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

ПО ДИСЦИПЛИНЕ «Алгоритмы и структуры данных»

Выполнил: Мещеряков Владимир Алевтинович

Группа: P3218

Преподаватель: Романов Алексей Андреевич

Волчек Дмитрий Геннадьевич

Санкт-Петербург

2019 г.

Week 6

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 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 ($1 \le n \le 105$). Во второй строке находятся n чисел в порядке неубывания — элементы массива. В третьей строке находится число m — число запросов ($1 \le m \le 105$). В следующей строке находятся m чисел — запросы. Элементы массива и запросы являются целыми числами, неотрицательны и не превышают 109.

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

```
#include <iostream>
#include <string>
#include "edx-io.hpp"

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  &&  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;
    io >> n;
    int* arr = new int[n+1];
    for (int i = 0; i < n; ++i) {
        io >> arr[i];
    }

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

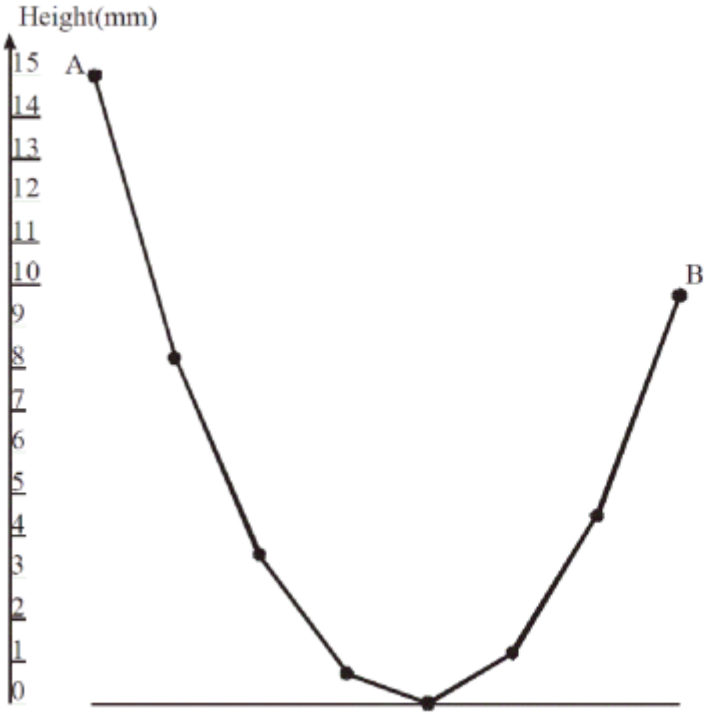
№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.062	4194304	1978102	1277538
1	OK	0.000	2224128	22	17
2	OK	0.000	2224128	20	38
3	OK	0.000	2236416	41	15
4	OK	0.000	2387968	204081	21587
5	OK	0.000	2392064	412716	21559
6	OK	0.000	2387968	412714	12243
7	OK	0.015	2527232	498728	612555
8	OK	0.015	3039232	1008458	612906
9	OK	0.015	3039232	1008832	341682
10	OK	0.031	2363392	471365	861755
11	OK	0.046	2842624	953290	859761
12	OK	0.046	2838528	953404	548738
13	OK	0.015	2371584	197660	51796
14	OK	0.015	2387968	399789	51761
15	OK	0.015	2371584	399826	29610
16	OK	0.046	2400256	511344	947660
17	OK	0.046	2924544	1034328	951787
18	OK	0.046	2924544	1034511	608920
19	OK	0.015	2449408	384717	274370
20	OK	0.031	2822144	777782	274601
21	OK	0.015	2822144	778270	152655
22	OK	0.015	2338816	219786	228823
23	OK	0.015	2363392	444845	228627
24	OK	0.000	2363392	444580	136297
25	OK	0.015	2613248	452007	84006
26	OK	0.015	3076096	914248	84077
27	OK	0.015	3076096	914384	46178
28	OK	0.015	2719744	534373	224808
29	OK	0.031	3264512	1080911	225002
30	OK	0.015	3264512	1080929	123417
31	OK	0.015	2641920	474858	115440
32	OK	0.015	3129344	960744	115495
33	OK	0.000	3129344	960330	63391
34	OK	0.046	3190784	977910	1277538
35	OK	0.046	4194304	1977816	1277396
36	OK	0.062	4194304	1978102	700050
37	OK	0.046	3182592	966605	1000288
38	OK	0.046	3182592	962679	1131278
39	OK	0.046	3219456	1000016	1200034
40	OK	0.046	3219456	1000016	1198665
41	OK	0.046	3076096	858730	1199466

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 ($h1=A$). 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

```
#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;
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.031	2633728	14	14
1	OK	0.015	2617344	9	9
2	OK	0.015	2621440	12	14
3	OK	0.000	2633728	9	9
4	OK	0.000	2617344	11	10
5	OK	0.015	2609152	9	9
6	OK	0.031	2617344	9	9
7	OK	0.015	2621440	14	14
8	OK	0.000	2621440	12	14
9	OK	0.031	2396160	11	14
10	OK	0.000	2404352	13	14
11	OK	0.015	2621440	10	10
12	OK	0.015	2408448	13	14
13	OK	0.000	2400256	10	9
14	OK	0.000	2621440	10	9
15	OK	0.000	2621440	12	14
16	OK	0.015	2400256	9	9
17	OK	0.031	2420736	12	14
18	OK	0.015	2412544	12	14
19	OK	0.000	2408448	12	13
20	OK	0.000	2400256	11	14
21	OK	0.015	2404352	11	14
22	OK	0.015	2412544	11	12
23	OK	0.000	2404352	11	9
24	OK	0.000	2416640	11	14
25	OK	0.000	2404352	12	14
26	OK	0.000	2396160	12	14
27	OK	0.000	2404352	12	14
28	OK	0.000	2416640	12	14
29	OK	0.000	2408448	12	14

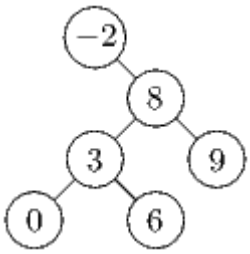
Высота дерева

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

```
#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;
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.031	16986112	3989144	6
1	OK	0.015	2441216	46	1
2	OK	0.000	2445312	3	1
3	OK	0.000	2445312	11	1
4	OK	0.000	2437120	18	1
5	OK	0.015	2445312	103	1
6	OK	0.015	2445312	76	2
7	OK	0.000	2457600	155	2
8	OK	0.015	2449408	163	2
9	OK	0.015	2445312	57	1
10	OK	0.000	2445312	161	1
11	OK	0.000	2445312	2099	1
12	OK	0.000	2220032	1197	3
13	OK	0.015	2236416	2073	3
14	OK	0.000	2220032	2139	3
15	OK	0.000	2224128	686	1
16	OK	0.000	2220032	2128	2
17	OK	0.000	2228224	8777	1
18	OK	0.000	2269184	10426	3
19	OK	0.000	2277376	16336	3
20	OK	0.000	2269184	16835	3
21	OK	0.000	2224128	3520	1
22	OK	0.015	2232320	16969	2
23	OK	0.000	2240512	36534	2
24	OK	0.000	2396160	38820	4
25	OK	0.000	2396160	55707	4
26	OK	0.000	2392064	57235	4
27	OK	0.000	2232320	7784	2
28	OK	0.000	2248704	56607	2
29	OK	0.015	2285568	149518	2
30	OK	0.015	2727936	117171	4
31	OK	0.015	2740224	164193	4
32	OK	0.015	2727936	168789	4
33	OK	0.000	2240512	29385	2
34	OK	0.000	2310144	171161	2

4) Удаление поддеревьев

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

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 109. 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.

Формат входного файла

Входной файл содержит описание двоичного дерева и описание запросов на удаление.

В первой строке файла находится число N ($1 \le N \le 2 \cdot 105$) — 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.

В следующей строке находится число M ($1 \le M \le 2 \cdot 105$) — 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 109 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.

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 6 9 7 8	5 4 4 1

```
#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;
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.078	11272192	6029382	1077960
1	OK	0.000	2441216	58	12
2	OK	0.015	2441216	27	12
3	OK	0.000	2457600	34	15
4	OK	0.015	2441216	211	30
5	OK	0.000	2220032	246	30
6	OK	0.015	2224128	3437	457
7	OK	0.000	2236416	3363	483
8	OK	0.000	2453504	18842	4247
9	OK	0.015	2252800	25683	3739
10	OK	0.000	2260992	69351	14791
11	OK	0.015	2277376	88936	11629
12	OK	0.015	2359296	244892	40297
13	OK	0.000	2367488	255614	37596
14	OK	0.015	3543040	978616	141281
15	OK	0.015	3346432	992647	137802
16	OK	0.046	5582848	2488583	634135
17	OK	0.046	7184384	3489729	483105
18	OK	0.078	8785920	4639039	1077960
19	OK	0.078	10997760	6007604	931260
20	OK	0.078	11272192	6029382	916969