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)

Студенка Жетесова Дана группы P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

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.

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

```
#include <fstream>
#include <vector>

using namespace std;

int binsearch_right(vector<int> & a, int x)
{
   int left = 0;
   int right = a.size();

   while (left < right)
   {
      int middle = (left + right) / 2;
      if (x < a[middle])
      {
         right = middle;
      }
      else
      {
         left = middle + 1;
      }
   }
   return a[left - 1] == x ? left : -1;
}
```

```
int binsearch_left(vector<int> & a, int x)
{
   int left = 0;
   int right = a.size();

   while (left < right)
   {
      int middle = (left + right) / 2;
      if (a[middle] < x)
      {
         left = middle + 1;
      }
      else
      {
         right = middle;
      }
   }
   return a[left] == x ? left + 1 : -1;
}

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

   int n, m = 0;
   input >> n;

   vector<int> a(n);

   for (int i = 0; i < n; ++i)
   {
      input >> a[i];
   }

   input >> m;

   for (int i = 0; i < m; ++i)
   {
      int x = 0;
      input >> x;
      int leftmost = binsearch_left(a, x);
      int rightmost = binsearch_right(a, x);
      output << leftmost << " " << rightmost << "\n";
   }
}
```

Бенчмарк к задаче 1

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.375	2392064	1978102	1277538

1	OK	0.015	2371584	22	17
2	OK	0.015	2375680	20	38
3	OK	0.000	2375680	41	15
4	OK	0.046	2367488	204081	21587
5	OK	0.046	2383872	412716	21559
6	OK	0.062	2371584	412714	12243
7	OK	0.156	2379776	498728	612555
8	OK	0.203	2383872	1008458	612906
9	OK	0.187	2387968	1008832	341682
10	OK	0.218	2379776	471365	861755
11	OK	0.234	2383872	953290	859761
12	OK	0.218	2379776	953404	548738
13	OK	0.046	2383872	197660	51796
14	OK	0.046	2363392	399789	51761
15	OK	0.046	2383872	399826	29610
16	OK	0.234	2379776	511344	947660
17	OK	0.265	2383872	1034328	951787
18	OK	0.250	2383872	1034511	608920
19	OK	0.125	2379776	384717	274370
20	OK	0.125	2367488	777782	274601
21	OK	0.109	2383872	778270	152655
22	OK	0.078	2371584	219786	228823
23	OK	0.093	2383872	444845	228627
24	OK	0.078	2379776	444580	136297
25	OK	0.078	2371584	452007	84006
26	OK	0.109	2371584	914248	84077
27	OK	0.109	2379776	914384	46178
28	OK	0.125	2383872	534373	224808
29	OK	0.140	2383872	1080911	225002

30	OK	0.140	2383872	1080929	123417
31	OK	0.109	2371584	474858	115440
32	OK	0.109	2387968	960744	115495
33	OK	0.125	2379776	960330	63391
34	OK	0.343	2379776	977910	1277538
35	OK	0.375	2379776	1977816	1277396
36	OK	0.375	2383872	1978102	700050
37	OK	0.312	2392064	966605	1000288
38	OK	0.312	2383872	962679	1131278
39	OK	0.328	2379776	1000016	1200034
40	OK	0.312	2379776	1000016	1198665
41	OK	0.312	2383872	858730	1199466

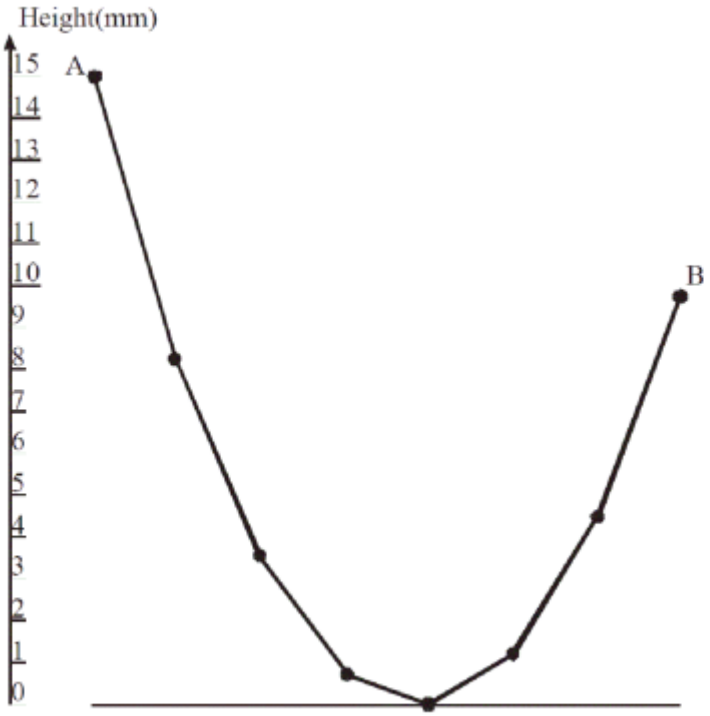
Гирлянда

2.0 из 2.0 баллов (оценивается)

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 ($h1=A$). 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}.

Примеры

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

Исходный код к задаче 2

```
#include <fstream>
#include <iostream>
#include <iomanip>

using namespace std;

double optimal_B(int n, double A)
{
   double hi = A;
   double low = 0;
   double B = 0;
   double EPS = 0.00000000001;

   while (hi - low > EPS)
```

```
    {
        double middle = (hi + low) / 2;

        double prev_lamp = A;
        double curr_lamp = middle;
        double possible_B = 0;
        int at_ground = 0;

        for (int i = 2; i < n; ++i)
        {
            double next_lamp = 2 * curr_lamp + 2 - prev_lamp;
            if (next_lamp == 0)
            {
                ++at_ground;
            }
            if (next_lamp < 0)
            {
                at_ground = 2000;
                break;
            }
            prev_lamp = curr_lamp;
            curr_lamp = next_lamp;
            possible_B = next_lamp;
        }

        if (at_ground <= 1)
        {
            hi = middle;
            B = possible_B;
        }
        else
        {
            low = middle;
        }
    }

    return B;
}


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

    int n = 0;
    double A = 0;

    input >> n >> A;

    output << setprecision(6) << fixed << optimal_B(n, A);

    return 0;
}
```

Бенчмарк к задаче 2

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.031	2412544	14	15
1	OK	0.000	2396160	9	10
2	OK	0.000	2396160	12	15
3	OK	0.000	2404352	9	10
4	OK	0.000	2408448	11	10
5	OK	0.000	2392064	9	10
6	OK	0.015	2396160	9	10
7	OK	0.000	2408448	14	15
8	OK	0.000	2396160	12	15
9	OK	0.000	2396160	11	15
10	OK	0.000	2396160	13	15
11	OK	0.000	2396160	10	10
12	OK	0.000	2408448	13	15
13	OK	0.015	2408448	10	10
14	OK	0.000	2392064	10	10
15	OK	0.015	2392064	12	15
16	OK	0.000	2408448	9	10
17	OK	0.000	2396160	12	15
18	OK	0.015	2396160	12	15
19	OK	0.000	2392064	12	14
20	OK	0.000	2408448	11	15
21	OK	0.000	2396160	11	15
22	OK	0.000	2404352	11	13
23	OK	0.000	2396160	11	10
24	OK	0.015	2392064	11	15
25	OK	0.000	2396160	12	15
26	OK	0.000	2396160	12	15

27	OK	0.000	2392064	12	15
28	OK	0.015	2396160	12	15
29	OK	0.000	2408448	12	15
30	OK	0.015	2396160	11	10
31	OK	0.000	2396160	12	15
32	OK	0.000	2392064	12	14
33	OK	0.015	2392064	11	10
34	OK	0.000	2392064	12	15
35	OK	0.000	2396160	12	14
36	OK	0.000	2396160	12	15
37	OK	0.000	2392064	12	14
38	OK	0.000	2396160	11	12
39	OK	0.000	2408448	12	15
40	OK	0.000	2392064	12	15
41	OK	0.000	2396160	12	14
42	OK	0.000	2408448	12	14
43	OK	0.000	2396160	11	13
44	OK	0.000	2392064	12	15
45	OK	0.015	2396160	12	14
46	OK	0.015	2392064	11	10
47	OK	0.000	2396160	12	15
48	OK	0.000	2392064	12	15
49	OK	0.000	2396160	11	12
50	OK	0.000	2396160	11	15
51	OK	0.000	2396160	12	15
52	OK	0.000	2412544	12	15
53	OK	0.031	2408448	11	12
54	OK	0.000	2396160	12	15
55	OK	0.015	2387968	12	15

56	OK	0.000	2412544	12	15
57	OK	0.015	2392064	12	15
58	OK	0.000	2408448	12	14
59	OK	0.000	2396160	12	15
60	OK	0.000	2404352	12	15
61	OK	0.000	2408448	12	15
62	OK	0.000	2408448	10	13
63	OK	0.000	2392064	12	14
64	OK	0.015	2392064	11	14
65	OK	0.000	2396160	12	15
66	OK	0.000	2392064	12	15
67	OK	0.000	2408448	10	13
68	OK	0.000	2408448	12	15
69	OK	0.015	2396160	12	14
70	OK	0.000	2396160	12	15
71	OK	0.000	2396160	11	13
72	OK	0.015	2396160	12	15
73	OK	0.015	2396160	12	15
74	OK	0.015	2396160	12	15
75	OK	0.000	2408448	12	15
76	OK	0.000	2408448	12	15
77	OK	0.015	2392064	12	15
78	OK	0.000	2408448	12	15
79	OK	0.000	2408448	12	15
80	OK	0.000	2408448	11	12
81	OK	0.015	2396160	12	15
82	OK	0.015	2396160	12	14
83	OK	0.015	2392064	11	15
84	OK	0.015	2396160	12	15

85	OK	0.015	2408448	11	13
86	OK	0.015	2408448	12	15
87	OK	0.015	2396160	12	15
88	OK	0.000	2387968	11	13
89	OK	0.000	2396160	12	15
90	OK	0.000	2396160	12	15
91	OK	0.000	2396160	12	13
92	OK	0.015	2396160	12	15
93	OK	0.015	2396160	12	15
94	OK	0.000	2392064	12	14
95	OK	0.000	2396160	12	15
96	OK	0.000	2396160	12	15
97	OK	0.000	2396160	12	15
98	OK	0.000	2396160	12	14
99	OK	0.015	2392064	11	14
100	OK	0.015	2392064	11	10
101	OK	0.000	2396160	12	14
102	OK	0.015	2408448	11	15
103	OK	0.000	2396160	12	15
104	OK	0.000	2392064	11	11
105	OK	0.015	2392064	12	15
106	OK	0.000	2396160	11	13
107	OK	0.000	2396160	12	15
108	OK	0.000	2396160	11	13
109	OK	0.000	2387968	12	15
110	OK	0.000	2396160	12	14
111	OK	0.000	2396160	11	13
112	OK	0.015	2408448	12	14
113	OK	0.000	2392064	12	15

114	OK	0.015	2396160	11	14
115	OK	0.000	2396160	12	14
116	OK	0.000	2408448	12	15
117	OK	0.015	2396160	12	15
118	OK	0.000	2396160	12	15
119	OK	0.000	2383872	11	15
120	OK	0.000	2396160	12	15
121	OK	0.000	2404352	12	14
122	OK	0.015	2396160	12	15
123	OK	0.000	2396160	12	14
124	OK	0.015	2408448	12	15
125	OK	0.000	2396160	12	15
126	OK	0.000	2396160	12	15
127	OK	0.000	2392064	12	15
128	OK	0.015	2396160	12	15
129	OK	0.000	2392064	12	15
130	OK	0.000	2396160	12	15
131	OK	0.015	2396160	12	15
132	OK	0.015	2396160	12	14
133	OK	0.000	2392064	12	15
134	OK	0.015	2392064	12	14
135	OK	0.015	2396160	12	15
136	OK	0.000	2396160	12	15
137	OK	0.015	2396160	12	15
138	OK	0.000	2408448	12	14
139	OK	0.000	2392064	12	14
140	OK	0.000	2396160	12	14
141	OK	0.000	2392064	12	14
142	OK	0.000	2396160	12	14

143	OK	0.000	2408448	12	15
144	OK	0.015	2396160	12	15
145	OK	0.000	2392064	12	15
146	OK	0.015	2396160	12	13
147	OK	0.000	2396160	12	15
148	OK	0.000	2396160	12	13
149	OK	0.000	2408448	12	15
150	OK	0.000	2396160	11	13
151	OK	0.015	2412544	12	15
152	OK	0.015	2392064	12	15
153	OK	0.000	2396160	12	15
154	OK	0.015	2396160	12	15
155	OK	0.015	2396160	12	14
156	OK	0.015	2396160	12	15
157	OK	0.015	2404352	12	15
158	OK	0.000	2396160	12	15
159	OK	0.015	2408448	12	15
160	OK	0.000	2408448	12	13
161	OK	0.000	2392064	12	15
162	OK	0.000	2387968	11	15
163	OK	0.000	2392064	11	15
164	OK	0.000	2408448	12	14
165	OK	0.015	2396160	12	15
166	OK	0.000	2396160	12	15
167	OK	0.000	2392064	12	14
168	OK	0.000	2396160	12	15
169	OK	0.000	2392064	12	15
170	OK	0.015	2392064	12	15
171	OK	0.000	2396160	12	15

172	OK	0.000	2392064	12	15
173	OK	0.000	2396160	12	15
174	OK	0.000	2408448	12	14
175	OK	0.015	2392064	12	15
176	OK	0.000	2396160	12	14
177	OK	0.000	2396160	12	15
178	OK	0.015	2408448	12	15
179	OK	0.000	2396160	12	15
180	OK	0.000	2408448	12	15
181	OK	0.000	2392064	12	15
182	OK	0.000	2392064	12	15
183	OK	0.000	2408448	12	15
184	OK	0.000	2396160	12	15
185	OK	0.000	2392064	12	14
186	OK	0.000	2392064	11	15
187	OK	0.000	2396160	12	15
188	OK	0.000	2396160	9	10
189	OK	0.000	2396160	11	14
190	OK	0.015	2392064	12	15
191	OK	0.000	2396160	12	15
192	OK	0.000	2412544	12	15
193	OK	0.000	2392064	12	14
194	OK	0.000	2396160	12	15
195	OK	0.015	2396160	12	15
196	OK	0.000	2392064	12	15
197	OK	0.015	2396160	12	14
198	OK	0.000	2392064	12	14
199	OK	0.015	2396160	12	15
200	OK	0.015	2396160	11	15

201	OK	0.000	2396160	12	15
202	OK	0.000	2396160	12	13
203	OK	0.000	2396160	12	14
204	OK	0.000	2392064	12	15
205	OK	0.000	2396160	12	15
206	OK	0.000	2396160	12	15
207	OK	0.015	2392064	12	15
208	OK	0.031	2400256	11	14
209	OK	0.000	2392064	12	15
210	OK	0.000	2392064	11	15
211	OK	0.015	2396160	11	15
212	OK	0.000	2396160	11	15
213	OK	0.000	2408448	10	13
214	OK	0.015	2400256	12	13
215	OK	0.000	2408448	12	15
216	OK	0.015	2408448	12	15
217	OK	0.015	2396160	12	15
218	OK	0.015	2392064	11	13
219	OK	0.000	2387968	12	13
220	OK	0.000	2408448	11	14
221	OK	0.000	2392064	12	15
222	OK	0.000	2396160	12	15
223	OK	0.031	2392064	11	13
224	OK	0.000	2396160	11	14
225	OK	0.015	2404352	12	15
226	OK	0.015	2396160	12	15
227	OK	0.015	2408448	12	14
228	OK	0.000	2396160	12	15
229	OK	0.015	2396160	12	15

230	OK	0.000	2396160	12	15
231	OK	0.000	2408448	12	14
232	OK	0.000	2396160	12	15
233	OK	0.015	2396160	12	15
234	OK	0.000	2396160	12	15
235	OK	0.015	2396160	12	15
236	OK	0.015	2408448	11	15
237	OK	0.000	2392064	11	15
238	OK	0.000	2396160	11	12
239	OK	0.000	2408448	12	13
240	OK	0.000	2396160	12	15
241	OK	0.000	2396160	12	15
242	OK	0.000	2396160	12	14
243	OK	0.015	2396160	12	15
244	OK	0.015	2392064	12	15
245	OK	0.000	2396160	12	15
246	OK	0.000	2392064	10	10
247	OK	0.000	2396160	11	13
248	OK	0.000	2396160	12	14
249	OK	0.015	2396160	12	14
250	OK	0.000	2396160	12	15

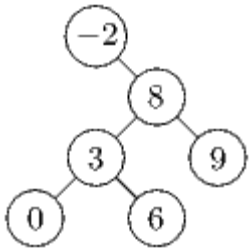
Высота дерева

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

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.

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.

Исходный код к задаче 3

```
#include <iostream>
#include <fstream>
#include <vector>
#include <queue>
#include <memory>

using namespace std;

struct Node
{
   int key;
   size_t children[2];

   Node(int key, size_t left, size_t right) : key(key)
   {
      children[0] = left;
```

```
        children[1] = right;
    }
};

int height(vector<Node*> & tree)
{
    if (tree.empty())
    {
        return 0;
    }

    int max_distance = 0;
    queue<pair<Node*, int>> queue;
    queue.push(make_pair(tree[0], 1));

    while (!queue.empty())
    {
        Node* vertex = queue.front().first;
        int distance = queue.front().second;
        max_distance = max(distance, max_distance);
        for (size_t child : vertex->children)
        {
            if (child != -1)
            {
                queue.push(make_pair(tree[child], distance + 1));
            }
        }
        queue.pop();
    }

    return max_distance;
}

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

    size_t n;
    input >> n;
    vector<Node*> nodes;

    for (size_t i = 0; i < n; i++)
    {
        int key;
        size_t left, right;
        input >> key >> left >> right;
        nodes.push_back(new Node(key, left-1, right-1));
    }

    output << height(nodes);

    return 0;
}
```

Бенчмарк к задаче 1

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.546	11214848	3989144	6
1	OK	0.000	2383872	46	1
2	OK	0.015	2367488	3	1
3	OK	0.015	2387968	11	1
4	OK	0.000	2371584	18	1
5	OK	0.031	2371584	103	1
6	OK	0.015	2371584	76	2
7	OK	0.000	2371584	155	2
8	OK	0.000	2379776	163	2
9	OK	0.000	2383872	57	1
10	OK	0.000	2371584	161	1
11	OK	0.015	2379776	2099	1
12	OK	0.000	2375680	1197	3
13	OK	0.000	2375680	2073	3
14	OK	0.000	2371584	2139	3
15	OK	0.000	2371584	686	1
16	OK	0.015	2383872	2128	2
17	OK	0.015	2416640	8777	1
18	OK	0.000	2420736	10426	3
19	OK	0.000	2420736	16336	3
20	OK	0.000	2420736	16835	3
21	OK	0.015	2383872	3520	1
22	OK	0.015	2428928	16969	2
23	OK	0.015	2482176	36534	2
24	OK	0.015	2531328	38820	4
25	OK	0.000	2531328	55707	4
26	OK	0.015	2531328	57235	4

27	OK	0.015	2412544	7784	2
28	OK	0.000	2543616	56607	2
29	OK	0.031	2695168	149518	2
30	OK	0.015	2756608	117171	4
31	OK	0.031	2777088	164193	4
32	OK	0.031	2764800	168789	4
33	OK	0.000	2486272	29385	2
34	OK	0.031	2768896	171161	2
35	OK	0.093	4464640	624213	2
36	OK	0.078	3551232	489475	5
37	OK	0.093	3555328	637029	5
38	OK	0.093	3543040	654072	5
39	OK	0.015	2506752	62037	2
40	OK	0.093	3723264	666913	2
41	OK	0.171	6725632	1259549	2
42	OK	0.281	6852608	1788745	6
43	OK	0.312	6803456	2254723	6
44	OK	0.328	6848512	2313971	6
45	OK	0.031	2691072	152298	2
46	OK	0.312	7618560	2306482	2
47	OK	0.359	11214848	2561292	2
48	OK	0.484	10149888	3177798	6
49	OK	0.531	10104832	3888903	6
50	OK	0.546	10104832	3989144	6
51	OK	0.046	2826240	200543	2
52	OK	0.531	10924032	3953465	2