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 №6

Неделя шестая

Выполнил:

Жумиков Егор Олегович

Преподаватели:

Романов Алексей Андреевич

Волчек Дмитрий Геннадьевич

Санкт-Петербург, 2019

Оглавление

Задача «Двоичный поиск»

Условие

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.

Решение

```
import bisect

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()]
m = input()

for x in list(map(int, input().split())):
    l = bisect.bisect_left(arr, x)
    if l >= len(arr) or arr[l] != x:
        print('-1 -1')
    else:
        print(f'{l + 1} {bisect.bisect_right(arr, x)}')
```

Результат

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.734	27009024	1978102	1277538
1	OK	0.031	8966144	22	17
2	OK	0.046	8978432	20	38
3	OK	0.046	8994816	41	15
4	OK	0.062	13709312	204081	21587
5	OK	0.062	14008320	412716	21559
6	OK	0.078	14274560	412714	12243
7	OK	0.359	17199104	498728	612555

8	OK	0.390	17649664	1008458	612906
9	OK	0.296	17666048	1008832	341682
10	OK	0.515	18722816	471365	861755
11	OK	0.484	20250624	953290	859761
12	OK	0.390	20189184	953404	548738
13	OK	0.078	12935168	197660	51796
14	OK	0.093	13881344	399789	51761
15	OK	0.062	13619200	399826	29610
16	OK	0.546	19636224	511344	947660
17	OK	0.546	21262336	1034328	951787
18	OK	0.421	21270528	1034511	608920
19	OK	0.187	15273984	384717	274370
20	OK	0.234	16465920	777782	274601
21	OK	0.171	16465920	778270	152655
22	OK	0.156	12550144	219786	228823
23	OK	0.171	12652544	444845	228627
24	OK	0.171	12509184	444580	136297
25	OK	0.109	20049920	452007	84006
26	OK	0.125	20033536	914248	84077
27	OK	0.109	20041728	914384	46178
28	OK	0.203	21049344	534373	224808
29	OK	0.203	21098496	1080911	225002
30	OK	0.171	21078016	1080929	123417
31	OK	0.140	20258816	474858	115440
32	OK	0.140	20652032	960744	115495
33	OK	0.140	20660224	960330	63391
34	OK	0.718	25178112	977910	1277538
35	OK	0.734	26353664	1977816	1277396
36	OK	0.609	27009024	1978102	700050
37	OK	0.734	25231360	966605	1000288
38	OK	0.703	26730496	962679	1131278
39	OK	0.718	25829376	1000016	1200034
40	OK	0.703	26136576	1000016	1198665
41	OK	0.671	22781952	858730	1199466

Задача «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 ($h_i = (h_{i-1}+h_{i+1}) / 2 - 1$ 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 $10-6$.

Решение

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <queue>
#include <deque>

using namespace std;

#ifdef LOCAL

#define cin std::cin
#define cout std::cout

#else

#endif


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

#define cin in
#define cout out

	int n;
	bool ok;
	cin >> n;
	double *h = new double[n];
	cin >> 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];
                }
        }

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


        in.close();
        out.close();

        return 0;
}
```

Результат

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.046	2420736	14	14
1	OK	0.015	2400256	9	9
2	OK	0.000	2400256	12	14
3	OK	0.015	2404352	9	9
4	OK	0.000	2404352	11	10
5	OK	0.015	2404352	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	0.015	2400256	11	14
10	OK	0.000	2404352	13	14
11	OK	0.000	2400256	10	10
12	OK	0.015	2408448	13	14
13	OK	0.000	2416640	10	9
14	OK	0.000	2404352	10	9
15	OK	0.015	2400256	12	14

16	OK	0.000	2400256	9	9
17	OK	0.000	2408448	12	14
18	OK	0.015	2404352	12	14
19	OK	0.000	2396160	12	13
20	OK	0.015	2400256	11	14
21	OK	0.000	2416640	11	14
22	OK	0.015	2396160	11	12
23	OK	0.000	2404352	11	9
24	OK	0.000	2416640	11	14
25	OK	0.015	2404352	12	14
26	OK	0.000	2408448	12	14
27	OK	0.000	2404352	12	14
28	OK	0.000	2412544	12	14
29	OK	0.015	2408448	12	14
30	OK	0.015	2404352	11	9
31	OK	0.000	2404352	12	14
32	OK	0.000	2412544	12	13
33	OK	0.015	2404352	11	10
34	OK	0.015	2400256	12	14
35	OK	0.000	2396160	12	13
36	OK	0.015	2396160	12	14
37	OK	0.000	2396160	12	13
38	OK	0.015	2400256	11	11
39	OK	0.031	2408448	12	14
40	OK	0.000	2400256	12	14
41	OK	0.000	2396160	12	13
42	OK	0.015	2396160	12	13
43	OK	0.015	2400256	11	12
44	OK	0.000	2400256	12	14
45	OK	0.031	2408448	12	13
46	OK	0.000	2400256	11	10
47	OK	0.015	2400256	12	14
48	OK	0.015	2408448	12	14
49	OK	0.015	2396160	11	11
50	OK	0.000	2408448	11	14

51	OK	0.000	2404352	12	14
52	OK	0.015	2400256	12	14
53	OK	0.015	2392064	11	11
54	OK	0.000	2400256	12	14
55	OK	0.000	2420736	12	14
56	OK	0.000	2408448	12	14
57	OK	0.015	2400256	12	14
58	OK	0.015	2396160	12	13
59	OK	0.015	2400256	12	14
60	OK	0.000	2404352	12	14
61	OK	0.000	2404352	12	14
62	OK	0.000	2400256	10	12
63	OK	0.015	2408448	12	13
64	OK	0.015	2412544	11	13
65	OK	0.015	2400256	12	14
66	OK	0.000	2400256	12	14
67	OK	0.000	2400256	10	12
68	OK	0.000	2400256	12	14
69	OK	0.000	2396160	12	13
70	OK	0.000	2408448	12	14
71	OK	0.015	2400256	11	12
72	OK	0.000	2396160	12	14
73	OK	0.000	2412544	12	14
74	OK	0.000	2400256	12	14
75	OK	0.046	2420736	12	14
76	OK	0.015	2400256	12	14
77	OK	0.015	2400256	12	14
78	OK	0.000	2408448	12	14
79	OK	0.000	2404352	12	14
80	OK	0.000	2396160	11	11
81	OK	0.000	2400256	12	14
82	OK	0.000	2396160	12	13
83	OK	0.015	2408448	11	14
84	OK	0.000	2412544	12	14
85	OK	0.000	2400256	11	12

86	OK	0.015	2404352	12	14
87	OK	0.015	2408448	12	14
88	OK	0.015	2400256	11	12
89	OK	0.000	2412544	12	14
90	OK	0.015	2408448	12	14
91	OK	0.000	2400256	12	12
92	OK	0.000	2400256	12	14
93	OK	0.000	2412544	12	14
94	OK	0.000	2404352	12	13
95	OK	0.000	2404352	12	14
96	OK	0.015	2400256	12	14
97	OK	0.000	2400256	12	14
98	OK	0.015	2392064	12	13
99	OK	0.000	2396160	11	13
100	OK	0.000	2404352	11	10
101	OK	0.000	2392064	12	13
102	OK	0.000	2404352	11	14
103	OK	0.000	2400256	12	14
104	OK	0.000	2396160	11	10
105	OK	0.000	2400256	12	14
106	OK	0.000	2396160	11	12
107	OK	0.000	2400256	12	14
108	OK	0.000	2400256	11	12
109	OK	0.000	2404352	12	14
110	OK	0.000	2396160	12	13
111	OK	0.015	2400256	11	12
112	OK	0.000	2400256	12	13
113	OK	0.015	2412544	12	14
114	OK	0.000	2396160	11	13
115	OK	0.000	2396160	12	13
116	OK	0.015	2404352	12	14
117	OK	0.015	2404352	12	14
118	OK	0.015	2412544	12	14
119	OK	0.015	2408448	11	14
120	OK	0.015	2400256	12	14

121	OK	0.000	2408448	12	13
122	OK	0.000	2416640	12	14
123	OK	0.000	2400256	12	13
124	OK	0.015	2404352	12	14
125	OK	0.000	2404352	12	14
126	OK	0.015	2408448	12	14
127	OK	0.000	2404352	12	14
128	OK	0.000	2408448	12	14
129	OK	0.000	2404352	12	14
130	OK	0.015	2404352	12	14
131	OK	0.000	2408448	12	14
132	OK	0.015	2396160	12	13
133	OK	0.015	2404352	12	14
134	OK	0.000	2396160	12	13
135	OK	0.015	2404352	12	14
136	OK	0.031	2404352	12	14
137	OK	0.000	2412544	12	14
138	OK	0.015	2396160	12	13
139	OK	0.000	2396160	12	13
140	OK	0.015	2396160	12	13
141	OK	0.015	2408448	12	13
142	OK	0.000	2396160	12	13
143	OK	0.000	2400256	12	14
144	OK	0.046	2408448	12	14
145	OK	0.000	2408448	12	14
146	OK	0.000	2400256	12	12
147	OK	0.015	2400256	12	14
148	OK	0.000	2400256	12	12
149	OK	0.000	2408448	12	14
150	OK	0.000	2400256	11	12
151	OK	0.000	2412544	12	14
152	OK	0.015	2412544	12	14
153	OK	0.000	2400256	12	14
154	OK	0.000	2400256	12	14
155	OK	0.000	2396160	12	13

156	OK	0.000	2400256	12	14
157	OK	0.000	2416640	12	14
158	OK	0.015	2400256	12	14
159	OK	0.015	2420736	12	14
160	OK	0.000	2400256	12	12
161	OK	0.015	2400256	12	14
162	OK	0.000	2396160	11	14
163	OK	0.000	2412544	11	14
164	OK	0.000	2396160	12	13
165	OK	0.000	2408448	12	14
166	OK	0.015	2404352	12	14
167	OK	0.015	2408448	12	13
168	OK	0.015	2400256	12	14
169	OK	0.015	2408448	12	14
170	OK	0.015	2416640	12	14
171	OK	0.000	2400256	12	14
172	OK	0.015	2396160	12	14
173	OK	0.000	2416640	12	14
174	OK	0.000	2396160	12	13
175	OK	0.000	2400256	12	14
176	OK	0.015	2404352	12	13
177	OK	0.015	2400256	12	14
178	OK	0.015	2404352	12	14
179	OK	0.015	2412544	12	14
180	OK	0.000	2396160	12	14
181	OK	0.000	2396160	12	14
182	OK	0.015	2400256	12	14
183	OK	0.015	2404352	12	14
184	OK	0.000	2400256	12	14
185	OK	0.000	2396160	12	13
186	OK	0.015	2404352	11	14
187	OK	0.000	2400256	12	14
188	OK	0.015	2400256	9	9
189	OK	0.015	2396160	11	13
190	OK	0.000	2396160	12	14

191	OK	0.015	2400256	12	14
192	OK	0.000	2416640	12	14
193	OK	0.000	2404352	12	13
194	OK	0.015	2400256	12	14
195	OK	0.000	2396160	12	14
196	OK	0.000	2404352	12	14
197	OK	0.000	2404352	12	13
198	OK	0.000	2404352	12	13
199	OK	0.000	2400256	12	14
200	OK	0.000	2400256	11	14
201	OK	0.000	2408448	12	14
202	OK	0.000	2412544	12	12
203	OK	0.000	2400256	12	13
204	OK	0.015	2400256	12	14
205	OK	0.000	2416640	12	14
206	OK	0.015	2408448	12	14
207	OK	0.000	2396160	12	14
208	OK	0.015	2400256	11	13
209	OK	0.015	2408448	12	14
210	OK	0.015	2400256	11	14
211	OK	0.000	2404352	11	14
212	OK	0.015	2408448	11	14
213	OK	0.015	2400256	10	12
214	OK	0.000	2396160	12	12
215	OK	0.000	2408448	12	14
216	OK	0.015	2408448	12	14
217	OK	0.000	2420736	12	14
218	OK	0.015	2396160	11	12
219	OK	0.000	2396160	12	12
220	OK	0.015	2396160	11	13
221	OK	0.000	2404352	12	14
222	OK	0.000	2400256	12	14
223	OK	0.015	2412544	11	12
224	OK	0.015	2396160	11	13
225	OK	0.015	2400256	12	14

226	OK	0.031	2408448	12	14
227	OK	0.000	2392064	12	13
228	OK	0.000	2408448	12	14
229	OK	0.000	2400256	12	14
230	OK	0.015	2404352	12	14
231	OK	0.015	2404352	12	13
232	OK	0.000	2404352	12	14
233	OK	0.000	2408448	12	14
234	OK	0.000	2404352	12	14
235	OK	0.000	2412544	12	14
236	OK	0.000	2400256	11	14
237	OK	0.015	2400256	11	14
238	OK	0.015	2396160	11	11
239	OK	0.000	2400256	12	12
240	OK	0.000	2404352	12	14
241	OK	0.015	2400256	12	14
242	OK	0.000	2396160	12	13
243	OK	0.000	2416640	12	14
244	OK	0.000	2404352	12	14
245	OK	0.000	2400256	12	14
246	OK	0.000	2416640	10	10
247	OK	0.000	2396160	11	12
248	OK	0.000	2396160	12	13
249	OK	0.000	2396160	12	13
250	OK	0.000	2396160	12	14

Задача «Высота дерева»

Условие

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).

Формат выходного файла

Выведите одно целое число — высоту дерева.

Решение

```
#include <iostream>
#include <string>
#include <queue>
#include <deque>

using namespace std;

#ifdef LOCAL

#define cin std::cin
#define cout std::cout

#else

#include "edx-io.hpp"
#define cin io
#define cout 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;
	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;
}
```

Результат

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.046	16994304	3989144	6
1	OK	0.015	2220032	46	1
2	OK	0.015	2232320	3	1
3	OK	0.000	2236416	11	1
4	OK	0.015	2224128	18	1
5	OK	0.000	2224128	103	1
6	OK	0.015	2224128	76	2
7	OK	0.000	2224128	155	2
8	OK	0.000	2220032	163	2
9	OK	0.015	2224128	57	1
10	OK	0.015	2236416	161	1
11	OK	0.000	2224128	2099	1
12	OK	0.000	2224128	1197	3
13	OK	0.015	2224128	2073	3
14	OK	0.015	2224128	2139	3
15	OK	0.000	2236416	686	1
16	OK	0.015	2224128	2128	2
17	OK	0.000	2232320	8777	1
18	OK	0.000	2281472	10426	3
19	OK	0.000	2281472	16336	3
20	OK	0.000	2265088	16835	3
21	OK	0.000	2224128	3520	1
22	OK	0.000	2244608	16969	2
23	OK	0.000	2236416	36534	2
24	OK	0.000	2392064	38820	4
25	OK	0.000	2408448	55707	4
26	OK	0.000	2396160	57235	4
27	OK	0.000	2232320	7784	2
28	OK	0.000	2265088	56607	2
29	OK	0.000	2289664	149518	2
30	OK	0.000	2740224	117171	4
31	OK	0.015	2740224	164193	4

32	OK	0.000	2723840	168789	4
33	OK	0.015	2240512	29385	2
34	OK	0.000	2297856	171161	2
35	OK	0.000	2699264	624213	2
36	OK	0.015	4202496	489475	5
37	OK	0.000	4354048	637029	5
38	OK	0.015	4366336	654072	5
39	OK	0.000	2248704	62037	2
40	OK	0.000	2748416	666913	2
41	OK	0.015	3596288	1259549	2
42	OK	0.031	10219520	1788745	6
43	OK	0.031	10686464	2254723	6
44	OK	0.031	10743808	2313971	6
45	OK	0.000	2293760	152298	2
46	OK	0.031	5070848	2306482	2
47	OK	0.015	5427200	2561292	2
48	OK	0.046	16183296	3177798	6
49	OK	0.031	16891904	3888903	6
50	OK	0.046	16994304	3989144	6
51	OK	0.000	2314240	200543	2
52	OK	0.015	7368704	3953465	2

Задача «Удаление поддеревьев»

Условие

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.

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).

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.

Решение

```
#include <iostream>
#include <string>
#include <queue>
#include <deque>

using namespace std;

#ifdef LOCAL

#define cin std::cin
#define cout std::cout

#else

#include "edx-io.hpp"
#define cin io
#define cout io

#endif

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

int *keys, *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;
}

#define KEY(a) ((a) + 100000000)


int main() {
        int n, k, 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;
}
```

Результат

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.093	11042816	6029382	1077960
1	OK	0.000	2224128	58	12
2	OK	0.000	2224128	27	12
3	OK	0.015	2220032	34	15
4	OK	0.000	2224128	211	30
5	OK	0.000	2220032	246	30
6	OK	0.015	2224128	3437	457
7	OK	0.015	2220032	3363	483
8	OK	0.000	2244608	18842	4247
9	OK	0.000	2244608	25683	3739
10	OK	0.000	2256896	69351	14791
11	OK	0.000	2277376	88936	11629
12	OK	0.000	2359296	244892	40297
13	OK	0.015	2379776	255614	37596
14	OK	0.000	3321856	978616	141281
15	OK	0.031	3346432	992647	137802
16	OK	0.062	5361664	2488583	634135
17	OK	0.046	7184384	3489729	483105
18	OK	0.093	8560640	4639039	1077960
19	OK	0.078	10997760	6007604	931260
20	OK	0.078	11042816	6029382	916969