Санкт-Петербургский национальный исследовательский университет

Информационных технологий, механики и оптики

Отчет по решению задач первой недели
По курсу «Алгоритмы и структуры данных»

на Openedu

Выполнил: Сыроватский Павел Валентинович

Группа P3218

Санкт-Петербург

2019

Двоичный поиск

1.0 из 1.0 балла (оценивается)

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дан массив из n элементов, упорядоченный в порядке неубывания, и m запросов: найти первое и последнее вхождение некоторого числа в массив. Требуется ответить на эти запросы.

Формат входного файла

В первой строке входного файла содержится одно число n — размер массива ($1 \le n \le 10^5$). Во второй строке находятся n чисел в порядке неубывания — элементы массива. В третьей строке находится число m — число запросов ($1 \le m \le 10^5$). В следующей строке находятся m чисел — запросы. Элементы массива и запросы являются целыми числами, неотрицательны и не превышают 10^9.

Формат выходного файла

Для каждого запроса выведите в отдельной строке номер (индекс) первого и последнего вхождения этого числа в массив. Если числа в массиве нет, выведите два раза -1.

```cpp
#include <iostream>
#include <string>

using namespace std;


int findMin(const int *arr, int n, int value) {
	int l = -1;
	int r = n;
	while (r > l + 1) {
		int m = (l + r) / 2;
		if (arr[m] < value) {
			l = m;
		}
		else {
			r = m;
		}
	}
	if (r < n and arr[r] == value) {
		return r;
	}
	else {
		return -1;
	}
}


int findMax(int *arr, int n, int value, int leftIndex) {
	int l = leftIndex;
	int r = n;
	while (r > l + 1) {
		int m = (l + r) / 2;
		if (arr[m] == value) {
			l = m;
		}
		else {
			r = m;
		}
	}
	return l;
}

int main() {
	int n;
	input >> n;
	int* arr = new int[n + 1];
	for (int i = 0; i < n; ++i) {
		input >> arr[i];
	}

	int m;
	input >> m;
	for (int i = 0; i < m; ++i) {
		int value;
		input >> value;
		int indexMin = findMin(arr, n, value);
		if (indexMin == -1) {
			output << indexMin << ' ' << indexMin << '\n';
		}
		else {
			int indexMax = findMax(arr, n, value, indexMin);
			output << indexMin + 1 << ' ' << indexMax + 1 << '\n';
		}
	}
	return 0;
```

Результат выполнения первой задачи

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.750	26288128	1978102	1277538
1	OK	0.031	9031680	22	17
2	OK	0.062	8953856	20	38
3	OK	0.031	8998912	41	15
4	OK	0.078	13709312	204081	21587
5	OK	0.062	14024704	412716	21559
6	OK	0.078	14004224	412714	12243
7	OK	0.375	17354752	498728	612555
8	OK	0.390	17723392	1008458	612906
9	OK	0.328	17715200	1008832	341682
10	OK	0.484	18759680	471365	861755
11	OK	0.515	20160512	953290	859761
12	OK	0.406	20246528	953404	548738
13	OK	0.093	12963840	197660	51796
14	OK	0.093	13565952	399789	51761
15	OK	0.078	13836288	399826	29610
16	OK	0.546	19644416	511344	947660
17	OK	0.546	20410368	1034328	951787
18	OK	0.421	21270528	1034511	608920
19	OK	0.187	16142336	384717	274370
20	OK	0.234	16449536	777782	274601
21	OK	0.171	16408576	778270	152655
22	OK	0.156	12550144	219786	228823
23	OK	0.171	12738560	444845	228627
24	OK	0.140	12640256	444580	136297
25	OK	0.109	19619840	452007	84006
26	OK	0.140	19869696	914248	84077

Гирлянда

2.0 из 2.0 баллов (оценивается)

Гирлянда состоит из n лампочек на общем проводе. Один её конец закреплён на заданной высоте A мм ($h1=A$). Благодаря силе тяжести гирлянда прогибается: высота каждой неконцевой лампы на 1 мм меньше, чем средняя высота ближайших соседей ($hi=hi-1+hi+12-1$ для $1<i<N$).
Требуется найти минимальное значение высоты второго конца B ($B=hn$), такое что для любого $\varepsilon>0$при высоте второго конца $B+\varepsilon$ для всех лампочек выполняется условие $hi>0$. Обратите внимание на то, что при данном значении высоты либо ровно одна, либо две соседних лампочки будут иметь нулевую высоту.

Подсказка: для решения этой задачи можно использовать двоичный поиск (метод дихотомии).

Формат входного файла

В первой строке входного файла содержится два числа n и A ($3\le n\le 1000$, n — целое,$10\le A\le 1000$, A — вещественное и дано не более чем с тремя знаками после десятичной точки).

Формат выходного файла

Выведите одно вещественное число B — минимальную высоту второго конца. Ваш ответ будет засчитан, если он будет отличаться от правильного не более, чем на $10-6$.

Реализация программы на C++

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


	int n;
	bool ok;
	input >> n;
	double *h = new double[n];
	input >> h[0];

	double l = 0, r = h[0];

	while (r - l > 0.000000000001) {
		h[1] = (r + l) / 2;
		ok = true;

		for (int i = 2; i < n; ++i) {
			h[i] = 2 * h[i - 1] - h[i - 2] + 2;

			if (h[i] < 0) {
				ok = false;
				break;
			}
		}

		if (ok) {
			r = h[1];
		}
		else {
			l = h[1];
		}
	}

	output << fixed;
	output << setprecision(7);
	output << h[n - 1];


	input.close();
	output.close();

	return 0;
}
```

Результат решения задачи

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.031	2424832	14	14
1	OK	0.000	2400256	9	9
2	OK	0.000	2400256	12	14
3	OK	0.000	2404352	9	9
4	OK	0.000	2404352	11	10
5	OK	0.000	2387968	9	9
6	OK	0.000	2396160	9	9
7	OK	0.000	2408448	14	14
8	OK	0.000	2408448	12	14
9	OK	0.000	2400256	11	14
10	OK	0.000	2404352	13	14
11	OK	0.015	2404352	10	10
12	OK	0.000	2408448	13	14
13	OK	0.015	2404352	10	9
14	OK	0.000	2404352	10	9
15	OK	0.000	2408448	12	14
16	OK	0.000	2400256	9	9
17	OK	0.015	2424832	12	14
18	OK	0.000	2396160	12	14

Высотой дерева называется максимальное число вершин дерева в цепочке, начинающейся в корне дерева, заканчивающейся в одном из его листьев, и не содержащей никакую вершину дважды.

Так, высота дерева, состоящего из единственной вершины, равна единице. Высота пустого дерева (да, бывает и такое!) равна нулю. Высота дерева, изображенного на рисунке, равна четырем.

Дано двоичное дерево поиска. В вершинах этого дерева записаны ключи — целые числа, по модулю не превышающие 109. Для каждой вершины дерева V выполняется следующее условие:

- все ключи вершин из левого поддерева меньше ключа вершины V;
- все ключи вершин из правого поддерева больше ключа вершины V.

Найдите высоту данного дерева.

Формат входного файла

Входной файл содержит описание двоичного дерева. В первой строке файла находится число N ($0 \le N \le 2 \cdot 105$) — число вершин в дереве. В последующих N строках файла находятся описания вершин дерева. В $(i+1)$-ой строке файла ($1 \le i \le N$) находится описание i-ой вершины, состоящее из трех чисел Ki, Li, Ri, разделенных пробелами — ключа в i-ой вершине ($|Ki| \le 109$), номера левого ребенка i-ой вершины ($i < Li \le N$ или $Li = 0$, если левого ребенка нет) и номера правого ребенка i-ой вершины ($i < Ri \le N$ или $Ri = 0$, если правого ребенка нет).

Все ключи различны. Гарантируется, что данное дерево является деревом поиска.

Формат выходного файла

Выведите одно целое число — высоту дерева.

Реализация программы на C++

```
#include "edx-io.hpp"
#define input io
#define output io

#endif

struct t_node {
	int left, right;
} *tree;

int depth(int i) {
	int d = 1;

	if (tree[i].left) {
		d = max(depth(tree[i].left - 1) + 1, d);
	}
	if (tree[i].right) {
		d = max(depth(tree[i].right - 1) + 1, d);
	}

	return d;
}


int main() {
	int n, k, l, r;
	input >> n;

	if (n) {
		tree = new t_node[n];

		for (int i = 0; i < n; ++i) {
			input >> k >> tree[i].left >> tree[i].right;
		}

		output << depth(0);
	}
	else {
		output << 0;
	}

	return 0;
}
```

Результат решения задачи

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.031	16986112	3989144	6
1	OK	0.000	2441216	46	1
2	OK	0.015	2441216	3	1
3	OK	0.000	2441216	11	1
4	OK	0.015	2445312	18	1
5	OK	0.015	2445312	103	1
6	OK	0.015	2441216	76	2
7	OK	0.015	2449408	155	2
8	OK	0.000	2441216	163	2
9	OK	0.000	2445312	57	1
10	OK	0.015	2224128	161	1
11	OK	0.015	2240512	2099	1
12	OK	0.000	2224128	1197	3
13	OK	0.015	2224128	2073	3
14	OK	0.015	2240512	2139	3
15	OK	0.015	2224128	686	1
16	OK	0.000	2224128	2128	2
17	OK	0.000	2244608	8777	1
18	OK	0.000	2269184	10426	3
19	OK	0.015	2281472	16336	3
20	OK	0.015	2281472	16835	3
21	OK	0.000	2236416	3520	1
22	OK	0.000	2232320	16969	2
23	OK	0.000	2240512	36534	2

Дано некоторое двоичное дерево поиска. Также даны запросы на удаление из него вершин, имеющих заданные ключи, причем вершины удаляются целиком вместе со своими поддеревьями.

После каждого запроса на удаление выведите число оставшихся вершин в дереве.

В вершинах данного дерева записаны ключи — целые числа, по модулю не превышающие 109. Гарантируется, что данное дерево является двоичным деревом поиска, в частности, для каждой вершины дерева V выполняется следующее условие:

- все ключи вершин из левого поддерева меньше ключа вершины V;
- все ключи вершин из правого поддерева больше ключа вершины V.

Высота дерева не превосходит 25, таким образом, можно считать, что оно сбалансировано.

Формат входного файла

Входной файл содержит описание двоичного дерева и описание запросов на удаление.

В первой строке файла находится число N ($1 \le N \le 2 \cdot 105$) — число вершин в дереве. В последующих N строках файла находятся описания вершин дерева. В $(i+1)$-ой строке файла ($1 \le i \le N$) находится описание i-ой вершины, состоящее из трех чисел Ki, Li, Ri, разделенных пробелами — ключа в i-ой вершине ($|Ki| \le 109$), номера левого ребенка i-ой вершины ($i < Li \le N$ или $Li = 0$, если левого ребенка нет) и номера правого ребенка i-ой вершины ($i < Ri \le N$ или $Ri = 0$, если правого ребенка нет).

Все ключи различны. Гарантируется, что данное дерево является деревом поиска.

В следующей строке находится число M ($1 \le M \le 2 \cdot 105$) — число запросов на удаление. В следующей строке находятся M чисел, разделенных пробелами — ключи, вершины с которыми (вместе с их поддеревьями) необходимо удалить. Все эти числа не превосходят 109 по абсолютному значению. Вершина с таким ключом не обязана существовать в дереве — в этом случае дерево изменять не требуется. Гарантируется, что корень дерева никогда не будет удален.

Формат выходного файла

Выведите M строк. На i-ой строке требуется вывести число вершин, оставшихся в дереве после выполнения i-го запроса на удаление.

Реализация программы на C++

```
typedef struct Node{
    Node* left;
    Node* right;
    Node* parent;
    int value;
} Node;

int getCount(Node* node){
    int depth = 1;
    int depthLeft = 0;
    if (node->left != nullptr) depthLeft = getCount(node->left);
    int depthRight = 0;
    if (node->right != nullptr) depthRight = getCount(node->right);
    return depth + depthLeft + depthRight;
}

int deleteNode(Node *node, int value, bool isRight){
    if (node->value == value){
        if (isRight){
            node->parent->right = nullptr;
        } else {
            node->parent->left = nullptr;
        }
        return getCount(node);
    } else{
        if (value < node->value){
            if (node->left != nullptr) return deleteNode(node->left,value, false);
        } else {
            if (node->right != nullptr) return deleteNode(node->right,value, true);
        }

    }
    return 0;
}

int main() {
    int n;
    cin >> n;
    Node *nodes = new Node[n];
    for (int i = 0; i < n; ++i) {
        nodes[i].right = nullptr;
        nodes[i].left = nullptr;
        nodes[i].parent = nullptr;
    }
       //заполняем узлы
    for (int i = 0; i < n; ++i) {
        int k,l,r;
        input >> k >> l >> r;
        nodes[i].value = k;
        if (l != 0){
            nodes[i].left = &(nodes[l - 1]);
            nodes[l-1].parent = &(nodes[i]);
        }
        if (r != 0){
            nodes[i].right = &(nodes[r - 1]);
            nodes[r-1].parent = &(nodes[i]);
        }
    }
    int n2;
       input >> n2;
    int count = n;
    for (int i = 0; i < n2; ++i) {
        int value;
```

```
            input >> value;
        count -= deleteNode(nodes,value,true);
        output << count << '\n';
    }

    return 0;
}
```

Результат решения задачи

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.093	11038720	6029382	1077960
1	OK	0.000	2441216	58	12
2	OK	0.000	2461696	27	12
3	OK	0.000	2445312	34	15
4	OK	0.000	2453504	211	30
5	OK	0.015	2437120	246	30
6	OK	0.000	2445312	3437	457
7	OK	0.000	2441216	3363	483
8	OK	0.031	2465792	18842	4247
9	OK	0.000	2457600	25683	3739
10	OK	0.015	2473984	69351	14791
11	OK	0.000	2277376	88936	11629
12	OK	0.000	2371584	244892	40297
13	OK	0.000	2383872	255614	37596
14	OK	0.015	3321856	978616	141281
15	OK	0.015	3346432	992647	137802