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 Неделя №6

OpenEdu

Группа: P3217
Выполнил: Минин Александр

г. Санкт-Петербург
2018г.

Задача №1

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

```
package week6;

import mooc.EdxIO;

public class Week6_1 {
    private static EdxIO edxIO;

    public static void main(String[] args) {
        edxIO = EdxIO.create();

        int n = edxIO.nextInt();
        int[] a = new int[n];
        for (int i = 0; i < n; i++) {
            a[i] = edxIO.nextInt();
        }

        int m = edxIO.nextInt();
        for (int i = 0, num; i < m; i++) {
            num = edxIO.nextInt();
            int leftIndex = binarySearch(num, a);
            if (leftIndex >= 0) {
                edxIO.print((leftIndex + 1) + " ");
                edxIO.println(binarySearchRightIterative(num, a) + 1);
            } else {
                edxIO.println("-1 -1");
            }

        }
        edxIO.close();
    }

    public static int binarySearch(int x, int[] arr) {
        int ind = binarySearchLeftIterative(x, arr);
        if (ind == arr.length || arr[ind] != x) {
            return -ind - 1;
        } else {
            return ind;
        }
    }
```

```
    public static int binarySearchLeftIterative(int x, int[] arr) {
        int l = 0;
        int r = arr.length;
        while (l < r ) {
            int mid = l + (r - l) / 2;
            if (arr[mid] < x) {
                l = mid + 1;
            } else {
                r = mid;
            }
        }
        return r;
    }

    public static int binarySearchRightIterative(int x, int[] arr) {
        int l = 0;
        int r = arr.length;
        while (l < r) {
            int mid = l + (r - l) / 2;
            if (arr[mid] <= x) {
                l = mid + 1;
            } else {
                r = mid;
            }
        }
        return l - 1;
    }
}
```

Результаты

№ теста	Результат	Время, мс	Память	Размер входного файла	Размер выходного файла
Max		375	40480768	1978102	1277538
1	OK	109	21622784	22	17
2	OK	109	21602304	20	38
3	OK	109	21606400	41	15
4	OK	140	23539712	204081	21587
5	OK	125	23867392	412716	21559
6	OK	156	23654400	412714	12243
7	OK	234	30912512	498728	612555
8	OK	265	31186944	1008458	612906
9	OK	156	24383488	1008832	341682
10	OK	281	32681984	471365	861755
11	OK	281	33226752	953290	859761

12	OK	187	24293376	953404	548738
13	OK	156	23871488	197660	51796
14	OK	125	24096768	399789	51761
15	OK	140	23658496	399826	29610
16	OK	234	33624064	511344	947660
17	OK	328	34279424	1034328	951787
18	OK	234	24367104	1034511	608920
19	OK	203	27054080	384717	274370
20	OK	171	27414528	777782	274601
21	OK	156	24264704	778270	152655
22	OK	234	25952256	219786	228823
23	OK	218	26148864	444845	228627
24	OK	140	23629824	444580	136297
25	OK	156	25014272	452007	84006
26	OK	125	25276416	914248	84077
27	OK	125	24342528	914384	46178
28	OK	203	26927104	534373	224808
29	OK	187	27430912	1080911	225002
30	OK	171	24670208	1080929	123417
31	OK	140	25534464	474858	115440
32	OK	187	26001408	960744	115495
33	OK	156	24465408	960330	63391
34	OK	375	38641664	977910	1277538
35	OK	343	38801408	1977816	1277396
36	OK	203	26312704	1978102	700050
37	OK	312	40071168	966605	1000288
38	OK	296	40480768	962679	1131278
39	OK	343	37859328	1000016	1200034
40	OK	328	37023744	1000016	1198665

41	OK	296	40267776	858730	1199466

Задача №2

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.

Решение

```
package week6;

import mooc.EdxIO;

public class Week6_2 {
    private static EdxIO edxIO;

    private static final double EPS = 0.000000000001;

    public static void main(String[] args) {
        edxIO = EdxIO.create();

        int n = edxIO.nextInt();
        double[] h = new double[n];
        h[0] = edxIO.nextDoublePrecise();

        double l = 0;
        double r = h[0];
        boolean above;

        while (r - l > EPS) {
            h[1] = (r + l) / 2;
            above = true;

            for (int i = 2; i < n; i++) {
                h[i] = 2 * h[i - 1] - h[i - 2] + 2;

                if (h[i] < 0) {
                    above = false;
                    break;
                }
            }

            if (above) {
                r = h[1];
            } else {
                l = h[1];
            }
```

```
        }

        edxIO.print(h[n - 1]);

        edxIO.close();
    }

}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		171	21786624	14	23
1	OK	125	21737472	9	17
2	OK	109	21729280	12	17
3	OK	125	21733376	9	21
4	OK	140	21766144	11	22
5	OK	140	21647360	9	3
6	OK	125	21757952	9	3
7	OK	171	21721088	14	9
8	OK	140	21729280	12	17
9	OK	93	21766144	11	17
10	OK	125	21774336	13	18
11	OK	140	21692416	10	23
12	OK	125	21725184	13	17
13	OK	109	21708800	10	22
14	OK	109	21716992	10	21
15	OK	156	21729280	12	17
16	OK	109	21729280	9	17
17	OK	109	21692416	12	17
18	OK	109	21737472	12	18
19	OK	93	21721088	12	17
20	OK	125	21721088	11	18

21	OK	140	21741568	11	18
22	OK	109	21721088	11	18
23	OK	109	21696512	11	22
24	OK	125	21745664	11	18
25	OK	140	21762048	12	17
26	OK	125	21733376	12	18
27	OK	109	21778432	12	17
28	OK	156	21725184	12	17
29	OK	156	21766144	12	17
30	OK	125	21708800	11	21
31	OK	125	21716992	12	17
32	OK	109	21778432	12	18
33	OK	125	21753856	11	23
34	OK	109	21733376	12	17
35	OK	109	21721088	12	17
36	OK	156	21725184	12	17
37	OK	140	21766144	12	18
38	OK	109	21753856	11	16
39	OK	109	21704704	12	17
40	OK	109	21721088	12	16
41	OK	140	21749760	12	17
42	OK	109	21733376	12	18
43	OK	109	21721088	11	18
44	OK	140	21786624	12	17
45	OK	109	21684224	12	17
46	OK	125	21712896	11	23
47	OK	125	21725184	12	18
48	OK	109	21733376	12	17
49	OK	109	21667840	11	18

50	OK	109	21757952	11	17
51	OK	140	21749760	12	17
52	OK	125	21721088	12	17
53	OK	109	21708800	11	18
54	OK	109	21774336	12	18
55	OK	125	21708800	12	17
56	OK	140	21757952	12	17
57	OK	140	21712896	12	17
58	OK	125	21721088	12	16
59	OK	125	21741568	12	17
60	OK	109	21684224	12	18
61	OK	140	21757952	12	18
62	OK	125	21676032	10	18
63	OK	109	21725184	12	17
64	OK	109	21757952	11	17
65	OK	140	21762048	12	17
66	OK	125	21716992	12	18
67	OK	125	21741568	10	16
68	OK	125	21729280	12	17
69	OK	109	21708800	12	17
70	OK	109	21721088	12	17
71	OK	125	21741568	11	18
72	OK	125	21753856	12	18
73	OK	125	21716992	12	17
74	OK	140	21712896	12	17
75	OK	125	21741568	12	17
76	OK	125	21778432	12	18
77	OK	171	21749760	12	17
78	OK	125	21716992	12	17

79	OK	125	21745664	12	18
80	OK	140	21725184	11	15
81	OK	125	21745664	12	18
82	OK	125	21700608	12	17
83	OK	125	21733376	11	17
84	OK	125	21762048	12	17
85	OK	109	21766144	11	17
86	OK	171	21745664	12	17
87	OK	125	21762048	12	17
88	OK	125	21737472	11	17
89	OK	125	21725184	12	17
90	OK	125	21725184	12	17
91	OK	109	21721088	12	17
92	OK	125	21757952	12	17
93	OK	125	21741568	12	17
94	OK	109	21745664	12	17
95	OK	125	21696512	12	18
96	OK	109	21733376	12	17
97	OK	125	21725184	12	16
98	OK	156	21782528	12	17
99	OK	109	21737472	11	17
100	OK	109	21721088	11	22
101	OK	109	21753856	12	18
102	OK	109	21733376	11	18
103	OK	125	21745664	12	18
104	OK	93	21745664	11	18
105	OK	156	21704704	12	18
106	OK	125	21725184	11	17
107	OK	125	21749760	12	17

108	OK	156	21708800	11	18
109	OK	109	21782528	12	18
110	OK	109	21737472	12	18
111	OK	125	21737472	11	18
112	OK	109	21753856	12	18
113	OK	140	21770240	12	18
114	OK	109	21704704	11	17
115	OK	109	21733376	12	17
116	OK	125	21712896	12	18
117	OK	109	21696512	12	18
118	OK	125	21700608	12	17
119	OK	109	21725184	11	16
120	OK	125	21737472	12	16
121	OK	156	21757952	12	16
122	OK	109	21708800	12	18
123	OK	125	21729280	12	18
124	OK	140	21766144	12	18
125	OK	125	21696512	12	18
126	OK	125	21733376	12	17
127	OK	109	21725184	12	17
128	OK	125	21725184	12	17
129	OK	109	21741568	12	17
130	OK	109	21737472	12	18
131	OK	156	21766144	12	16
132	OK	125	21757952	12	17
133	OK	109	21753856	12	18
134	OK	125	21716992	12	17
135	OK	109	21712896	12	17
136	OK	109	21704704	12	18

137	OK	140	21712896	12	18
138	OK	109	21684224	12	18
139	OK	125	21663744	12	18
140	OK	109	21737472	12	17
141	OK	125	21741568	12	18
142	OK	156	21753856	12	17
143	OK	125	21741568	12	18
144	OK	109	21725184	12	17
145	OK	140	21741568	12	17
146	OK	140	21741568	12	18
147	OK	109	21749760	12	18
148	OK	109	21712896	12	17
149	OK	109	21721088	12	15
150	OK	109	21745664	11	17
151	OK	140	21696512	12	18
152	OK	125	21753856	12	18
153	OK	140	21733376	12	16
154	OK	140	21729280	12	16
155	OK	125	21737472	12	17
156	OK	125	21782528	12	17
157	OK	109	21725184	12	16
158	OK	140	21745664	12	17
159	OK	140	21749760	12	17
160	OK	140	21696512	12	17
161	OK	125	21716992	12	18
162	OK	109	21704704	11	17
163	OK	125	21712896	11	16
164	OK	109	21729280	12	17
165	OK	125	21716992	12	17

166	OK	125	21745664	12	17
167	OK	140	21762048	12	18
168	OK	125	21745664	12	17
169	OK	140	21762048	12	17
170	OK	109	21733376	12	17
171	OK	109	21716992	12	18
172	OK	109	21749760	12	17
173	OK	109	21708800	12	18
174	OK	109	21745664	12	17
175	OK	125	21692416	12	18
176	OK	125	21721088	12	17
177	OK	140	21729280	12	17
178	OK	140	21753856	12	17
179	OK	140	21725184	12	18
180	OK	125	21766144	12	18
181	OK	125	21745664	12	17
182	OK	125	21762048	12	17
183	OK	140	21716992	12	17
184	OK	140	21716992	12	17
185	OK	109	21737472	12	17
186	OK	125	21725184	11	17
187	OK	125	21725184	12	18
188	OK	125	21721088	9	21
189	OK	125	21757952	11	17
190	OK	140	21741568	12	18
191	OK	125	21774336	12	18
192	OK	125	21725184	12	18
193	OK	109	21733376	12	18
194	OK	109	21741568	12	18

195	OK	171	21721088	12	18
196	OK	140	21770240	12	18
197	OK	125	21737472	12	17
198	OK	171	21741568	12	18
199	OK	125	21737472	12	17
200	OK	125	21712896	11	17
201	OK	109	21762048	12	17
202	OK	125	21770240	12	17
203	OK	109	21721088	12	18
204	OK	171	21741568	12	17
205	OK	109	21696512	12	18
206	OK	125	21692416	12	17
207	OK	140	21716992	12	18
208	OK	109	21704704	11	18
209	OK	109	21737472	12	17
210	OK	125	21700608	11	17
211	OK	109	21708800	11	16
212	OK	156	21770240	11	16
213	OK	93	21721088	10	17
214	OK	109	21786624	12	17
215	OK	125	21737472	12	17
216	OK	109	21762048	12	18
217	OK	140	21762048	12	17
218	OK	125	21712896	11	16
219	OK	109	21770240	12	17
220	OK	125	21700608	11	16
221	OK	109	21745664	12	18
222	OK	109	21733376	12	17
223	OK	125	21684224	11	17

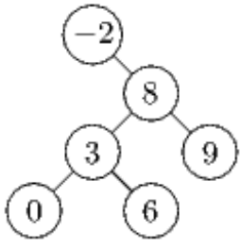
224	OK	109	21696512	11	18
225	OK	125	21725184	12	17
226	OK	140	21753856	12	17
227	OK	171	21737472	12	18
228	OK	140	21745664	12	17
229	OK	140	21753856	12	18
230	OK	109	21721088	12	17
231	OK	109	21696512	12	17
232	OK	140	21749760	12	17
233	OK	156	21745664	12	17
234	OK	140	21737472	12	18
235	OK	140	21753856	12	17
236	OK	109	21716992	11	18
237	OK	140	21757952	11	17
238	OK	125	21708800	11	17
239	OK	140	21712896	12	17
240	OK	125	21700608	12	18
241	OK	125	21733376	12	17
242	OK	125	21729280	12	17
243	OK	156	21778432	12	17
244	OK	171	21741568	12	17
245	OK	125	21745664	12	18
246	OK	109	21749760	10	23
247	OK	125	21696512	11	17
248	OK	140	21745664	12	16
249	OK	171	21716992	12	17
250	OK	109	21729280	12	18

Задача №3

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.

Решение

```
package week6;

import mooc.EdxIO;

public class Week6_3 {

    private static EdxIO edxIO;

    private static Node[] tree;

    public static void main(String[] args) {
        edxIO = EdxIO.create();

        int n = edxIO.nextInt();

        if (n == 0) {
            edxIO.println(0);
            edxIO.close();
            return;
        }

        tree = new Node[n];

        for (int i = 0; i < n; i++) {
            tree[i] = new Node(edxIO.nextInt(), edxIO.nextInt(), edxIO.nextInt());
        }

        edxIO.println(tree[0].getHeight());

        edxIO.close();
```

```
    }

    private static class Node {
        private int key;
        private int left;
        private int right;

        public Node(int key, int left, int right) {
            this.key = key;
            this.left = left;
            this.right = right;
        }

        private int getHeight() {
            int h = 1;

            if (left != 0) {
                h = Integer.max(h, tree[left - 1].getHeight() + 1);
            }
            if (right != 0) {
                h = Integer.max(h, tree[right - 1].getHeight() + 1);
            }

            return h;
        }
    }


}
```

Результаты

№ теста	Результат	Время, мс	Память	Размер входного файла	Размер выходного файла
Max		1406	42151936	3989144	8
1	OK	125	21590016	46	3
2	OK	109	21585920	3	3
3	OK	125	21590016	11	3
4	OK	125	21594112	18	3
5	OK	109	21549056	103	3
6	OK	109	21606400	76	4
7	OK	125	21618688	155	4
8	OK	125	21585920	163	4
9	OK	125	21614592	57	3
10	OK	109	21565440	161	3

11	OK	109	21635072	2099	3
12	OK	109	21594112	1197	5
13	OK	109	21590016	2073	5
14	OK	109	21581824	2139	5
15	OK	125	21606400	686	3
16	OK	109	21618688	2128	4
17	OK	140	21827584	8777	3
18	OK	125	21749760	10426	5
19	OK	125	21884928	16336	5
20	OK	125	21876736	16835	5
21	OK	109	21606400	3520	3
22	OK	109	21889024	16969	4
23	OK	109	22552576	36534	4
24	OK	109	22667264	38820	6
25	OK	140	22728704	55707	6
26	OK	125	22712320	57235	6
27	OK	125	21823488	7784	4
28	OK	109	22757376	56607	4
29	OK	109	23212032	149518	4
30	OK	125	23785472	117171	6
31	OK	125	23965696	164193	6
32	OK	109	23949312	168789	6
33	OK	125	22069248	29385	4
34	OK	125	23212032	171161	4
35	OK	140	24440832	624213	4
36	OK	250	26316800	489475	7
37	OK	250	26619904	637029	7
38	OK	156	27512832	654072	7
39	OK	140	22777856	62037	4

40	OK	156	24592384	666913	4
41	OK	140	26132480	1259549	4
42	OK	859	34250752	1788745	8
43	OK	843	34676736	2254723	8
44	OK	781	35586048	2313971	8
45	OK	203	23228416	152298	4
46	OK	203	29646848	2306482	4
47	OK	171	30396416	2561292	4
48	OK	1390	39895040	3177798	8
49	OK	1406	40562688	3888903	8
50	OK	1328	42151936	3989144	8
51	OK	156	23425024	200543	4
52	OK	203	34713600	3953465	4

Задача №4

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.

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.

Решение

```
package week6;

import mooc.EdxIO;

public class Week6_4 {

    private static EdxIO edxIO;
```

```
private static Node[] tree;
private static int[] parents;
private static int size;

public static void main(String[] args) {
    edxIO = EdxIO.create();

    int n = edxIO.nextInt();

    size = n;
    tree = new Node[n];
    parents = new int[n];

    for (int i = 0; i < n; i++) {
        tree[i] = new Node(edxIO.nextInt(), edxIO.nextInt(), edxIO.nextInt());

        if (tree[i].hasLeft()) {
            parents[tree[i].getLeft() - 1] = i;
        }
        if (tree[i].hasRight()) {
            parents[tree[i].getRight() - 1] = i;
        }
    }

    int m = edxIO.nextInt();
    for (int i = 0; i < m; i++) {
        int x = edxIO.nextInt();
        x = find(x);

        if (x >= 0) {
                if (tree[parents[x]].left - 1 == x) {
                    tree[parents[x]].left = 0;
                }
                else {
                    tree[parents[x]].right = 0;
                }
            size -= tree[x].getSize();
        }
        edxIO.println(size);
    }

    edxIO.close();
}

private static int find(int x) {
    int i = 0;
    while (tree[i].getKey() != x) {
        if (x < tree[i].getKey()) {
            if (tree[i].hasLeft()) {
                i = tree[i].left - 1;
            }
            else {
                return -1;
            }
        }
        else {
            if (tree[i].hasRight()) {
                i = tree[i].right - 1;
            }
```

```
            else {
                return -1;
            }
        }
    }

    return i;
}

private static class Node {
    private int key;
    private int left;
    private int right;

    public Node(int key, int left, int right) {
        this.key = key;
        this.left = left;
        this.right = right;
    }

    public int getKey() {
        return key;
    }

    public void setKey(int key) {
        this.key = key;
    }

    public int getLeft() {
        return left;
    }

    public void setLeft(int left) {
        this.left = left;
    }

    public int getRight() {
        return right;
    }

    public void setRight(int right) {
        this.right = right;
    }

    public boolean hasLeft() {
        return left != 0;
    }

    public boolean hasRight() {
        return right != 0;
    }

    private int getHeight() {
        int h = 1;

        if (hasLeft()) {
            h = Integer.max(h, tree[left - 1].getHeight() + 1);
        }
        if (hasRight()) {
            h = Integer.max(h, tree[right - 1].getHeight() + 1);
```

```
            }

            return h;
        }

        private int getSize() {
            int size = 1;

            if (hasLeft()) {
                size += tree[left - 1].getSize();
            }
            if (hasRight()) {
                size += tree[right - 1].getSize();
            }

            return size;
        }
    }
}
```

Результаты

№ теста	Результат	Время, мс	Память	Размер входного файла	Размер выходного файла
Max		343	38735872	6029382	1077960
1	OK	109	21639168	58	12
2	OK	125	21590016	27	12
3	OK	156	21549056	34	15
4	OK	109	21663744	211	30
5	OK	125	21573632	246	30
6	OK	125	21630976	3437	457
7	OK	140	21647360	3363	483
8	OK	125	21839872	18842	4247
9	OK	156	22077440	25683	3739
10	OK	140	23425024	69351	14791
11	OK	125	23363584	88936	11629
12	OK	156	23678976	244892	40297
13	OK	156	23605248	255614	37596
14	OK	187	25653248	978616	141281
15	OK	218	26103808	992647	137802
16	OK	218	29327360	2488583	634135
17	OK	265	32636928	3489729	483105
18	OK	343	34811904	4639039	1077960
19	OK	328	38731776	6007604	931260
20	OK	296	38735872	6029382	916969