Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа № 6

по дисциплине «Алгоритмы и структуры данных»

Openedu – неделя 6

Подготовил:

студент группы P3217

Бураков Илья Алексеевич

Преподаватели:

Романов Алексей Андреевич

Волчек Дмитрий Геннадьевич

Санкт-Петербург, 2019

Двоичный поиск

Условие

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.

Пример

input.txt	output.txt
5 1 1 2 2 2 3 1 2 3	1 2 3 5 -1 -1

Решение

openedu/week6/lab6_1.cpp

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

using namespace std;

int n;
vector<int> vec;

pair<int, int> bsearch(int e) {
	int l = -1,
		r = n - 1;

	while (r != l + 1) {
		int midi = (r + l) / 2;
		if (vec[midi] < e) {
			l = midi;
		} else {
			r = midi;
		}
	}
```

```
	if (vec[r] != e) {
		// +1 will be done later
		return make_pair(-2, -2);
	}

	int leftb = r;
	l = -1,
	r = n;

	while (r != l + 1) {
		int midi = (r + l) / 2;
		if (vec[midi] <= e) {
			l = midi;
		} else {
			r = midi;
		}
	}

	int rightb = l;

	return make_pair(leftb, rightb);
}

int main() {
	io >> n;
	vec = vector<int>(n);
	for (auto &e : vec) io >> e;

	int m;
	io >> m;
	for (int i = 0; i < m; i++) {
		int e;
		io >> e;
		auto result = bsearch(e);
		io << result.first + 1 << " " << result.second + 1 << "\n";
	}

	return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.078	4198400	1978102	1277538
1	OK	0.000	2240512	22	17
2	OK	0.000	2224128	20	38
3	OK	0.000	2244608	41	15
4	OK	0.015	2224128	204081	21587
5	OK	0.015	2392064	412716	21559
6	OK	0.015	2396160	412714	12243
7	OK	0.031	2531328	498728	612555
8	OK	0.031	3043328	1008458	612906
9	OK	0.031	3043328	1008832	341682
10	OK	0.046	2367488	471365	861755
11	OK	0.046	2846720	953290	859761
12	OK	0.046	2842624	953404	548738
13	OK	0.000	2224128	197660	51796

14	OK	0.015	2367488	399789	51761
15	OK	0.015	2367488	399826	29610
16	OK	0.062	2404352	511344	947660
17	OK	0.062	2924544	1034328	951787
18	OK	0.062	2928640	1034511	608920
19	OK	0.015	2428928	384717	274370
20	OK	0.015	2826240	777782	274601
21	OK	0.031	2830336	778270	152655
22	OK	0.015	2224128	219786	228823
23	OK	0.015	2367488	444845	228627
24	OK	0.000	2367488	444580	136297
25	OK	0.015	2617344	452007	84006
26	OK	0.015	3080192	914248	84077
27	OK	0.015	3080192	914384	46178
28	OK	0.015	2723840	534373	224808
29	OK	0.015	3268608	1080911	225002
30	OK	0.015	3264512	1080929	123417
31	OK	0.015	2650112	474858	115440
32	OK	0.015	3137536	960744	115495
33	OK	0.015	3137536	960330	63391
34	OK	0.078	3198976	977910	1277538
35	OK	0.062	4198400	1977816	1277396
36	OK	0.062	4198400	1978102	700050
37	OK	0.062	3186688	966605	1000288
38	OK	0.062	3186688	962679	1131278
39	OK	0.062	3223552	1000016	1200034
40	OK	0.078	3223552	1000016	1198665
41	OK	0.062	3076096	858730	1199466

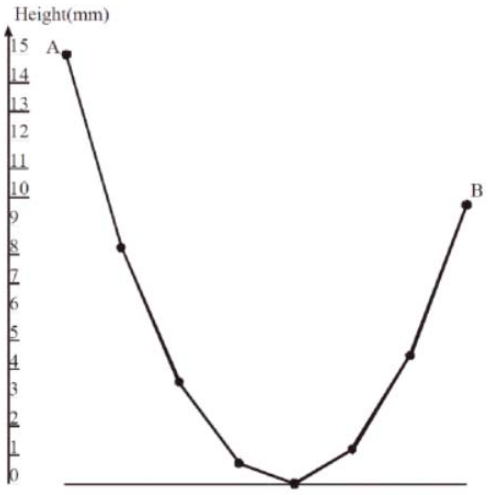
Гирлянда

Условие

Гирлянда состоит из n лампочек на общем проводе. Один её конец закреплён на заданной высоте A мм ($h_1 = A$). Благодаря силе тяжести гирлянда прогибается: высота каждой неконцевой лампы на 1 мм меньше, чем средняя высота ближайших соседей ($h_i = \frac{h_{i-1}+h_{i+1}}{2} - 1$ для $1 < i < N$).

Требуется найти минимальное значение высоты второго конца B ($B = h_n$), такое что для любого $\varepsilon > 0$ при высоте второго конца $B + \varepsilon$ для всех лампочек выполняется условие $h_i > 0$. 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 ($3 \leq n \leq 1000$, n — целое, $10 \leq A \leq 1000$, 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}.

Примеры

input.txt	output.txt
8 15	9.75
692 532.81	446113.34434782615

Решение

openedu/week6/lab6_2.cpp

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

using namespace std;

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

	// lower and upper boundary
	double lowerb = 0, upperb = a;

	// height of the second lamp
	double h2;

	// last successful result
	double last_result;

	while (upperb - lowerb > 1e-10) {
		// guessing h2
		h2 = (upperb + lowerb) / 2;
```

```
		// calculating lamp heights
		double prev = h2;
		double prevprev = a;
		double cur;
		for (int i = 2; i < n; i++) {
			cur = 2 * prev - prevprev + 2;
			if (cur < 0) {
				// one of the lamps is below zero, invalidating cur,
breaking
				cur = NAN;
				break;
			}

			prevprev = prev;
			prev = cur;
		}

		// checking if min height is fine
		if (!isnan(cur)) {
			// still above zero - lowering
			upperb = h2;
			// cur = last lamp's height
			last_result = cur;
		} else {
			// below zero - raising higher
			lowerb = h2;
		}
	}

	io << last_result;
	return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.031	2285568	14	22
1	OK	0.000	2281472	9	18
2	OK	0.015	2281472	12	18
3	OK	0.000	2269184	9	22
4	OK	0.000	2269184	11	22
5	OK	0.015	2265088	9	1
6	OK	0.015	2269184	9	1
7	OK	0.015	2265088	14	18
8	OK	0.015	2260992	12	18
9	OK	0.000	2269184	11	18
10	OK	0.000	2281472	13	18
11	OK	0.000	2269184	10	22
12	OK	0.000	2269184	13	18
13	OK	0.015	2277376	10	22
14	OK	0.000	2269184	10	22
15	OK	0.000	2281472	12	18
16	OK	0.000	2269184	9	17
17	OK	0.000	2265088	12	18

18	OK	0.000	2269184	12	18
19	OK	0.000	2269184	12	18
20	OK	0.000	2269184	11	18
21	OK	0.000	2269184	11	18
22	OK	0.015	2269184	11	17
23	OK	0.015	2265088	11	22
24	OK	0.000	2277376	11	18
25	OK	0.000	2265088	12	18
26	OK	0.000	2269184	12	17
27	OK	0.015	2269184	12	18
28	OK	0.000	2281472	12	18
29	OK	0.000	2269184	12	18
30	OK	0.000	2285568	11	22
31	OK	0.015	2281472	12	18
32	OK	0.000	2265088	12	18
33	OK	0.000	2265088	11	22
34	OK	0.000	2269184	12	17
35	OK	0.000	2269184	12	18
36	OK	0.000	2281472	12	18
37	OK	0.000	2281472	12	18
38	OK	0.000	2269184	11	18
39	OK	0.000	2281472	12	18
40	OK	0.000	2285568	12	17
41	OK	0.000	2281472	12	18
42	OK	0.000	2277376	12	18
43	OK	0.000	2269184	11	18
44	OK	0.015	2269184	12	18
45	OK	0.015	2269184	12	18
46	OK	0.000	2269184	11	22
47	OK	0.000	2269184	12	18
48	OK	0.000	2281472	12	18
49	OK	0.000	2269184	11	17
50	OK	0.000	2281472	11	17
51	OK	0.000	2281472	12	18
52	OK	0.000	2265088	12	17
53	OK	0.000	2269184	11	18
54	OK	0.000	2269184	12	18
55	OK	0.000	2269184	12	18
56	OK	0.000	2281472	12	18
57	OK	0.000	2269184	12	18
58	OK	0.000	2269184	12	18
59	OK	0.000	2265088	12	18

60	OK	0.000	2269184	12	18
61	OK	0.000	2281472	12	18
62	OK	0.015	2281472	10	18
63	OK	0.000	2269184	12	18
64	OK	0.000	2281472	11	18
65	OK	0.000	2265088	12	18
66	OK	0.015	2269184	12	18
67	OK	0.000	2269184	10	18
68	OK	0.031	2281472	12	18
69	OK	0.000	2281472	12	17
70	OK	0.015	2269184	12	18
71	OK	0.015	2269184	11	18
72	OK	0.000	2281472	12	18
73	OK	0.000	2281472	12	17
74	OK	0.000	2281472	12	17
75	OK	0.000	2281472	12	18
76	OK	0.000	2269184	12	18
77	OK	0.000	2269184	12	18
78	OK	0.000	2281472	12	18
79	OK	0.015	2269184	12	17
80	OK	0.015	2281472	11	18
81	OK	0.000	2269184	12	18
82	OK	0.015	2281472	12	18
83	OK	0.000	2265088	11	18
84	OK	0.000	2281472	12	18
85	OK	0.015	2281472	11	17
86	OK	0.015	2269184	12	18
87	OK	0.015	2281472	12	18
88	OK	0.000	2269184	11	18
89	OK	0.015	2269184	12	18
90	OK	0.015	2265088	12	18
91	OK	0.015	2269184	12	18
92	OK	0.031	2281472	12	18
93	OK	0.000	2269184	12	18
94	OK	0.015	2265088	12	18
95	OK	0.000	2265088	12	18
96	OK	0.015	2269184	12	17
97	OK	0.000	2281472	12	18
98	OK	0.000	2265088	12	18
99	OK	0.000	2269184	11	18
100	OK	0.000	2269184	11	22
101	OK	0.015	2269184	12	18

102	OK	0.000	2269184	11	18
103	OK	0.000	2269184	12	18
104	OK	0.000	2265088	11	18
105	OK	0.000	2281472	12	18
106	OK	0.000	2269184	11	18
107	OK	0.000	2265088	12	18
108	OK	0.000	2269184	11	18
109	OK	0.015	2269184	12	17
110	OK	0.000	2269184	12	18
111	OK	0.000	2260992	11	18
112	OK	0.000	2269184	12	18
113	OK	0.015	2269184	12	18
114	OK	0.000	2269184	11	18
115	OK	0.015	2281472	12	18
116	OK	0.015	2269184	12	18
117	OK	0.000	2265088	12	18
118	OK	0.000	2269184	12	18
119	OK	0.000	2281472	11	18
120	OK	0.015	2265088	12	18
121	OK	0.000	2269184	12	18
122	OK	0.000	2265088	12	18
123	OK	0.015	2281472	12	18
124	OK	0.000	2269184	12	18
125	OK	0.000	2269184	12	18
126	OK	0.015	2265088	12	18
127	OK	0.000	2269184	12	18
128	OK	0.000	2269184	12	18
129	OK	0.000	2269184	12	18
130	OK	0.000	2285568	12	18
131	OK	0.000	2265088	12	18
132	OK	0.000	2269184	12	17
133	OK	0.015	2269184	12	18
134	OK	0.000	2269184	12	17
135	OK	0.015	2269184	12	18
136	OK	0.000	2265088	12	18
137	OK	0.000	2269184	12	18
138	OK	0.000	2269184	12	18
139	OK	0.015	2269184	12	18
140	OK	0.000	2269184	12	18
141	OK	0.015	2269184	12	18
142	OK	0.000	2269184	12	17
143	OK	0.015	2281472	12	18

144	OK	0.000	2269184	12	18
145	OK	0.000	2269184	12	18
146	OK	0.000	2277376	12	18
147	OK	0.015	2269184	12	18
148	OK	0.015	2281472	12	18
149	OK	0.015	2269184	12	18
150	OK	0.000	2269184	11	18
151	OK	0.000	2269184	12	18
152	OK	0.015	2281472	12	18
153	OK	0.000	2269184	12	18
154	OK	0.000	2269184	12	18
155	OK	0.000	2269184	12	18
156	OK	0.000	2281472	12	18
157	OK	0.000	2265088	12	18
158	OK	0.015	2269184	12	18
159	OK	0.000	2281472	12	18
160	OK	0.000	2269184	12	18
161	OK	0.000	2265088	12	18
162	OK	0.015	2265088	11	18
163	OK	0.000	2269184	11	18
164	OK	0.015	2269184	12	18
165	OK	0.000	2269184	12	18
166	OK	0.015	2269184	12	17
167	OK	0.000	2269184	12	18
168	OK	0.000	2269184	12	18
169	OK	0.000	2269184	12	18
170	OK	0.000	2269184	12	18
171	OK	0.000	2269184	12	18
172	OK	0.000	2269184	12	18
173	OK	0.000	2269184	12	18
174	OK	0.000	2281472	12	18
175	OK	0.000	2269184	12	18
176	OK	0.015	2269184	12	18
177	OK	0.000	2281472	12	17
178	OK	0.015	2269184	12	18
179	OK	0.000	2281472	12	18
180	OK	0.015	2269184	12	18
181	OK	0.015	2269184	12	18
182	OK	0.000	2265088	12	17
183	OK	0.015	2265088	12	17
184	OK	0.000	2281472	12	18
185	OK	0.015	2269184	12	18

186	OK	0.000	2281472	11	18
187	OK	0.000	2281472	12	18
188	OK	0.000	2269184	9	22
189	OK	0.000	2269184	11	18
190	OK	0.000	2269184	12	18
191	OK	0.015	2269184	12	18
192	OK	0.000	2277376	12	18
193	OK	0.015	2269184	12	18
194	OK	0.000	2285568	12	18
195	OK	0.000	2269184	12	18
196	OK	0.000	2269184	12	18
197	OK	0.015	2265088	12	18
198	OK	0.000	2281472	12	18
199	OK	0.015	2269184	12	18
200	OK	0.000	2269184	11	18
201	OK	0.000	2281472	12	18
202	OK	0.031	2269184	12	18
203	OK	0.031	2269184	12	17
204	OK	0.015	2269184	12	18
205	OK	0.000	2265088	12	18
206	OK	0.000	2269184	12	18
207	OK	0.000	2269184	12	18
208	OK	0.000	2269184	11	18
209	OK	0.031	2269184	12	18
210	OK	0.000	2269184	11	18
211	OK	0.000	2269184	11	18
212	OK	0.000	2281472	11	18
213	OK	0.000	2265088	10	18
214	OK	0.015	2269184	12	18
215	OK	0.000	2269184	12	18
216	OK	0.031	2281472	12	18
217	OK	0.000	2269184	12	18
218	OK	0.000	2269184	11	16
219	OK	0.000	2269184	12	18
220	OK	0.015	2269184	11	18
221	OK	0.015	2277376	12	18
222	OK	0.000	2269184	12	18
223	OK	0.015	2265088	11	18
224	OK	0.015	2269184	11	17
225	OK	0.000	2269184	12	18
226	OK	0.000	2269184	12	18
227	OK	0.000	2269184	12	18

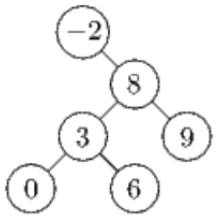
228	OK	0.015	2269184	12	18
229	OK	0.000	2269184	12	18
230	OK	0.000	2269184	12	18
231	OK	0.000	2265088	12	18
232	OK	0.015	2269184	12	18
233	OK	0.015	2269184	12	18
234	OK	0.000	2265088	12	18
235	OK	0.015	2269184	12	18
236	OK	0.000	2269184	11	18
237	OK	0.000	2269184	11	18
238	OK	0.000	2269184	11	18
239	OK	0.000	2265088	12	18
240	OK	0.015	2269184	12	18
241	OK	0.015	2269184	12	18
242	OK	0.000	2269184	12	18
243	OK	0.015	2281472	12	18
244	OK	0.000	2269184	12	18
245	OK	0.000	2269184	12	18
246	OK	0.015	2269184	10	22
247	OK	0.015	2269184	11	18
248	OK	0.000	2265088	12	18
249	OK	0.015	2281472	12	18
250	OK	0.000	2281472	12	18

Высота дерева

Условие

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 10^9. 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 $(0 \le N \le 2 \cdot 10^5)$ — 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 K_i, L_i, R_i, разделенных пробелами — ключа в i-ой вершине $(|K_i| \le 10^9)$, номера левого ребенка i-ой вершины ($i < L_i \le N$ или $L_i = 0$, если левого ребенка нет) и номера правого ребенка i-ой вершины ($i < R_i \le N$ или $R_i = 0$, если правого ребенка нет).

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

Формат выходного файла

Выведите одно целое число — высоту дерева.

Пример

input.txt	output.txt
6 -2 0 2 8 4 3 9 0 0 3 6 5 6 0 0 0 0 0	4

Примечание

Во входном файле задано то же дерево, что и изображено на рисунке.

Решение

openedu/week6/lab6_3.cpp

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

using namespace std;

struct NodeDescription {
	int k;
	int ln;
	int rn;
};

vector<NodeDescription> descriptions;

struct Node {
	int k;
	Node* l;
	Node* r;
```

```
	static Node* createFromDescription(NodeDescription &descr) {
		auto n = new Node();
		n->k = descr.k;
		n->l = descr.ln ? createFromDescription(descriptions[descr.ln - 1]) :
nullptr;
		n->r = descr.rn ? createFromDescription(descriptions[descr.rn - 1]) :
nullptr;
		return n;
	}
};

int find_depth(Node* tree) {
	if (tree == nullptr) {
		return 0;
	}

	return max(find_depth(tree->l), find_depth(tree->r)) + 1;
}

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

	if (n == 0) {
		io << 0;
		return 0;
	}

	descriptions = vector<NodeDescription>(n);
	for (auto &e : descriptions) {
		e = NodeDescription();
		io >> e.k >> e.ln >> e.rn;
	}

	Node* tree = Node::createFromDescription(descriptions[0]);

	io << find_depth(tree);

	return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.062	24301568	3989144	6
1	OK	0.000	2220032	46	1
2	OK	0.000	2224128	3	1
3	OK	0.015	2224128	11	1
4	OK	0.000	2224128	18	1
5	OK	0.015	2220032	103	1
6	OK	0.000	2220032	76	2
7	OK	0.000	2224128	155	2
8	OK	0.015	2236416	163	2
9	OK	0.015	2224128	57	1
10	OK	0.000	2236416	161	1
11	OK	0.000	2240512	2099	1

12	OK	0.000	2232320	1197	3
13	OK	0.000	2244608	2073	3
14	OK	0.000	2228224	2139	3
15	OK	0.000	2232320	686	1
16	OK	0.015	2244608	2128	2
17	OK	0.000	2244608	8777	1
18	OK	0.000	2306048	10426	3
19	OK	0.015	2301952	16336	3
20	OK	0.000	2306048	16835	3
21	OK	0.000	2232320	3520	1
22	OK	0.000	2277376	16969	2
23	OK	0.000	2334720	36534	2
24	OK	0.000	2527232	38820	4
25	OK	0.000	2527232	55707	4
26	OK	0.000	2527232	57235	4
27	OK	0.000	2248704	7784	2
28	OK	0.000	2400256	56607	2
29	OK	0.000	2506752	149518	2
30	OK	0.000	3039232	117171	4
31	OK	0.015	3026944	164193	4
32	OK	0.000	3026944	168789	4
33	OK	0.015	2314240	29385	2
34	OK	0.015	2596864	171161	2
35	OK	0.015	3960832	624213	2
36	OK	0.031	5472256	489475	5
37	OK	0.015	5619712	637029	5
38	OK	0.015	5636096	654072	5
39	OK	0.000	2347008	62037	2
40	OK	0.015	4018176	666913	2
41	OK	0.031	6041600	1259549	2
42	OK	0.031	14553088	1788745	6
43	OK	0.031	15015936	2254723	6
44	OK	0.031	15077376	2313971	6
45	OK	0.015	2531328	152298	2
46	OK	0.015	9404416	2306482	2
47	OK	0.015	10231808	2561292	2
48	OK	0.046	23490560	3177798	6
49	OK	0.062	24203264	3888903	6
50	OK	0.062	24301568	3989144	6
51	OK	0.000	2670592	200543	2
52	OK	0.046	14675968	3953465	2

Удаление поддеревьев

Условие

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

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 10^9. 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 $(1 \le N \le 2 \cdot 10^5)$ — 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 K_i, L_i, R_i, разделенных пробелами — ключа в i-ой вершине $(|K_i| \le 10^9)$, номера левого ребенка i-ой вершины $(i < L_i \le N$ или $L_i = 0$, если левого ребенка нет) и номера правого ребенка i-ой вершины $(i < R_i \le N$ или $R_i = 0$, если правого ребенка нет).

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

В следующей строке находится число M $(1 \le M \le 2 \cdot 10^5)$ — 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 10^9 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.

Пример

input.txt	output.txt
6 -2 0 2 8 4 3 9 0 0 3 6 5 6 0 0 0 0 0 4 6 9 7 8	5 4 4 1

Решение

openedu/week6/lab6_4.cpp

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

using namespace std;

struct NodeDescription {
	int k;
	int ln;
	int rn;
};

vector<NodeDescription> descriptions;

struct Node {
	int k;
	Node* parent;
	Node* l = nullptr;
	Node* r = nullptr;

	int getSize() {
		return (size == -1) ? recalculateSize() : size;
	}

	int recalculateSize() {
		int lsize = l ? l->getSize() : 0;
		int rsize = r ? r->getSize() : 0;

		size = lsize + rsize + 1;
		return size;
	}

	static Node* createFromDescription(NodeDescription &descr) {
		auto n = new Node();
		n->k = descr.k;

		if (descr.ln) {
			n->l = createFromDescription(descriptions[descr.ln - 1]);
			n->l->parent = n;
		}

		if (descr.rn) {
```

```
                    n->r = createFromDescription(descriptions[descr.rn - 1]);
                    n->r->parent = n;
             }

             return n;
      }

private:
      int size = -1;
};

void dropNode(Node* tree, int key) {
      if (key < tree->k) {
             if (tree->l) dropNode(tree->l, key);
      } else if (key > tree->k) {
             if (tree->r) dropNode(tree->r, key);
      } else {
             // screw memory management! nogc!
             if (tree->parent->l == tree) {
                    tree->parent->l = nullptr;
             } else {
                    tree->parent->r = nullptr;
             }

             // but efficient size calculation management is important
             do {
                    tree = tree->parent;
                    tree->recalculateSize();
             } while (tree->parent);
      }
}

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

      descriptions = vector<NodeDescription>(n);
      for (auto &e : descriptions) {
             e = NodeDescription();
             io >> e.k >> e.ln >> e.rn;
      }

      Node* tree = Node::createFromDescription(descriptions[0]);

      int kn;
      io >> kn;
      for (int i = 0; i < kn; i++) {
             int key;
             io >> key;
             dropNode(tree, key);
             io << tree->getSize() << "\n";
      }

      return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.109	19943424	6029382	1077960
1	OK	0.000	2236416	58	12
2	OK	0.000	2224128	27	12
3	OK	0.000	2224128	34	15

4	OK	0.000	2220032	211	30
5	OK	0.000	2220032	246	30
6	OK	0.000	2232320	3437	457
7	OK	0.000	2224128	3363	483
8	OK	0.000	2273280	18842	4247
9	OK	0.015	2285568	25683	3739
10	OK	0.000	2359296	69351	14791
11	OK	0.015	2437120	88936	11629
12	OK	0.000	2650112	244892	40297
13	OK	0.015	2727936	255614	37596
14	OK	0.031	4796416	978616	141281
15	OK	0.031	4890624	992647	137802
16	OK	0.062	8306688	2488583	634135
17	OK	0.062	12435456	3489729	483105
18	OK	0.109	14438400	4639039	1077960
19	OK	0.109	19832832	6007604	931260
20	OK	0.093	19943424	6029382	916969