ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет программной инженерии и компьютерной техники

Лабораторная работа №5

Неделя пятая

Выполнил:

Жумиков Егор Олегович

Преподаватели:

Романов Алексей Андреевич

Волчек Дмитрий Геннадьевич

Санкт-Петербург, 2019

Оглавление

Задача «Куча ли?»

Условие

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.

Решение

```
try:
    inp = open('input.txt', 'r')
    outp = open('output.txt', 'w')

    input = inp.readline
    print = lambda *args: outp.write(' '.join(map(str, args)) + '\n')
except:
    pass

n = input()
arr = [int(x) for x in input().split()]

ok = True
for i, x in enumerate(arr):
    if i > 0 and arr[(i + 1) // 2 - 1] > x:
        ok = False
        break

print('YES' if ok else 'NO')
```

Результат

Верное решение!
Результаты работы Вашего решения

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.156	131657728	10945420	5
1	OK	0.031	8962048	14	4

Задача «Очередь с приоритетами»

Условие

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 - 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 + 1действительно находится операция A, что этот элемент не был ранее удален операцией X, и что y меньше, чем предыдущее значение этого элемента.

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 «*».

Решение

```
#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;
}
```

Результат

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.453	22736896	12083657	5694235
1	OK	0.015	2240512	37	12