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 №6 (Week 6 Openedu)

Студент Дунаев Алексей Игоревич

Группа P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Задача 1 Двоичный поиск

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

Дан массив из элементов, упорядоченный в порядке неубывания, и 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 чисел в порядке неубывания — элементы массива. В третьей строке находится число m — число запросов ($1 \le n \le 10^5$). В следующей строке находятся 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

Исходный код к задаче 1

```
import bisect
fi = open('input.txt', 'r')
fo = open('output.txt', 'w')
n = fi.readline()
arr = [int(x) for x in fi.readline().split()]
m = fi.readline()
for x in list(map(int, fi.readline().split())):
    l = bisect.bisect_left(arr, x)
    if l >= len(arr) or arr[l] != x:
        fo.write('-1 -1\n')
    else:
        fo.write(f'{l + 1} {bisect.bisect_right(arr, x)}\n')
```

Бенчмарк к задаче 1

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла

Max		0.609	26976256	1978102	1277538
1	OK	0.031	8982528	22	17
2	OK	0.031	8966144	20	38
3	OK	0.046	8962048	41	15
4	OK	0.062	13701120	204081	21587
5	OK	0.062	13979648	412716	21559
6	OK	0.062	14344192	412714	12243
7	OK	0.296	17616896	498728	612555
8	OK	0.328	17743872	1008458	612906
9	OK	0.234	17661952	1008832	341682
10	OK	0.390	18776064	471365	861755
11	OK	0.406	20193280	953290	859761
12	OK	0.296	20250624	953404	548738
13	OK	0.062	12951552	197660	51796
14	OK	0.109	13578240	399789	51761
15	OK	0.062	13602816	399826	29610
16	OK	0.453	19677184	511344	947660
17	OK	0.437	21966848	1034328	951787
18	OK	0.343	21225472	1034511	608920
19	OK	0.171	15745024	384717	274370
20	OK	0.187	16449536	777782	274601
21	OK	0.140	16474112	778270	152655
22	OK	0.156	12562432	219786	228823
23	OK	0.140	12578816	444845	228627
24	OK	0.109	12632064	444580	136297
25	OK	0.125	19582976	452007	84006
26	OK	0.109	19795968	914248	84077
27	OK	0.109	20045824	914384	46178
28	OK	0.171	20410368	534373	224808
29	OK	0.171	20918272	1080911	225002

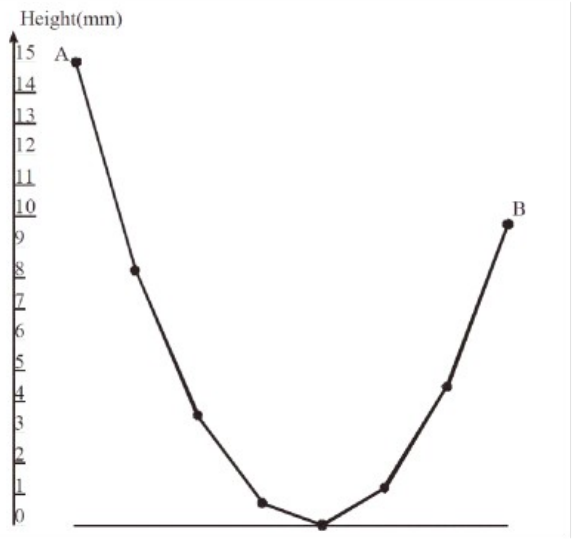
30	OK	0.140	20824064	1080929	123417
31	OK	0.109	19976192	474858	115440
32	OK	0.125	20647936	960744	115495
33	OK	0.125	20606976	960330	63391
34	OK	0.609	25038848	977910	1277538
35	OK	0.609	26812416	1977816	1277396
36	OK	0.515	26976256	1978102	700050
37	OK	0.562	26513408	966605	1000288
38	OK	0.562	25235456	962679	1131278
39	OK	0.562	25812992	1000016	1200034
40	OK	0.578	26624000	1000016	1198665
41	OK	0.562	22773760	858730	1199466

Задача 2. Гирлянда

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

Гирлянда состоит из n лампочек на общем проводе. Один её конец закреплён на заданной высоте A мм (h_i= 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.



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 $10^{|-6|}$.

Примеры

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

Исходный код к задаче 2

```
fi = open('input.txt', 'r')
fo = open('output.txt', 'w')
[n, a] = fi.readline().split()
n = int(n)
a = float(a)
l = 0
r = a
h = [0, 0, 0]
while abs(r - l) > 0.000000000001:
    m = 1001
    h[0] = a
    h[1] = (r + l) / 2
    # h[2] = 2*h[1]-h[0]+1
    for x in range(n - 2):
        h[(x + 2) % 3] = 2 * h[(x + 1) % 3] - h[x % 3] + 2
        if h[(x + 2) % 3] < m:
            m = h[(x + 2) % 3]
        if m < 0:
```

```
                break
        if m < 0:
            l = (r + l) / 2
        else:
            r = (r + l) / 2
        # print(m)
# print("asd")
h[0] = a
h[1] = (r + l) / 2
for x in range(n - 2):
    h[(x + 2) % 3] = 2 * h[(x + 1) % 3] - h[x % 3] + 2
    # print(h[(x+2)%3])
fo.write(str(h[(n + 2) % 3]))
```

Бенчмарк к задаче 2

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.093	9195520	14	23
1	OK	0.031	9064448	9	16
2	OK	0.046	9142272	12	18
3	OK	0.031	9125888	9	22
4	OK	0.031	9060352	11	22
5	OK	0.031	9035776	9	23
6	OK	0.062	9125888	9	18
7	OK	0.062	9101312	14	9
8	OK	0.078	9105408	12	17
9	OK	0.046	9080832	11	16
10	OK	0.031	9105408	13	18
11	OK	0.031	9097216	10	22
12	OK	0.062	9072640	13	17
13	OK	0.031	9089024	10	22
14	OK	0.031	9117696	10	22
15	OK	0.046	9093120	12	17
16	OK	0.031	9134080	9	17
17	OK	0.046	9084928	12	17
18	OK	0.046	9068544	12	18
19	OK	0.031	9101312	12	17

20	OK	0.062	9093120	11	16
21	OK	0.046	9105408	11	18
22	OK	0.046	9052160	11	18
23	OK	0.031	9093120	11	21
24	OK	0.093	9076736	11	17
25	OK	0.046	9068544	12	17
26	OK	0.031	9003008	12	16
27	OK	0.046	9101312	12	17
28	OK	0.046	9068544	12	17
29	OK	0.046	9093120	12	17
30	OK	0.031	9105408	11	23
31	OK	0.046	9084928	12	17
32	OK	0.031	9072640	12	16
33	OK	0.031	9097216	11	21
34	OK	0.046	9076736	12	17
35	OK	0.062	9101312	12	17
36	OK	0.046	9076736	12	18
37	OK	0.062	9031680	12	18
38	OK	0.031	9158656	11	17
39	OK	0.062	9043968	12	17
40	OK	0.046	9113600	12	16
41	OK	0.046	9105408	12	17
42	OK	0.031	9089024	12	18
43	OK	0.031	9084928	11	18
44	OK	0.046	9064448	12	17
45	OK	0.046	9072640	12	18
46	OK	0.031	9109504	11	23
47	OK	0.046	9027584	12	18
48	OK	0.062	9093120	12	16
49	OK	0.093	9146368	11	18

50	OK	0.078	9080832	11	17
51	OK	0.046	9043968	12	17
52	OK	0.046	9056256	12	17
53	OK	0.031	9060352	11	18
54	OK	0.046	9072640	12	17
55	OK	0.046	9060352	12	17
56	OK	0.046	9109504	12	17
57	OK	0.046	9080832	12	17
58	OK	0.046	9154560	12	17
59	OK	0.046	9060352	12	17
60	OK	0.046	9080832	12	18
61	OK	0.046	9097216	12	18
62	OK	0.031	9052160	10	18
63	OK	0.031	9129984	12	17
64	OK	0.031	9125888	11	17
65	OK	0.046	9068544	12	17
66	OK	0.031	9060352	12	18
67	OK	0.031	9064448	10	18
68	OK	0.046	9084928	12	18
69	OK	0.031	9039872	12	18
70	OK	0.062	9080832	12	17
71	OK	0.046	9146368	11	18
72	OK	0.046	9039872	12	18
73	OK	0.046	9076736	12	18
74	OK	0.046	9072640	12	16
75	OK	0.046	9105408	12	17
76	OK	0.046	9089024	12	18
77	OK	0.078	9060352	12	17
78	OK	0.062	9080832	12	17

79	OK	0.046	9072640	12	17
80	OK	0.031	9052160	11	17
81	OK	0.062	9072640	12	18
82	OK	0.046	9158656	12	17
83	OK	0.031	9072640	11	17
84	OK	0.046	9101312	12	16
85	OK	0.031	9097216	11	8
86	OK	0.062	9068544	12	17
87	OK	0.093	9080832	12	17
88	OK	0.031	9109504	11	18
89	OK	0.046	9068544	12	17
90	OK	0.046	9035776	12	17
91	OK	0.031	9072640	12	17
92	OK	0.046	9072640	12	17
93	OK	0.078	9187328	12	17
94	OK	0.046	9097216	12	17
95	OK	0.046	9048064	12	18
96	OK	0.046	9076736	12	17
97	OK	0.062	9138176	12	17
98	OK	0.046	9097216	12	17
99	OK	0.046	9056256	11	18
100	OK	0.031	9109504	11	23
101	OK	0.062	9080832	12	17
102	OK	0.046	9084928	11	16
103	OK	0.046	9064448	12	18
104	OK	0.031	9158656	11	17
105	OK	0.062	9134080	12	17
106	OK	0.062	9076736	11	17
107	OK	0.046	9064448	12	18

108	OK	0.031	9068544	11	18
109	OK	0.046	9097216	12	18
110	OK	0.046	9072640	12	17
111	OK	0.031	9125888	11	17
112	OK	0.031	9035776	12	18
113	OK	0.046	9080832	12	16
114	OK	0.046	9179136	11	17
115	OK	0.046	9080832	12	18
116	OK	0.046	9121792	12	18
117	OK	0.046	9084928	12	18
118	OK	0.046	9035776	12	17
119	OK	0.046	9089024	11	16
120	OK	0.062	9080832	12	17
121	OK	0.031	9043968	12	18
122	OK	0.046	9080832	12	18
123	OK	0.046	9027584	12	18
124	OK	0.046	9089024	12	18
125	OK	0.046	9089024	12	18
126	OK	0.046	9154560	12	17
127	OK	0.046	9097216	12	17
128	OK	0.046	9052160	12	17
129	OK	0.046	9068544	12	9
130	OK	0.046	9060352	12	17
131	OK	0.062	9097216	12	17
132	OK	0.046	9064448	12	17
133	OK	0.046	9068544	12	18
134	OK	0.031	9084928	12	16
135	OK	0.078	9121792	12	18
136	OK	0.062	9072640	12	18

137	OK	0.078	9138176	12	18
138	OK	0.031	9072640	12	17
139	OK	0.031	9048064	12	17
140	OK	0.062	9101312	12	17
141	OK	0.031	9072640	12	18
142	OK	0.046	9072640	12	17
143	OK	0.046	9072640	12	18
144	OK	0.031	9105408	12	17
145	OK	0.062	9056256	12	17
146	OK	0.031	9068544	12	17
147	OK	0.062	9084928	12	18
148	OK	0.046	9150464	12	17
149	OK	0.078	9150464	12	17
150	OK	0.031	9076736	11	16
151	OK	0.046	9101312	12	17
152	OK	0.046	9097216	12	17
153	OK	0.046	9072640	12	17
154	OK	0.062	9052160	12	17
155	OK	0.031	9093120	12	17
156	OK	0.046	9109504	12	17
157	OK	0.046	9121792	12	18
158	OK	0.062	9072640	12	17
159	OK	0.062	9195520	12	16
160	OK	0.031	9076736	12	17
161	OK	0.046	9015296	12	18
162	OK	0.062	9076736	11	18
163	OK	0.046	9093120	11	17
164	OK	0.031	9064448	12	17
165	OK	0.046	9121792	12	17

166	OK	0.046	9076736	12	18
167	OK	0.046	9043968	12	17
168	OK	0.062	9076736	12	17
169	OK	0.062	9101312	12	17
170	OK	0.062	9125888	12	17
171	OK	0.078	9170944	12	18
172	OK	0.078	9068544	12	17
173	OK	0.046	9117696	12	18
174	OK	0.031	9052160	12	17
175	OK	0.046	9076736	12	17
176	OK	0.062	9080832	12	17
177	OK	0.078	9109504	12	17
178	OK	0.062	9060352	12	17
179	OK	0.046	9113600	12	17
180	OK	0.046	9072640	12	18
181	OK	0.062	9113600	12	17
182	OK	0.046	9134080	12	17
183	OK	0.015	9043968	12	18
184	OK	0.062	9093120	12	17
185	OK	0.031	9068544	12	18
186	OK	0.046	9109504	11	17
187	OK	0.046	9072640	12	18
188	OK	0.031	9093120	9	22
189	OK	0.031	9052160	11	18
190	OK	0.046	9105408	12	18
191	OK	0.062	9117696	12	18
192	OK	0.078	9080832	12	18
193	OK	0.046	9142272	12	18
194	OK	0.046	9093120	12	17

195	OK	0.046	9060352	12	18
196	OK	0.046	9084928	12	18
197	OK	0.062	9052160	12	17
198	OK	0.031	9076736	12	18
199	OK	0.062	9076736	12	17
200	OK	0.078	9076736	11	17
201	OK	0.062	9072640	12	17
202	OK	0.031	9072640	12	16
203	OK	0.078	9076736	12	18
204	OK	0.078	9146368	12	17
205	OK	0.031	9101312	12	18
206	OK	0.062	9048064	12	17
207	OK	0.046	9052160	12	18
208	OK	0.031	9048064	11	17
209	OK	0.062	9080832	12	17
210	OK	0.046	9064448	11	17
211	OK	0.062	9084928	11	17
212	OK	0.046	9089024	11	18
213	OK	0.031	9138176	10	17
214	OK	0.062	9129984	12	17
215	OK	0.062	9089024	12	17
216	OK	0.046	9084928	12	17
217	OK	0.062	9048064	12	17
218	OK	0.031	9064448	11	16
219	OK	0.046	9080832	12	17
220	OK	0.046	9072640	11	17
221	OK	0.046	9072640	12	16
222	OK	0.031	9093120	12	18
223	OK	0.046	9134080	11	18

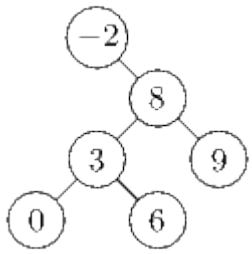
224	OK	0.031	9072640	11	18
225	OK	0.062	9154560	12	18
226	OK	0.046	9101312	12	17
227	OK	0.031	9089024	12	18
228	OK	0.062	9080832	12	17
229	OK	0.062	9064448	12	18
230	OK	0.046	9080832	12	18
231	OK	0.046	9076736	12	17
232	OK	0.078	9072640	12	17
233	OK	0.062	9064448	12	17
234	OK	0.046	9097216	12	18
235	OK	0.046	9080832	12	18
236	OK	0.046	9187328	11	16
237	OK	0.046	9121792	11	17
238	OK	0.031	9101312	11	16
239	OK	0.031	9039872	12	17
240	OK	0.078	9064448	12	18
241	OK	0.078	9072640	12	17
242	OK	0.031	9076736	12	17
243	OK	0.062	9134080	12	16
244	OK	0.062	9076736	12	17
245	OK	0.046	9039872	12	18
246	OK	0.078	9134080	10	23
247	OK	0.031	9162752	11	18
248	OK	0.031	9076736	12	17
249	OK	0.031	9117696	12	18
250	OK	0.062	9113600	12	18

Задача 3 Высота дерева

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

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 выполняется следующее условие:

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

Найдите высоту данного дерева.

Формат входного файла

Входной файл содержит описание двоичного дерева. В первой строке файла находится число $N\left(0 \le N \le 2*10^5\right)$ — 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 $-10^9 \le Ki \le 10^9$, номера левого ребенка i-ой вершины (i < Li < N или Li = 0 , если левого ребенка нет) и номера правого ребенка i-ой вершины (I < Ri < 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.

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

Исходный код к задаче 3

```
#include <iostream>
/**/
#include "edx-io.hpp"
#define cout io
#define cin io
/**/
using namespace std;
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;
     cin >> n;
     if (n) {
          tree = new t_node[n];
          for (int i = 0; i < n; ++i) {
               cin >> k >> tree[i].left >> tree[i].right;
          }
          cout << depth(0);
     } else {
          cout << 0;
     }
     return 0;
}
```

Бенчмарк к задаче 3

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.046	21790720	3989144	6
1	OK	0.015	3420160	46	1
2	OK	0.031	3428352	3	1
3	OK	0.000	3424256	11	1

4	OK	0.000	3424256	18	1
5	OK	0.015	3424256	103	1
6	OK	0.000	3428352	76	2
7	OK	0.015	3436544	155	2
8	OK	0.000	3448832	163	2
9	OK	0.015	3448832	57	1
10	OK	0.000	3436544	161	1
11	OK	0.015	3436544	2099	1
12	OK	0.015	3448832	1197	3
13	OK	0.000	3407872	2073	3
14	OK	0.000	3436544	2139	3
15	OK	0.015	3444736	686	1
16	OK	0.000	3440640	2128	2
17	OK	0.000	3444736	8777	1
18	OK	0.000	3489792	10426	3
19	OK	0.015	3485696	16336	3
20	OK	0.000	3510272	16835	3
21	OK	0.015	3448832	3520	1
22	OK	0.000	3457024	16969	2
23	OK	0.000	3440640	36534	2
24	OK	0.000	3661824	38820	4
25	OK	0.000	3661824	55707	4
26	OK	0.000	3653632	57235	4
27	OK	0.000	3424256	7784	2
28	OK	0.000	3457024	56607	2
29	OK	0.000	3600384	149518	2
30	OK	0.015	4128768	117171	4
31	OK	0.015	4214784	164193	4
32	OK	0.000	4214784	168789	4

33	OK	0.015	3428352	29385	2
34	OK	0.015	3633152	171161	2
35	OK	0.000	4247552	624213	2
36	OK	0.015	6324224	489475	5
37	OK	0.015	6467584	637029	5
38	OK	0.000	6492160	654072	5
39	OK	0.015	3473408	62037	2
40	OK	0.000	4317184	666913	2
41	OK	0.000	5156864	1259549	2
42	OK	0.015	13664256	1788745	6
43	OK	0.031	14127104	2254723	6
44	OK	0.031	14184448	2313971	6
45	OK	0.015	3604480	152298	2
46	OK	0.015	6623232	2306482	2
47	OK	0.031	7008256	2561292	2
48	OK	0.046	20959232	3177798	6
49	OK	0.031	21651456	3888903	6
50	OK	0.031	21790720	3989144	6
51	OK	0.000	3645440	200543	2
52	OK	0.031	8941568	3953465	2

Задача 4 Удаление поддеревьев

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 этого дерева записаны ключи — целые числа, по модулю не превышающие 10^9. Для каждой вершины дерева выполняется следующее условие:

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

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

Найдите высоту данного дерева.

Формат входного файла

Входной файл содержит описание двоичного дерева. В первой строке файла находится число $N\left(0 \le N \le 2*10^5\right)$ — 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 $-10^9 \le Ki \le 10^9$, номера левого ребенка i-ой вершины (i < Li < N или Li = 0 , если левого ребенка нет) и номера правого ребенка i-ой вершины (I < Ri < N или Ri = 0, если правого ребенка нет).

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

В следующей строке находится число M ($1 \le M \le 2*10^5$¿ — 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.

Исходный код к задаче 4

```
#include <iostream>
using namespace std;
/**/
#include "edx-io.hpp"
#define cout io
#define cin io
/**/
struct t_node {
     int left, right, key;
} *tree;
int *parents;
int sz;
int cnt(int i) {
     int d = 1;
     if (tree[i].left) {
          d += cnt(tree[i].left - 1);
     }
     if (tree[i].right) {
          d += cnt(tree[i].right - 1);
     }
     return d;
}
int find(int x) {
     int i = 0;
     while (tree[i].key != x) {
          if (x < tree[i].key) {
               if (tree[i].left) {
                    i = tree[i].left - 1;
               } else {
```

```
                    return -1;
                }
            } else {
                if (tree[i].right) {
                    i = tree[i].right - 1;
                } else {
                    return -1;
                }
            }
        }
        return i;
}
int main() {
        int n, m, x;
        cin >> n;
        tree = new t_node[sz = n];
        parents = new int[n];
        for (int i = 0; i < n; ++i) {
            cin >> tree[i].key >> tree[i].left >> tree[i].right;
            if (tree[i].left) {
                parents[tree[i].left - 1] = i;
            }
            if (tree[i].right) {
                parents[tree[i].right - 1] = i;
            }
        }
        cin >> m;
        for (int i = 0; i < m; ++i) {
            cin >> x;
            x = find(x);
            if (x >= 0) {
                if (x) {
                    if (tree[parents[x]].left - 1 == x) {
                        tree[parents[x]].left = 0;
                    } else {
                        tree[parents[x]].right = 0;
                    }
                }
                sz -= cnt(x);
            }
            cout << sz << "\n";
        }
        return 0;
}
```

Бенчмарк к задаче 4

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.093	12619776	6029382	1077960
1	OK	0.000	3432448	58	12
2	OK	0.000	3448832	27	12
3	OK	0.015	3436544	34	15
4	OK	0.000	3424256	211	30
5	OK	0.000	3420160	246	30

6	OK	0.000	3416064	3437	457
7	OK	0.000	3420160	3363	483
8	OK	0.015	3432448	18842	4247
9	OK	0.000	3436544	25683	3739
10	OK	0.015	3469312	69351	14791
11	OK	0.015	3506176	88936	11629
12	OK	0.000	3739648	244892	40297
13	OK	0.015	3801088	255614	37596
14	OK	0.015	4890624	978616	141281
15	OK	0.015	4915200	992647	137802
16	OK	0.046	6918144	2488583	634135
17	OK	0.031	8724480	3489729	483105
18	OK	0.093	10145792	4639039	1077960
19	OK	0.093	12566528	6007604	931260
20	OK	0.078	12619776	6029382	916969