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)

Студент Дунаев Алексей Игоревич

Группа P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

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:

- 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 $n\left(1 \le n \le 10^6\right)$. Вторая строка содержит целых чисел, по модулю не превосходящих $2 * 10^9$.

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

```
fo = open("output.txt", "w")
fi = open('input.txt', 'r')
n = int(fi.readline())
arr = [int(x) for x in fi.readline().split()]
isHeap = True
for i in range(1, n // 2):
    if arr[i - 1] > arr[2 * i - 1] or arr[i - 1] > arr[2 * i]:
        isHeap = False
        break
if isHeap:
```

```
    fo.write("YES")
else:
    fo.write("NO")
```

Бенчмарк к задаче 1

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.062	132915200	10945420	3
1	OK	0.046	10240000	14	2
2	OK	0.046	10227712	14	3
3	OK	0.062	10330112	1092	3
4	OK	0.031	10268672	889	3
5	OK	0.031	10219520	1099	2
6	OK	0.046	10235904	1100	3
7	OK	0.031	10194944	1098	3
8	OK	0.031	10317824	1093	3
9	OK	0.062	10219520	1105	2
10	OK	0.046	10194944	1095	2
11	OK	0.031	10387456	10931	3
12	OK	0.031	10309632	8837	3
13	OK	0.046	10461184	10928	2
14	OK	0.046	10457088	10934	3
15	OK	0.062	10444800	10989	3
16	OK	0.046	10387456	10934	3
17	OK	0.062	10469376	10978	2
18	OK	0.046	10383360	10960	2
19	OK	0.046	11374592	109474	3
20	OK	0.046	11161600	89095	3
21	OK	0.046	11456512	109362	2
22	OK	0.046	11489280	109479	3
23	OK	0.062	11415552	109486	3
24	OK	0.046	11423744	109443	2
25	OK	0.046	11407360	109565	2

26	OK	0.046	11509760	109493	2
27	OK	0.140	23576576	1094387	3
28	OK	0.156	21209088	886879	3
29	OK	0.125	23543808	1094726	2
30	OK	0.140	23539712	1094117	3
31	OK	0.140	23613440	1094308	3
32	OK	0.125	23994368	1094215	3
33	OK	0.125	24354816	1094084	2
34	OK	0.140	23552000	1094403	2
35	OK	1.000	130760704	10944156	3
36	OK	0.953	106229760	8876466	3
37	OK	0.718	132362240	10945179	2
38	OK	1.031	132087808	10945420	3
39	OK	1.062	131031040	10943533	3
40	OK	1.062	132915200	10944594	3
41	OK	0.906	132362240	10944330	2
42	OK	0.765	130809856	10944738	2

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 $n\left(1\le n\le 10^6\right)$ - 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 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 10^9.

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 <vector>
#include <algorithm>
#include <iostream>
using namespace std;
/**/
#include "edx-io.hpp"
#define cout io
#define cin io
/**/
void siftdown(vector<long long> &heap, long long startpos, long long pos) {
     long long newitem = heap[pos];
     while (pos > startpos) {
          long long parentpos = (pos - 1) / 2;
          long long parent = heap[parentpos];
          if (parent > newitem) {
               heap[pos] = parent;
               pos = parentpos;
               continue;
          }
          break;
```

```
    }
    heap[pos] = newitem;
}
void siftup(vector<long long> &heap, long long pos) {
    long long endpos = heap.size();
    long long startpos = pos;
    long long newitem = heap[pos];
    long long childpos = 2*pos + 1;
    while (childpos < endpos) {
        long long rightpos = childpos + 1;
        if ((rightpos < endpos) && !( heap[childpos] < heap[rightpos])) { //TODO
            childpos = rightpos;
            // cout << "A";
        }
        heap[pos] = heap[childpos];
        pos = childpos;
        childpos = 2*pos + 1;
    }
    heap[pos] = newitem;
    siftdown(heap, startpos, pos);
}
void heappush(vector<long long> &heap, long long item){
    heap.push_back(item);
    siftdown(heap, 0, heap.size()-1);
}
long long heappop(vector<long long> &heap){
    long long lastelt = heap.back();
    heap.pop_back();
    if (!heap.empty()) {
        long long returnitem = heap[0];
        heap[0] = lastelt;
        siftup(heap, 0);
        return returnitem;
    }
    return lastelt;
}
int main() {
    long long N;
    cin >> N;
    vector<long long> heap;
    heap.reserve(1000000);
    long long* array = new long long[N+1];
    char action;
    long long a;
    for (long long i = 0; i < N; i++) {
        cin >> action;
        switch (action)
        {
            case 'A':
                cin >> a;
                heappush(heap, a);
                array[i] = a;
                break;
            case 'X':
                if (heap.size() == 0) {
                    cout << '*' << '\n';
                }
                else {
                    long long pop = heappop(heap);
                    cout << pop << '\n';
                }
                break;
            case 'D':
                long long newNum;
```

```
                cin >> a;
                cin >> newNum;
                long long oldNum = array[a-1];
                array[a-1] = newNum;
                long long ind = find(heap.begin(), heap.end(), oldNum) -
heap.begin();
                heap[ind] = newNum;
                siftdown(heap, 0, ind);
                // siftup(heap, 0);
                break;
            }
        }
        return 0;
}
```

Бенчмарк к задаче 2

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.375	24125440	12083657	5694235
1	OK	0.000	3252224	37	12
2	OK	0.031	3268608	6	3
3	OK	0.000	3252224	11	3
4	OK	0.000	3260416	22	4
5	OK	0.015	3268608	19	6
6	OK	0.015	3289088	19	6
7	OK	0.000	3248128	19	6
8	OK	0.000	3235840	48	19
9	OK	0.000	3268608	58	29
10	OK	0.062	3239936	57	28
11	OK	0.015	3248128	48	19
12	OK	0.015	3248128	58	29
13	OK	0.000	3256320	57	28
14	OK	0.000	3244032	828	573
15	OK	0.000	3252224	1037	369
16	OK	0.031	3260416	828	573
17	OK	0.000	3272704	988	404
18	OK	0.000	3244032	1082	300
19	OK	0.000	3256320	1139	240
20	OK	0.015	3264512	930	377

21	OK	0.000	3272704	1190	280
22	OK	0.015	3252224	8184	5678
23	OK	0.031	3264512	10768	3637
24	OK	0.015	3276800	8206	5700
25	OK	0.000	3260416	9903	3928
26	OK	0.000	3244032	10814	3000
27	OK	0.000	3256320	11338	2400
28	OK	0.000	3268608	11138	3582
29	OK	0.000	3239936	10904	3851
30	OK	0.000	3383296	81951	56944
31	OK	0.015	3411968	110901	36274
32	OK	0.000	3338240	81971	56964
33	OK	0.000	3375104	99351	39719
34	OK	0.015	3420160	107882	30000
35	OK	0.031	3424256	113181	24000
36	OK	0.015	3375104	112799	37474
37	OK	0.015	3362816	114106	37576
38	OK	0.031	5046272	819273	569265
39	OK	0.031	5152768	1143615	361526
40	OK	0.031	4648960	819455	569447
41	OK	0.062	4997120	992441	396009
42	OK	0.031	5214208	1079125	300000
43	OK	0.015	5304320	1131016	240000
44	OK	0.015	4775936	1175194	377350
45	OK	0.031	4784128	1174192	378071
46	OK	0.375	21377024	8194244	5694235
47	OK	0.296	22962176	11753433	3632457
48	OK	0.265	17395712	8193883	5693874
49	OK	0.234	21143552	9926125	3963652
50	OK	0.218	23011328	10792079	3000000

51	OK	0.203	24125440	11312176	2400000
52	OK	0.171	19296256	12078250	3794039
53	OK	0.203	19304448	12083657	3795822