 10420224 | 48 | 19 |
| 9 | OK | 0.015 | 10489856 | 58 | 29 |
| 10 | OK | 0.031 | 10473472 | 57 | 28 |
| 11 | OK | 0.031 | 10448896 | 48 | 19 |
| 12 | OK | 0.046 | 10489856 | 58 | 29 |
| 13 | OK | 0.031 | 10465280 | 57 | 28 |
| 14 | OK | 0.031 | 10428416 | 828 | 573 |
| 15 | OK | 0.031 | 10498048 | 1037 | 369 |
| 16 | OK | 0.046 | 10489856 | 828 | 573 |
| 17 | OK | 0.031 | 10424320 | 988 | 404 |
| 18 | OK | 0.031 | 10436608 | 1082 | 300 |
| 19 | OK | 0.015 | 10498048 | 1139 | 240 |
| 20 | OK | 0.031 | 10432512 | 930 | 377 |
| 21 | OK | 0.031 | 10612736 | 1190 | 280 |
| 22 | OK | 0.031 | 10743808 | 8184 | 5678 |
| 23 | OK | 0.031 | 10813440 | 10768 | 3637 |
| 24 | OK | 0.031 | 10801152 | 8206 | 5700 |
| 25 | OK | 0.015 | 10780672 | 9903 | 3928 |
| 26 | OK | 0.031 | 10764288 | 10814 | 3000 |
| 27 | OK | 0.031 | 10752000 | 11338 | 2400 |
| 28 | OK | 0.031 | 10862592 | 11138 | 3582 |
| 29 | OK | 0.031 | 10829824 | 10904 | 3851 |
| 30 | OK | 0.062 | 61206528 | 81951 | 56944 |
| 31 | OK | 0.093 | 61452288 | 110901 | 36274 |
| 32 | OK | 0.062 | 61325312 | 81971 | 56964 |
| 33 | OK | 0.062 | 61661184 | 99351 | 39719 |
| 34 | OK | 0.062 | 61825024 | 107882 | 30000 |
| 35 | OK | 0.078 | 62025728 | 113181 | 24000 |
| 36 | OK | 0.078 | 61464576 | 112799 | 37474 |
| 37 | OK | 0.078 | 61419520 | 114106 | 37576 |
| 38 | OK | 0.187 | 76775424 | 819273 | 569265 |
| 39 | OK | 0.203 | 75620352 | 1143615 | 361526 |
| 40 | OK | 0.187 | 76890112 | 819455 | 569447 |
| 41 | OK | 0.187 | 77103104 | 992441 | 396009 |
| 42 | OK | 0.187 | 77557760 | 1079125 | 300000 |
| 43 | OK | 0.234 | 80744448 | 1131016 | 240000 |
| 44 | OK | 0.156 | 75272192 | 1175194 | 377350 |
| 45 | OK | 0.171 | 75378688 | 1174192 | 378071 |
| 46 | OK | 1.687 | 166100992 | 8194244 | 5694235 |
| 47 | OK | 1.531 | 160210944 | 11753433 | 3632457 |
| 48 | OK | 1.437 | 165986304 | 8193883 | 5693874 |
| 49 | OK | 1.578 | 181252096 | 9926125 | 3963652 |
| 50 | OK | 1.765 | 223821824 | 10792079 | 3000000 |
| 51 | OK | 1.859 | 226975744 | 11312176 | 2400000 |
| 52 | OK | 1.078 | 160870400 | 12078250 | 3794039 |
| 53 | OK | 1.062 | 160653312 | 12083657 | 3795822 |