Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Кафедра информатики и прикладной математики

Лабораторная работа №6
Дисциплина «Алгоритмы и структуры данных»

Выполнил:
Молодецкий Арсений Алексеевич
Группа P3217

Санкт-Петербург
2019

Задание №1

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.

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

using namespace std;


int binSearch(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 searchMax(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;
    cin >> n;
    int* arr = new int[n+1];
    for (int i = 0; i < n; ++i) {
```

```
        cin >> arr[i];
    }

    int requestCount;
    cin >> requestCount;
    for (int i = 0; i < requestCount; ++i) {
        int value;
        cin >> value;
        int indexMin = binSearch(arr, n, value);
        if (indexMin == -1){
            cout << indexMin << ' ' << indexMin << '\n';
        } else {
            int indexMax = searchMax(arr, n, value, indexMin);
            cout << indexMin + 1 << ' ' << indexMax + 1 << '\n';
        }
    }
    return 0;
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.109	5763072	1978102	1277538
1	OK	0.015	3428352	22	17
2	OK	0.015	3448832	20	38
3	OK	0.046	3424256	41	15
4	OK	0.000	3760128	204081	21587
5	OK	0.015	3964928	412716	21559
6	OK	0.000	3964928	412714	12243
7	OK	0.031	4116480	498728	612555
8	OK	0.046	4591616	1008458	612906
9	OK	0.031	4608000	1008832	341682
10	OK	0.046	3915776	471365	861755
11	OK	0.046	4407296	953290	859761
12	OK	0.046	4395008	953404	548738
13	OK	0.015	3715072	197660	51796
14	OK	0.015	3944448	399789	51761
15	OK	0.000	3915776	399826	29610
16	OK	0.046	3960832	511344	947660
17	OK	0.062	4493312	1034328	951787
18	OK	0.046	4509696	1034511	608920
19	OK	0.015	4009984	384717	274370
20	OK	0.015	4386816	777782	274601
21	OK	0.015	4378624	778270	152655
22	OK	0.046	3702784	219786	228823
23	OK	0.015	3944448	444845	228627
24	OK	0.031	3911680	444580	136297
25	OK	0.000	4190208	452007	84006
26	OK	0.015	4640768	914248	84077
27	OK	0.062	4657152	914384	46178
28	OK	0.015	4276224	534373	224808
29	OK	0.015	4845568	1080911	225002
30	OK	0.015	4841472	1080929	123417
31	OK	0.015	4227072	474858	115440
32	OK	0.015	4702208	960744	115495
33	OK	0.015	4694016	960330	63391
34	OK	0.062	4771840	977910	1277538
35	OK	0.078	5742592	1977816	1277396
36	OK	0.062	5763072	1978102	700050
37	OK	0.109	4734976	966605	1000288
38	OK	0.046	4751360	962679	1131278
39	OK	0.046	4763648	1000016	1200034
40	OK	0.062	4796416	1000016	1198665
41	OK	0.062	4640768	858730	1199466

Задание №2

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).

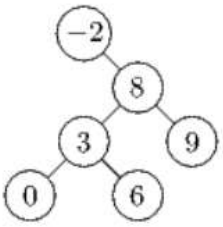
```
#include <string>

using namespace std;

int main() {
    FILE *in;
    in = fopen("input.txt","r");
    int n;
    fscanf(in,"%i",&n);

    auto *garland = new double[n];

    fscanf(in,"%lf",&(garland[0]));
    fclose(in);
    const double precision = 1e-10;

    double lowestEdge = 0;
    double highestEdge = garland[0];

    while (highestEdge - lowestEdge > precision){
        garland[1] = (lowestEdge + highestEdge) / 2;

        int i = 2;
        do{
            double value =  2 * garland[i-1] - garland[i-2] + 2;
            garland[i] = value;
            i++;
        } while (garland[i - 1] >= precision && i < n);

        if (i == n && garland[i-1] >= 0){
            highestEdge = garland[1];
        } else {
            lowestEdge = garland[1];
        }
    }

    FILE *out;
    out = fopen("output.txt","w");
    fprintf(out,"%lf", garland[n-1]);
    fclose(out);
    return 0;
}
```

№ теста	Результат	Время, с	Память,	Размер входного файла	Размер выходного файла
Max		0.031	2036588	34	13
1	ОК	0.008	2281472	9	8
2	ОК	0.008	2301952	12	13
3	ОК	0.008	2289664	9	8
4	ОК	0.008	2281472	11	9
5	ОК	0.015	2281472	9	8
6	ОК	0.008	2281472	9	8
7	ОК	0.008	2289664	14	13
8	ОК	0.008	2289664	12	13
9	ОК	0.008	2289664	11	13
10	ОК	0.008	2297856	13	13
11	ОК	0.015	2289664	10	9
12	ОК	0.008	2289664	13	13
13	ОК	0.008	2281472	10	8
14	ОК	0.015	2277376	10	8
15	ОК	0.008	2289664	12	13
16	ОК	0.015	2281472	9	8
17	ОК	0.008	2289664	12	13
18	ОК	0.015	2289664	10	13
19	ОК	0.015	2281472	12	12
20	ОК	0.015	2289664	11	13
21	ОК	0.015	2285568	11	13
22	ОК	0.008	2293760	11	11
23	ОК	0.008	2281472	11	9
24	ОК	0.008	2289664	11	13
25	ОК	0.008	2289664	12	13
26	ОК	0.008	2289664	12	13
27	ОК	0.008	2289664	12	13
28	ОК	0.008	2306048	12	13
29	ОК	0.008	2289664	12	13
30	ОК	0.008	2281472	11	9
31	ОК	0.008	2285568	12	13
32	ОК	0.008	2277376	12	12
33	ОК	0.008	2281472	11	9
34	ОК	0.015	2289664	12	13
35	ОК	0.008	2281472	12	12
36	ОК	0.008	2289664	10	13
37	ОК	0.008	2281472	12	12
38	ОК	0.008	2281472	11	10
39	ОК	0.015	2289664	12	13
40	ОК	0.015	2289664	10	13
41	ОК	0.015	2306048	12	12
42	ОК	0.015	2281472	12	12
43	ОК	0.008	2281472	11	11
44	ОК	0.015	2301952	10	13
45	ОК	0.008	2281472	12	12
46	ОК	0.015	2281472	11	8
47	ОК	0.008	2289664	10	13
48	ОК	0.015	2285568	12	13
49	ОК	0.008	2277376	11	10
50	ОК	0.008	2301952	11	13
51	ОК	0.008	2293760	12	13
52	ОК	0.008	2297856	12	13
53	ОК	0.015	2281472	11	10
54	ОК	0.008	2301952	12	13
55	ОК	0.008	2289664	12	13
56	ОК	0.008	2285568	12	13
57	ОК	0.008	2285568	12	13
58	ОК	0.008	2281472	12	12
59	ОК	0.008	2289664	12	13
60	ОК	0.008	2285568	12	13
61	ОК	0.015	2281472	12	13
62	ОК	0.008	2293760	10	11
63	ОК	0.008	2281472	12	12
64	ОК	0.008	2277376	11	12
65	ОК	0.008	2289664	12	13
66	ОК	0.008	2289664	12	13
67	ОК	0.008	2277376	10	11
68	ОК	0.015	2289664	12	13
69	ОК	0.008	2281472	12	12
70	ОК	0.008	2301952	12	13
71	ОК	0.008	2281472	11	11
72	ОК	0.008	2277376	12	13
73	ОК	0.015	2285568	12	14
74	ОК	0.008	2289664	12	13
75	ОК	0.008	2301952	12	13
76	ОК	0.008	2297856	12	13
77	ОК	0.008	2289664	12	13
78	ОК	0.008	2289664	12	13
79	ОК	0.008	2285568	12	13
80	ОК	0.031	2281472	11	10
81	ОК	0.008	2289664	12	13
82	ОК	0.015	2281472	12	12
83	ОК	0.015	2289664	11	13
84	ОК	0.008	2297856	12	14
85	ОК	0.008	2281472	11	11
86	ОК	0.008	2301952	12	13
87	ОК	0.008	2306048	12	13
88	ОК	0.008	2293760	11	11
89	ОК	0.008	2289664	12	13
90	ОК	0.008	2289664	12	13
91	ОК	0.015	2293760	12	11
92	ОК	0.008	2289664	12	13
93	ОК	0.008	2285568	12	13
94	ОК	0.015	2281472	12	12
95	ОК	0.015	2301952	12	13
96	ОК	0.008	2289664	12	13
97	ОК	0.008	2289664	12	13
98	ОК	0.008	2281472	12	12
99	ОК	0.008	2281472	11	12
100	ОК	0.008	2293760	11	9
101	ОК	0.015	2281472	12	12
102	ОК	0.008	2277376	11	13
103	ОК	0.008	2289664	12	13
104	ОК	0.008	2281472	11	9
105	ОК	0.008	2289664	12	13
106	ОК	0.015	2293760	11	11
107	ОК	0.015	2289664	12	13
108	ОК	0.031	2293760	11	11
109	ОК	0.008	2289664	12	13
110	ОК	0.008	2277376	12	12
111	ОК	0.008	2277376	11	11
112	ОК	0.008	2281472	12	12
113	ОК	0.008	2285568	12	13
114	ОК	0.015	2281472	11	12
115	ОК	0.008	2281472	12	12
116	ОК	0.015	2297856	12	13
117	ОК	0.008	2281472	12	13
118	ОК	0.015	2285568	12	13
119	ОК	0.015	2289664	11	13
120	ОК	0.015	2289664	12	13
121	ОК	0.008	2281472	12	12
122	ОК	0.015	2281472	12	12
123	ОК	0.008	2281472	12	12
124	ОК	0.008	2277376	12	13
125	ОК	0.008	2281472	12	13
126	ОК	0.015	2306048	12	13
127	ОК	0.015	2281472	12	13
128	ОК	0.008	2289664	12	13
129	ОК	0.008	2293760	12	13
130	ОК	0.015	2289664	12	13
131	ОК	0.008	2289664	12	13
132	ОК	0.008	2281472	12	12
133	ОК	0.008	2289664	12	13
134	ОК	0.008	2277376	12	12
135	ОК	0.008	2289664	12	13
136	ОК	0.008	2289664	12	13
137	ОК	0.008	2289664	12	13
138	ОК	0.008	2281472	12	12
139	ОК	0.008	2281472	12	12
140	ОК	0.008	2281472	12	12
141	ОК	0.015	2281472	12	12
142	ОК	0.008	2277376	12	12
143	ОК	0.008	2289664	12	13
144	ОК	0.008	2301952	12	13
145	ОК	0.015	2289664	12	13
146	ОК	0.015	2281472	12	11
147	ОК	0.008	2281472	12	13
148	ОК	0.015	2281472	12	11
149	ОК	0.008	2289664	12	13
150	ОК	0.008	2281472	11	11
151	ОК	0.008	2285568	12	13
152	ОК	0.015	2301952	12	13
153	ОК	0.015	2285568	12	13
154	ОК	0.008	2301952	12	13
155	ОК	0.015	2281472	12	12
156	ОК	0.015	2306048	12	13
157	ОК	0.008	2285568	12	13
158	ОК	0.008	2289664	12	13
159	ОК	0.008	2301952	12	13
160	ОК	0.008	2281472	12	11
161	ОК	0.015	2306048	12	13
162	ОК	0.008	2289664	11	13
163	ОК	0.008	2285568	11	13
164	ОК	0.008	2281472	12	12
165	ОК	0.015	2289664	12	13
166	ОК	0.008	2293760	12	13
167	ОК	0.015	2277376	12	12
168	ОК	0.008	2289664	12	13
169	ОК	0.008	2289664	12	13
170	ОК	0.008	2285568	12	13
171	ОК	0.008	2301952	12	13
172	ОК	0.015	2301952	12	13
173	ОК	0.008	2281472	12	13
174	ОК	0.008	2281472	10	12
175	ОК	0.015	2289664	12	13
176	ОК	0.008	2281472	12	12
177	ОК	0.008	2289664	12	13
178	ОК	0.015	2297856	12	13
179	ОК	0.008	2301952	12	13
180	ОК	0.008	2301952	12	13
181	ОК	0.008	2301952	12	13
182	ОК	0.008	2285568	12	13
183	ОК	0.008	2289664	12	13
184	ОК	0.008	2289664	12	13
185	ОК	0.008	2281472	12	12
186	ОК	0.008	2289664	11	13
187	ОК	0.008	2285568	12	13
188	ОК	0.008	2289664	9	8
189	ОК	0.015	2281472	11	12
190	ОК	0.008	2289664	12	13
191	ОК	0.008	2285568	12	13
192	ОК	0.015	2285568	12	13
193	ОК	0.015	2281472	12	12
194	ОК	0.015	2301952	12	13
195	ОК	0.008	2289664	12	13
196	ОК	0.008	2301952	12	13
197	ОК	0.008	2281472	12	12
198	ОК	0.008	2281472	12	12
199	ОК	0.008	2289664	12	13
200	ОК	0.008	2285568	11	13
201	ОК	0.008	2289664	12	13
202	ОК	0.008	2281472	12	11
203	ОК	0.008	2281472	10	12
204	ОК	0.008	2289664	12	13
205	ОК	0.015	2289664	12	13
206	ОК	0.008	2289664	12	13
207	ОК	0.008	2289664	12	13
208	ОК	0.015	2281472	11	12
209	ОК	0.008	2285568	12	13
210	ОК	0.015	2289664	11	13
211	ОК	0.008	2289664	11	13
212	ОК	0.008	2289664	11	13
213	ОК	0.031	2277376	10	11
214	ОК	0.008	2281472	12	11
215	ОК	0.015	2301952	12	13
216	ОК	0.008	2301952	12	13
217	ОК	0.008	2301952	12	13
218	ОК	0.008	2281472	11	11
219	ОК	0.015	2281472	12	11
220	ОК	0.008	2277376	11	12
221	ОК	0.008	2285568	12	13
222	ОК	0.008	2285568	12	13
223	ОК	0.008	2281472	11	11
224	ОК	0.008	2281472	11	12
225	ОК	0.015	2289664	12	13
226	ОК	0.008	2301952	12	13
227	ОК	0.015	2289664	12	12
228	ОК	0.008	2289664	12	13
229	ОК	0.015	2285568	12	13
230	ОК	0.015	2297856	12	13
231	ОК	0.015	2281472	12	12
232	ОК	0.008	2289664	12	13
233	ОК	0.008	2289664	12	13
234	ОК	0.015	2285568	12	13
235	ОК	0.008	2289664	13	13
236	ОК	0.008	2289664	11	13
237	ОК	0.008	2285568	11	13
238	ОК	0.008	2293760	11	10
239	ОК	0.008	2289664	12	11
240	ОК	0.008	2285568	12	13
241	ОК	0.008	2285568	12	13
242	ОК	0.008	2281472	12	12
243	ОК	0.008	2301952	12	13
244	ОК	0.008	2281472	12	13
245	ОК	0.008	2297856	12	13
246	ОК	0.015	2281472	10	9
247	ОК	0.008	2281472	11	11
248	ОК	0.008	2281472	12	12
249	ОК	0.008	2277376	12	12
250	ОК	0.015	2289664	12	13

Задание №3

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.

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

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

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;
    }
    for (int i = 0; i < n; ++i) {
        int k,l,r;
        cin >> k >> l >> r;
        nodes[i].value = k;
        if (l != 0) nodes[i].left = &(nodes[l - 1]);
        if (r != 0) nodes[i].right = &(nodes[r - 1]);
    }
    cout << findDepth(&(nodes[0]));
    return 0;
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.046	24965120	3989144	6
1	OK	0.046	3448832	46	1
2	OK	0.000	3440640	3	1
3	OK	0.000	3428352	11	1
4	OK	0.000	3424256	18	1
5	OK	0.015	3452928	103	1
6	OK	0.046	3452928	76	2
7	OK	0.000	3448832	155	2
8	OK	0.000	3457024	163	2
9	OK	0.000	3440640	57	1
10	OK	0.000	3420160	161	1
11	OK	0.015	3452928	2099	1
12	OK	0.000	3416064	1197	3
13	OK	0.015	3436544	2073	3
14	OK	0.015	3444736	2139	3
15	OK	0.000	3444736	686	1
16	OK	0.015	3428352	2128	2
17	OK	0.015	3465216	8777	1
18	OK	0.015	3502080	10426	3
19	OK	0.000	3493888	16336	3
20	OK	0.015	3510272	16835	3
21	OK	0.015	3416064	3520	1
22	OK	0.015	3457024	16969	2
23	OK	0.015	3473408	36534	2
24	OK	0.000	3715072	38820	4
25	OK	0.000	3706880	55707	4
26	OK	0.015	3715072	57235	4
27	OK	0.015	3457024	7784	2
28	OK	0.015	3506176	56607	2
29	OK	0.000	3710976	149518	2
30	OK	0.015	4292608	117171	4
31	OK	0.015	4341760	164193	4
32	OK	0.000	4317184	168789	4
33	OK	0.015	3461120	29385	2
34	OK	0.015	3747840	171161	2
35	OK	0.015	4771840	624213	2
36	OK	0.015	6873088	489475	5
37	OK	0.000	7024640	637029	5
38	OK	0.015	7028736	654072	5
39	OK	0.000	3518464	62037	2
40	OK	0.015	4857856	666913	2
41	OK	0.015	6217728	1259549	2
42	OK	0.015	15589376	1788745	6
43	OK	0.031	16044032	2254723	6
44	OK	0.015	16080896	2313971	6
45	OK	0.015	3751936	152298	2
46	OK	0.015	8523776	2306482	2
47	OK	0.031	9076736	2561292	2
48	OK	0.031	24141824	3177798	6
49	OK	0.031	24875008	3888903	6
50	OK	0.046	24965120	3989144	6
51	OK	0.015	3833856	200543	2
52	OK	0.015	12144640	3953465	2

Задание №4

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.

```
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;
    }
    for (int i = 0; i < n; ++i) {
        int k,l,r;
        cin >> 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;
    cin >> n2;
    int count = n;
    for (int i = 0; i < n2; ++i) {
        int value;
        cin >> value;
        count -= deleteNode(nodes,value,true);
        cout << count << '\n';
    }

    return 0;
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.078	15802368	6029382	1077960
1	OK	0.015	3424256	58	12
2	OK	0.000	3428352	27	12
3	OK	0.000	3424256	34	15
4	OK	0.000	3452928	211	30
5	OK	0.000	3448832	246	30
6	OK	0.000	3465216	3437	457
7	OK	0.015	3448832	3363	483
8	OK	0.015	3469312	18842	4247
9	OK	0.015	3461120	25683	3739
10	OK	0.000	3506176	69351	14791
11	OK	0.000	3559424	88936	11629
12	OK	0.000	3887104	244892	40297
13	OK	0.000	3952640	255614	37596
14	OK	0.015	5431296	978616	141281
15	OK	0.015	5435392	992647	137802
16	OK	0.062	7958528	2488583	634135
17	OK	0.046	10608640	3489729	483105
18	OK	0.078	12226560	4639039	1077960
19	OK	0.078	15720448	6007604	931260
20	OK	0.078	15802368	6029382	916969