Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа № 5

по дисциплине «Алгоритмы и структуры данных»

Openedu – неделя 5

Подготовил:

студент группы P3217

Бураков Илья Алексеевич

Преподаватели:

Романов Алексей Андреевич

Волчек Дмитрий Геннадьевич

Санкт-Петербург, 2019

Куча ли?

Условие

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

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

Решение

openedu/week5/lab5_1.cpp

```
#include "edx-io.hpp"
#include <vector>
#include <string>

using namespace std;

int main() {
	int n;
	io >> n;

	vector<int> arr(n);
	for (auto& v : arr) io >> v;

	bool result = true;
	for (int i = 1; i <= n / 2; i++) {
		if (arr[i - 1] > arr[2*i - 1]) {
			result = false;
			break;
		} else if (2 * i + 1 <= n && arr[i - 1] > arr[(2 * i + 1) - 1]) {
			result = false;
```

```
                    break;
                }
        }

        io << ((result) ? "YES" : "NO");
        return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.046	16760832	10945420	3
1	OK	0.000	2228224	14	2
2	OK	0.015	2228224	14	3
3	OK	0.000	2228224	1092	3
4	OK	0.000	2240512	889	3
5	OK	0.000	2228224	1099	2
6	OK	0.000	2240512	1100	3
7	OK	0.000	2228224	1098	3
8	OK	0.000	2240512	1093	3
9	OK	0.000	2228224	1105	2
10	OK	0.000	2228224	1095	2
11	OK	0.015	2236416	10931	3
12	OK	0.000	2248704	8837	3
13	OK	0.000	2236416	10928	2
14	OK	0.015	2236416	10934	3
15	OK	0.000	2236416	10989	3
16	OK	0.000	2236416	10934	3
17	OK	0.000	2232320	10978	2
18	OK	0.000	2236416	10960	2
19	OK	0.015	2281472	109474	3
20	OK	0.000	2281472	89095	3
21	OK	0.000	2277376	109362	2
22	OK	0.000	2281472	109479	3
23	OK	0.000	2265088	109486	3
24	OK	0.000	2269184	109443	2
25	OK	0.000	2269184	109565	2
26	OK	0.015	2281472	109493	2
27	OK	0.015	3313664	1094387	3
28	OK	0.000	3104768	886879	3
29	OK	0.015	3313664	1094726	2
30	OK	0.000	3309568	1094117	3
31	OK	0.000	3313664	1094308	3
32	OK	0.015	3313664	1094215	3
33	OK	0.000	3309568	1094084	2

34	OK	0.015	3313664	1094403	2
35	OK	0.046	16756736	10944156	3
36	OK	0.046	14692352	8876466	3
37	OK	0.046	16760832	10945179	2
38	OK	0.046	16760832	10945420	3
39	OK	0.046	16756736	10943533	3
40	OK	0.046	16760832	10944594	3
41	OK	0.046	16752640	10944330	2
42	OK	0.046	16760832	10944738	2

Очередь с приоритетами

Условие

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 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 *

Решение

openedu/week5/lab5_2.cpp

```
#include "edx-io.hpp"
#include <vector>
#include <string>

using namespace std;

#define MAX_OPS int(1e6)

auto heap = vector<int>();

// arrays for tracking heap elements position by string numbers on which they were
added
// need to be able to resolve the link in both directions, so there's two arrays
auto strn_to_loc = new int[1e6 + 1];
auto loc_to_strn = new int[1e6];


#define PARENT(i) (((i) + 1) / 2 - 1)
#define LCHILD(i) (2 * ((i) + 1) - 1)
#define RCHILD(i) (LCHILD(i) + 1)
#define PARENT_EXISTS(i) (i != 0)
#define LCHILD_EXISTS(i) (LCHILD(i) < heap.size())
#define RCHILD_EXISTS(i) (RCHILD(i) < heap.size())

void swap(int a1, int a2) {
	int t = heap[a1];
	heap[a1] = heap[a2];
	heap[a2] = t;

	// do position tracking housekeeping as well
	// update strn2loc resolver (swap corresponding elements gathered by
loc2strn)
	int strn1 = loc_to_strn[a1];
	int strn2 = loc_to_strn[a2];

	t = strn_to_loc[strn1];
	strn_to_loc[strn1] = strn_to_loc[strn2];
	strn_to_loc[strn2] = t;
```

```
	// update loc2strn resolver (swap based on just a1 and a2 to be in sync with
heap)
	t = loc_to_strn[a1];
	loc_to_strn[a1] = loc_to_strn[a2];
	loc_to_strn[a2] = t;
}

void heapify(int root) {
	while (LCHILD_EXISTS(root)) {
		int largest_i = root;

		if (heap[LCHILD(root)] < heap[largest_i]) {
			largest_i = LCHILD(root);
		}

		if (RCHILD_EXISTS(root) && heap[RCHILD(root)] < heap[largest_i]) {
			largest_i = RCHILD(root);
		}

		if (largest_i == root) {
			// done, heap is correct
			break;
		}

		// swap and go deeper
		swap(root, largest_i);
		root = largest_i;
	}
}

void bubble_up(int current) {
	// bubble up until new element isn't violating heap condition
	while (PARENT_EXISTS(current) && heap[PARENT(current)] > heap[current]) {
		swap(PARENT(current), current);
		current = PARENT(current);
	}
}

void add(int val) {
	heap.push_back(val);
	bubble_up(heap.size() - 1);
}

int main() {
	int ops_size;
	io >> ops_size;

	heap.reserve(ops_size);

	for (int strn = 1; strn <= ops_size; strn++) {
		char cmd;

		io >> cmd;
		switch (cmd) {
			case 'A': {
				int val;
				io >> val;

				// register added element to location trackers
				strn_to_loc[strn] = heap.size();
				loc_to_strn[heap.size()] = strn;

				add(val);
			} break;
			case 'X': {
				if (empty(heap)) {
					io << "*\n";
```

```
                        } else {
                                io << heap.front() << "\n";
                                swap(0, heap.size() - 1);
                                heap.pop_back();
                                heapify(0);
                        }
                } break;
                case 'D': {
                        int target_strn, newval;
                        io >> target_strn >> newval;

                        int target_index = strn_to_loc[target_strn];
                        heap[target_index] = newval;
                        bubble_up(target_index);
                } break;
            }
        }

        return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.421	22736896	12083657	5694235
1	OK	0.000	2232320	37	12
2	OK	0.000	2248704	6	3
3	OK	0.000	2244608	11	3
4	OK	0.000	2244608	22	4
5	OK	0.000	2228224	19	6
6	OK	0.000	2232320	19	6
7	OK	0.000	2244608	19	6
8	OK	0.000	2244608	48	19
9	OK	0.000	2228224	58	29
10	OK	0.000	2232320	57	28
11	OK	0.000	2244608	48	19
12	OK	0.000	2244608	58	29
13	OK	0.015	2244608	57	28
14	OK	0.000	2232320	828	573
15	OK	0.000	2244608	1037	369
16	OK	0.000	2244608	828	573
17	OK	0.015	2232320	988	404
18	OK	0.000	2248704	1082	300
19	OK	0.000	2224128	1139	240
20	OK	0.000	2228224	930	377
21	OK	0.015	2228224	1190	280
22	OK	0.000	2240512	8184	5678
23	OK	0.000	2240512	10768	3637
24	OK	0.000	2240512	8206	5700

25	OK	0.000	2252800	9903	3928
26	OK	0.015	2240512	10814	3000
27	OK	0.000	2256896	11338	2400
28	OK	0.000	2252800	11138	3582
29	OK	0.000	2240512	10904	3851
30	OK	0.000	2301952	81951	56944
31	OK	0.015	2301952	110901	36274
32	OK	0.015	2289664	81971	56964
33	OK	0.000	2314240	99351	39719
34	OK	0.015	2342912	107882	30000
35	OK	0.000	2338816	113181	24000
36	OK	0.015	2281472	112799	37474
37	OK	0.000	2293760	114106	37576
38	OK	0.031	3248128	819273	569265
39	OK	0.031	3506176	1143615	361526
40	OK	0.031	3244032	819455	569447
41	OK	0.031	3624960	992441	396009
42	OK	0.031	3805184	1079125	300000
43	OK	0.015	3923968	1131016	240000
44	OK	0.031	3407872	1175194	377350
45	OK	0.015	3407872	1174192	378071
46	OK	0.421	16015360	8194244	5694235
47	OK	0.312	18911232	11753433	3632457
48	OK	0.359	16019456	8193883	5693874
49	OK	0.328	19755008	9926125	3963652
50	OK	0.312	21614592	10792079	3000000
51	OK	0.296	22736896	11312176	2400000
52	OK	0.203	17903616	12078250	3794039
53	OK	0.203	17911808	12083657	3795822