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 «Алгоритмы и структуры данных»

ОТЧЁТ

по лабораторной работе №5 (Week 5 Openedu)

Студенка Жетесова Дана группы P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

Куча ли?

0 возможных балла (не 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 $1 \le i \le n$ выполняются условия:

если $2i \le n$, то $a[i] \le a[2i]$;

если $2i+1 \le n$, то $a[i] \le a[2i+1]$.

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

Формат входного файла

Первая строка входного файла содержит целое число n ($1 \le n \le 106$). Вторая строка содержит n целых чисел, по модулю не превосходящих $2 \cdot 109$.

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

```
#include <fstream>
#include <iostream>

using namespace std;

int main() {
   ifstream input ("input.txt");
   ofstream output ("output.txt");

   int n;
   input >> n;
   int half = n / 2;
   int* heap = new int[n + 1];

   for (int i = 1; i <= n; i++) {
      input >> heap[i];
   }

   for (int i = 1; i < half; i++) {
      if (!(heap[i] <= heap[i * 2] && heap[i] <= heap[i * 2 + 1])) {
         output << "NO";
         return 0;
      }
```

```
    }

    output << "YES";
    return 0;
}
```

Бенчмарк к задаче 1

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.062	6356992	10945420	3
1	OK	0.015	2338816	14	2
2	OK	0.000	2334720	14	3
3	OK	0.015	2334720	1092	3
4	OK	0.000	2342912	889	3
5	OK	0.015	2338816	1099	2
6	OK	0.015	2342912	1100	3
7	OK	0.015	2338816	1098	3
8	OK	0.015	2338816	1093	3
9	OK	0.000	2338816	1105	2
10	OK	0.015	2338816	1095	2
11	OK	0.015	2347008	10931	3
12	OK	0.000	2359296	8837	3
13	OK	0.000	2347008	10928	2
14	OK	0.015	2342912	10934	3
15	OK	0.015	2347008	10989	3
16	OK	0.000	2338816	10934	3
17	OK	0.000	2355200	10978	2
18	OK	0.000	2355200	10960	2
19	OK	0.015	2383872	109474	3
20	OK	0.015	2379776	89095	3
21	OK	0.015	2379776	109362	2
22	OK	0.000	2371584	109479	3
23	OK	0.015	2396160	109486	3

24	OK	0.015	2400256	109443	2
25	OK	0.015	2392064	109565	2
26	OK	0.015	2375680	109493	2
27	OK	0.109	2744320	1094387	3
28	OK	0.078	2748416	886879	3
29	OK	0.109	2732032	1094726	2
30	OK	0.109	2748416	1094117	3
31	OK	0.109	2732032	1094308	3
32	OK	0.109	2727936	1094215	3
33	OK	0.109	2744320	1094084	2
34	OK	0.109	2748416	1094403	2
35	OK	1.031	6356992	10944156	3
36	OK	0.812	6352896	8876466	3
37	OK	1.046	6352896	10945179	2
38	OK	1.046	6356992	10945420	3
39	OK	1.046	6352896	10943533	3
40	OK	1.031	6352896	10944594	3
41	OK	1.046	6348800	10944330	2
42	OK	1.062	6356992	10944738	2

Очередь с приоритетами

0 возможных балла (не 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.

Формат входного файла

В первой строке входного файла содержится число n ($1 \le n \le 10^6$) - число операций с очередью. Следующие n строк 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 x+1 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 109.

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

```
#include <fstream>
#include <iostream>

using namespace std;

class Heap {
public:
  Heap() : heap(new Node[1'000'001]), size(0), positions(new int[1'000'001]) {}

  struct Node {
    int position;
    int value;

    bool operator<(const Node &rhs) {
      return value < rhs.value;
    }
  };

  int get_size() {
    return size;
  }

  void append(int value, int position) {
    heap[++size] = {position, value};
    positions[position] = size;
    siftUp(size);
  }
```

```
  int remove() {
    int result = heap[1].value;
    heap[1] = heap[size--];
    positions[heap[1].position] = 1;
    heapify(1);
    return result;
  }

  void change(int position, int new_value) {
    heap[positions[position]].value = new_value;
    siftUp(positions[position]);
  }

  void printStats() {
    for (int i = 1; i <= size; i++)
      cout << '{' << heap[i].value << " : " << heap[i].position << "} ";
    cout << '\n';
    for (int i = 1; i <= size; i++)
      cout << '{' << positions[i] << "} ";
    cout << '\n';
  }

private:
  void siftUp(int index) {
    while (heap[index] < heap[index / 2] && index > 1) {
      swap(positions[heap[index].position], positions[heap[index / 2].position]);
      swap(heap[index], heap[index / 2]);
      index /= 2;
    }
  }

  int min_index(int index1, int index2, int index3) {
    int min = index1;
    Node min_result = heap[index1];
    if (heap[index2] < min_result) {
      min_result = heap[index2];
      min = index2;
    }
    if (heap[index3] < min_result) {
      min = index3;
    }
    return min;
  }

  void heapify(int index) {
    int prev_index;
    while (index * 2 <= size) {
      prev_index = index;
      if (index * 2 + 1 <= size) {
        index = min_index(index, index * 2, index * 2 + 1);
      } else {
        index = min_index(index, index, index * 2);
      }
```

```
            if (index == prev_index)
                break;
            swap(positions[heap[prev_index].position], positions[heap[index].position]);
            swap(heap[prev_index], heap[index]);
        };
    }


    Node *heap;
    int *positions;
    int size;
};

int main() {
    ifstream input("input.txt");
    ofstream output("output.txt");

    int n, number1, number2;
    input >> n;
    Heap heap;

    char temp;
    for (int i = 1; i <= n; i++) {
        input >> temp;
        if (temp == 'A') {
            input >> number1;
            heap.append(number1, i);
            //heap.printStats();
            continue;
        }
        if (temp == 'X') {
            if (heap.get_size() == 0) {
                output << '*' << '\n';
            } else {
                output << heap.remove() << '\n';
            }
            //heap.printStats();
            continue;
        }
        if (temp == 'D') {
            input >> number1 >> number2;
            heap.change(number1, number2);
            //heap.printStats();
        }
    }
}
```

Бенчмарк к задаче 2

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.578	11980800	12083657	5694235

1	OK	0.015	2375680	37	12
2	OK	0.015	2363392	6	3
3	OK	0.015	2371584	11	3
4	OK	0.000	2383872	22	4
5	OK	0.000	2371584	19	6
6	OK	0.015	2371584	19	6
7	OK	0.015	2387968	19	6
8	OK	0.000	2371584	48	19
9	OK	0.000	2387968	58	29
10	OK	0.000	2387968	57	28
11	OK	0.015	2371584	48	19
12	OK	0.000	2371584	58	29
13	OK	0.000	2387968	57	28
14	OK	0.015	2371584	828	573
15	OK	0.000	2371584	1037	369
16	OK	0.015	2387968	828	573
17	OK	0.015	2371584	988	404
18	OK	0.000	2371584	1082	300
19	OK	0.000	2371584	1139	240
20	OK	0.000	2387968	930	377
21	OK	0.000	2371584	1190	280
22	OK	0.015	2375680	8184	5678
23	OK	0.015	2371584	10768	3637
24	OK	0.000	2367488	8206	5700
25	OK	0.000	2375680	9903	3928
26	OK	0.000	2387968	10814	3000
27	OK	0.015	2392064	11338	2400
28	OK	0.000	2375680	11138	3582
29	OK	0.000	2371584	10904	3851

30	OK	0.015	2424832	81951	56944
31	OK	0.031	2420736	110901	36274
32	OK	0.031	2428928	81971	56964
33	OK	0.015	2445312	99351	39719
34	OK	0.015	2457600	107882	30000
35	OK	0.015	2465792	113181	24000
36	OK	0.031	2408448	112799	37474
37	OK	0.015	2408448	114106	37576
38	OK	0.125	2977792	819273	569265
39	OK	0.156	2916352	1143615	361526
40	OK	0.125	2981888	819455	569447
41	OK	0.125	3182592	992441	396009
42	OK	0.125	3276800	1079125	300000
43	OK	0.140	3342336	1131016	240000
44	OK	0.156	2785280	1175194	377350
45	OK	0.156	2789376	1174192	378071
46	OK	1.312	8380416	8194244	5694235
47	OK	1.578	7716864	11753433	3632457
48	OK	1.250	8380416	8193883	5693874
49	OK	1.281	10379264	9926125	3963652
50	OK	1.296	11378688	10792079	3000000
51	OK	1.312	11980800	11312176	2400000
52	OK	1.500	6385664	12078250	3794039
53	OK	1.531	6389760	12083657	3795822