**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 «Aлгоритмы и структуры данных»

**ОТЧЁТ**

по лабораторной работе №6 (Week 6 Openedu)

Студенка Кузенкова Елизавета группы P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

[Задача 1 Двоичный поиск 3](#_Toc5114884)

[Исходный код к задаче 1 3](#_Toc5114885)

[Бенчмарк к задаче 1 4](#_Toc5114886)

[Задача 2. Гирлянда 6](#_Toc5114887)

[Исходный код к задаче 2 7](#_Toc5114888)

[Бенчмарк к задаче 2 8](#_Toc5114889)

[Задача 3 Высота дерева 17](#_Toc5114890)

[Исходный код к задаче 3 19](#_Toc5114891)

[Бенчмарк к задаче 3 20](#_Toc5114892)

[Задача 4 Удаление поддеревьев 22](#_Toc5114893)

[Исходный код к задаче 4 23](#_Toc5114894)

[Бенчмарк к задаче 4 25](#_Toc5114895)

# Задача 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 элементов, упорядоченный в порядке неубывания, и 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 (). Во второй строке находятся чисел в порядке неубывания — элементы массива. В третьей строке находится число m — число запросов (). В следующей строке находятся чисел — запросы. Элементы массива и запросы являются целыми числами, неотрицательны и не превышают .

#### Формат выходного файла

Для каждого запроса выведите в отдельной строке номер (индекс) первого и последнего вхождения этого числа в массив. Ecли 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

class Lab6\_1

{

static Dictionary<int, string> \_results = new Dictionary<int, string>();

public static void Main(string[] args)

{

var app = new Lab6\_1();

app.DoWork(args);

}

private void DoWork(string[] args)

{

using (var sw = new StreamWriter("output.txt"))

{

string[] stdin = File.ReadAllLines("input.txt");

int[] array = stdin[1].Split(' ').Select(x => int.Parse(x)).ToArray();

int[] requests = stdin[3].Split(' ').Select(x => int.Parse(x)).ToArray();

for (int i = 0; i < requests.Length; i++)

if (\_results.ContainsKey(requests[i]))

sw.WriteLine(\_results[requests[i]]);

else

{

this.BinarySearch(array, requests[i]);

sw.WriteLine(\_results[requests[i]]);

}

}

}

void BinarySearch(int[] array, int value)

{

int l = -1, r = array.Length;

while (r > l + 1 || (r >= array.Length && l < 0 && array[l] != value && array[r] != value))

{

int m = (l + r) / 2;

if (array[m] < value)

l = m;

else

r = m;

}

if (r < array.Length && array[r] == value)

{

while (r < array.Length && array[r] == value) r++;

while (l >= 0 && array[l] == value) l--;

l += 2;

\_results.Add(value, string.Format("{0} {1}", l, r));

}

else

{

\_results.Add(value, string.Format("-1 -1"));

}

}

}

## Бенчмарк к задаче 1

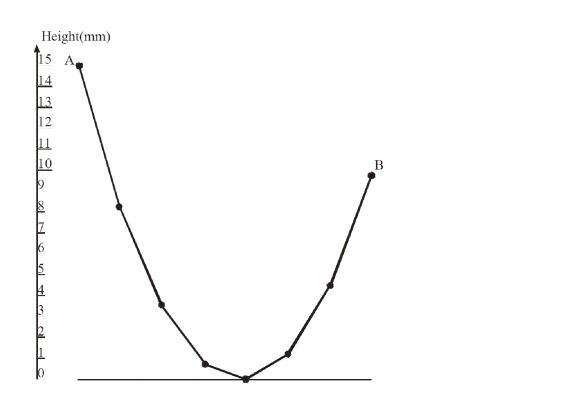
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.218 | 41418752 | 1978102 | 1277538 |
| 1 | OK | 0.031 | 11374592 | 22 | 17 |
| 2 | OK | 0.031 | 11390976 | 20 | 38 |
| 3 | OK | 0.031 | 11370496 | 41 | 15 |
| 4 | OK | 0.046 | 15769600 | 204081 | 21587 |
| 5 | OK | 0.046 | 16019456 | 412716 | 21559 |
| 6 | OK | 0.046 | 15749120 | 412714 | 12243 |
| 7 | OK | 0.078 | 20537344 | 498728 | 612555 |
| 8 | OK | 0.093 | 27041792 | 1008458 | 612906 |
| 9 | OK | 0.109 | 29265920 | 1008832 | 341682 |
| 10 | OK | 0.093 | 22646784 | 471365 | 861755 |
| 11 | OK | 0.093 | 27324416 | 953290 | 859761 |
| 12 | OK | 0.156 | 33837056 | 953404 | 548738 |
| 13 | OK | 0.046 | 16052224 | 197660 | 51796 |
| 14 | OK | 0.046 | 15785984 | 399789 | 51761 |
| 15 | OK | 0.062 | 15503360 | 399826 | 29610 |
| 16 | OK | 0.093 | 25362432 | 511344 | 947660 |
| 17 | OK | 0.109 | 28278784 | 1034328 | 951787 |
| 18 | OK | 0.140 | 34459648 | 1034511 | 608920 |
| 19 | OK | 0.062 | 17551360 | 384717 | 274370 |
| 20 | OK | 0.078 | 22843392 | 777782 | 274601 |
| 21 | OK | 0.078 | 22810624 | 778270 | 152655 |
| 22 | OK | 0.093 | 18374656 | 219786 | 228823 |
| 23 | OK | 0.046 | 17141760 | 444845 | 228627 |
| 24 | OK | 0.046 | 16699392 | 444580 | 136297 |
| 25 | OK | 0.062 | 20316160 | 452007 | 84006 |
| 26 | OK | 0.078 | 23511040 | 914248 | 84077 |
| 27 | OK | 0.062 | 22220800 | 914384 | 46178 |
| 28 | OK | 0.093 | 22716416 | 534373 | 224808 |
| 29 | OK | 0.078 | 26451968 | 1080911 | 225002 |
| 30 | OK | 0.078 | 26599424 | 1080929 | 123417 |
| 31 | OK | 0.078 | 20946944 | 474858 | 115440 |
| 32 | OK | 0.078 | 24301568 | 960744 | 115495 |
| 33 | OK | 0.078 | 23859200 | 960330 | 63391 |
| 34 | OK | 0.125 | 33361920 | 977910 | 1277538 |
| 35 | OK | 0.140 | 38150144 | 1977816 | 1277396 |
| 36 | OK | 0.218 | 41418752 | 1978102 | 700050 |
| 37 | OK | 0.093 | 31944704 | 966605 | 1000288 |
| 38 | OK | 0.093 | 31956992 | 962679 | 1131278 |
| 39 | OK | 0.093 | 32804864 | 1000016 | 1200034 |
| 40 | OK | 0.140 | 32808960 | 1000016 | 1198665 |
| 41 | OK | 0.093 | 29274112 | 858730 | 1199466 |

# Задача 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 (= A). Благодаря силе тяжести гирлянда прогибается: высота каждой неконцевой лампы на 1 мм меньше, чем средняя высота ближайших соседей ( для 1 < I < N).

Требуется найти минимальное значение высоты второго конца B (B = , такое что для любого Ɛ > 0 при высоте второго конца B + Ɛ для всех лампочек выполняется условие . 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 (, n — целое,  — 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 .

#### Примеры

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

## Исходный код к задаче 2

class Lab6\_2

{

public static void Main(string[] args)

{

var app = new Lab6\_2();

app.DoWork(args);

}

private void DoWork(string[] args)

{

using (var sw = new StreamWriter("output.txt"))

{

string[] stdin = File.ReadAllText("input.txt").Split(' ');

int n = int.Parse(stdin[0]);

double a = double.Parse(stdin[1]);

sw.WriteLine(findMinLastHeight(n, 0, a));

}

double findMinLastHeight(double n, double left, double right)

{

double last = -1;

double leftHeight = right;

while ((right - left) > 0.00000000001)

{

double mid = (left + right) / 2;

double prev = leftHeight;

double cur = mid;

bool aboveGround = cur > 0;

for (int i = 3; i <= n; i++)

{

double next = 2 \* cur - prev + 2;

aboveGround &= next > 0;

prev = cur;

cur = next;

}

if (aboveGround)

{

right = mid;

last = cur;

}

else

left = mid;

}

return last;

}

}

}

## Бенчмарк к задаче 2

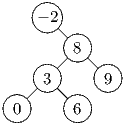
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.078 | 10481664 | 14 | 22 |
| 1 | OK | 0.062 | 10346496 | 9 | 17 |
| 2 | OK | 0.031 | 10481664 | 12 | 18 |
| 3 | OK | 0.031 | 10399744 | 9 | 21 |
| 4 | OK | 0.031 | 10383360 | 11 | 21 |
| 5 | OK | 0.031 | 10465280 | 9 | 22 |
| 6 | OK | 0.078 | 10366976 | 9 | 18 |
| 7 | OK | 0.031 | 10440704 | 14 | 18 |
| 8 | OK | 0.031 | 10448896 | 12 | 18 |
| 9 | OK | 0.046 | 10375168 | 11 | 18 |
| 10 | OK | 0.015 | 10371072 | 13 | 18 |
| 11 | OK | 0.031 | 10391552 | 10 | 21 |
| 12 | OK | 0.031 | 10199040 | 13 | 18 |
| 13 | OK | 0.046 | 10219520 | 10 | 22 |
| 14 | OK | 0.031 | 10170368 | 10 | 22 |
| 15 | OK | 0.031 | 10166272 | 12 | 18 |
| 16 | OK | 0.031 | 10190848 | 9 | 18 |
| 17 | OK | 0.015 | 10170368 | 12 | 18 |
| 18 | OK | 0.015 | 10235904 | 12 | 18 |
| 19 | OK | 0.031 | 10194944 | 12 | 18 |
| 20 | OK | 0.015 | 10199040 | 11 | 18 |
| 21 | OK | 0.031 | 10194944 | 11 | 18 |
| 22 | OK | 0.031 | 10158080 | 11 | 18 |
| 23 | OK | 0.031 | 10297344 | 11 | 22 |
| 24 | OK | 0.015 | 10145792 | 11 | 18 |
| 25 | OK | 0.062 | 10289152 | 12 | 18 |
| 26 | OK | 0.015 | 10207232 | 12 | 18 |
| 27 | OK | 0.031 | 10182656 | 12 | 18 |
| 28 | OK | 0.031 | 10194944 | 12 | 18 |
| 29 | OK | 0.031 | 10162176 | 12 | 18 |
| 30 | OK | 0.015 | 10235904 | 11 | 22 |
| 31 | OK | 0.031 | 10199040 | 12 | 17 |
| 32 | OK | 0.031 | 10190848 | 12 | 18 |
| 33 | OK | 0.031 | 10223616 | 11 | 22 |
| 34 | OK | 0.046 | 10162176 | 12 | 18 |
| 35 | OK | 0.046 | 10137600 | 12 | 18 |
| 36 | OK | 0.031 | 10153984 | 12 | 18 |
| 37 | OK | 0.015 | 10186752 | 12 | 18 |
| 38 | OK | 0.031 | 10166272 | 11 | 18 |
| 39 | OK | 0.031 | 10199040 | 12 | 18 |
| 40 | OK | 0.031 | 10149888 | 12 | 18 |
| 41 | OK | 0.015 | 10231808 | 12 | 18 |
| 42 | OK | 0.031 | 10297344 | 12 | 18 |
| 43 | OK | 0.015 | 10211328 | 11 | 17 |
| 44 | OK | 0.031 | 10207232 | 12 | 18 |
| 45 | OK | 0.031 | 10194944 | 12 | 18 |
| 46 | OK | 0.031 | 10194944 | 11 | 22 |
| 47 | OK | 0.015 | 10153984 | 12 | 18 |
| 48 | OK | 0.015 | 10178560 | 12 | 18 |
| 49 | OK | 0.015 | 10149888 | 11 | 17 |
| 50 | OK | 0.031 | 10166272 | 11 | 18 |
| 51 | OK | 0.015 | 10182656 | 12 | 18 |
| 52 | OK | 0.031 | 10199040 | 12 | 18 |
| 53 | OK | 0.031 | 10227712 | 11 | 18 |
| 54 | OK | 0.031 | 10219520 | 12 | 18 |
| 55 | OK | 0.031 | 10194944 | 12 | 17 |
| 56 | OK | 0.031 | 10166272 | 12 | 18 |
| 57 | OK | 0.015 | 10145792 | 12 | 18 |
| 58 | OK | 0.031 | 10158080 | 12 | 18 |
| 59 | OK | 0.046 | 10194944 | 12 | 18 |
| 60 | OK | 0.031 | 10141696 | 12 | 18 |
| 61 | OK | 0.015 | 10166272 | 12 | 18 |
| 62 | OK | 0.015 | 10162176 | 10 | 18 |
| 63 | OK | 0.031 | 10219520 | 12 | 17 |
| 64 | OK | 0.031 | 10219520 | 11 | 18 |
| 65 | OK | 0.031 | 10231808 | 12 | 18 |
| 66 | OK | 0.031 | 10203136 | 12 | 18 |
| 67 | OK | 0.031 | 10199040 | 10 | 17 |
| 68 | OK | 0.031 | 10166272 | 12 | 18 |
| 69 | OK | 0.031 | 10203136 | 12 | 18 |
| 70 | OK | 0.031 | 10153984 | 12 | 18 |
| 71 | OK | 0.046 | 10149888 | 11 | 18 |
| 72 | OK | 0.031 | 10162176 | 12 | 18 |
| 73 | OK | 0.031 | 10178560 | 12 | 18 |
| 74 | OK | 0.031 | 10231808 | 12 | 18 |
| 75 | OK | 0.015 | 10227712 | 12 | 17 |
| 76 | OK | 0.015 | 10240000 | 12 | 18 |
| 77 | OK | 0.031 | 10227712 | 12 | 18 |
| 78 | OK | 0.015 | 10231808 | 12 | 18 |
| 79 | OK | 0.031 | 10190848 | 12 | 18 |
| 80 | OK | 0.015 | 10174464 | 11 | 18 |
| 81 | OK | 0.031 | 10178560 | 12 | 18 |
| 82 | OK | 0.031 | 10153984 | 12 | 18 |
| 83 | OK | 0.031 | 10178560 | 11 | 18 |
| 84 | OK | 0.031 | 10158080 | 12 | 18 |
| 85 | OK | 0.031 | 10199040 | 11 | 18 |
| 86 | OK | 0.031 | 10199040 | 12 | 18 |
| 87 | OK | 0.015 | 10223616 | 12 | 18 |
| 88 | OK | 0.046 | 10227712 | 11 | 18 |
| 89 | OK | 0.031 | 10199040 | 12 | 18 |
| 90 | OK | 0.031 | 10153984 | 12 | 18 |
| 91 | OK | 0.015 | 10178560 | 12 | 18 |
| 92 | OK | 0.015 | 10190848 | 12 | 18 |
| 93 | OK | 0.031 | 10137600 | 12 | 18 |
| 94 | OK | 0.031 | 10223616 | 12 | 18 |
| 95 | OK | 0.015 | 10178560 | 12 | 18 |
| 96 | OK | 0.031 | 10186752 | 12 | 17 |
| 97 | OK | 0.031 | 10215424 | 12 | 18 |
| 98 | OK | 0.031 | 10231808 | 12 | 17 |
| 99 | OK | 0.031 | 10186752 | 11 | 17 |
| 100 | OK | 0.046 | 10203136 | 11 | 21 |
| 101 | OK | 0.015 | 10231808 | 12 | 18 |
| 102 | OK | 0.031 | 10186752 | 11 | 18 |
| 103 | OK | 0.031 | 10207232 | 12 | 18 |
| 104 | OK | 0.031 | 10223616 | 11 | 18 |
| 105 | OK | 0.031 | 10149888 | 12 | 17 |
| 106 | OK | 0.015 | 10166272 | 11 | 18 |
| 107 | OK | 0.015 | 10227712 | 12 | 18 |
| 108 | OK | 0.015 | 10170368 | 11 | 18 |
| 109 | OK | 0.031 | 10211328 | 12 | 18 |
| 110 | OK | 0.031 | 10199040 | 12 | 18 |
| 111 | OK | 0.015 | 10149888 | 11 | 18 |
| 112 | OK | 0.015 | 10231808 | 12 | 17 |
| 113 | OK | 0.031 | 10186752 | 12 | 18 |
| 114 | OK | 0.015 | 10190848 | 11 | 18 |
| 115 | OK | 0.015 | 10223616 | 12 | 18 |
| 116 | OK | 0.031 | 10145792 | 12 | 18 |
| 117 | OK | 0.015 | 10207232 | 12 | 18 |
| 118 | OK | 0.015 | 10182656 | 12 | 18 |
| 119 | OK | 0.031 | 10170368 | 11 | 18 |
| 120 | OK | 0.015 | 10178560 | 12 | 18 |
| 121 | OK | 0.031 | 10186752 | 12 | 18 |
| 122 | OK | 0.031 | 10178560 | 12 | 18 |
| 123 | OK | 0.031 | 10219520 | 12 | 18 |
| 124 | OK | 0.031 | 10186752 | 12 | 18 |
| 125 | OK | 0.015 | 10223616 | 12 | 18 |
| 126 | OK | 0.031 | 10182656 | 12 | 18 |
| 127 | OK | 0.031 | 10199040 | 12 | 18 |
| 128 | OK | 0.031 | 10190848 | 12 | 18 |
| 129 | OK | 0.015 | 10199040 | 12 | 18 |
| 130 | OK | 0.015 | 10252288 | 12 | 18 |
| 131 | OK | 0.031 | 10199040 | 12 | 18 |
| 132 | OK | 0.031 | 10211328 | 12 | 18 |
| 133 | OK | 0.015 | 10145792 | 12 | 18 |
| 134 | OK | 0.031 | 10285056 | 12 | 18 |
| 135 | OK | 0.031 | 10170368 | 12 | 16 |
| 136 | OK | 0.031 | 10211328 | 12 | 18 |
| 137 | OK | 0.031 | 10194944 | 12 | 18 |
| 138 | OK | 0.031 | 10145792 | 12 | 17 |
| 139 | OK | 0.031 | 10219520 | 12 | 18 |
| 140 | OK | 0.031 | 10141696 | 12 | 18 |
| 141 | OK | 0.031 | 10178560 | 12 | 17 |
| 142 | OK | 0.015 | 10256384 | 12 | 18 |
| 143 | OK | 0.031 | 10158080 | 12 | 17 |
| 144 | OK | 0.031 | 10158080 | 12 | 17 |
| 145 | OK | 0.031 | 10219520 | 12 | 18 |
| 146 | OK | 0.031 | 10215424 | 12 | 18 |
| 147 | OK | 0.015 | 10215424 | 12 | 18 |
| 148 | OK | 0.015 | 10297344 | 12 | 18 |
| 149 | OK | 0.031 | 10178560 | 12 | 18 |
| 150 | OK | 0.015 | 10190848 | 11 | 18 |
| 151 | OK | 0.031 | 10141696 | 12 | 18 |
| 152 | OK | 0.015 | 10158080 | 12 | 17 |
| 153 | OK | 0.015 | 10174464 | 12 | 18 |
| 154 | OK | 0.031 | 10162176 | 12 | 18 |
| 155 | OK | 0.031 | 10145792 | 12 | 18 |
| 156 | OK | 0.031 | 10223616 | 12 | 18 |
| 157 | OK | 0.031 | 10231808 | 12 | 18 |
| 158 | OK | 0.031 | 10203136 | 12 | 18 |
| 159 | OK | 0.031 | 10227712 | 12 | 18 |
| 160 | OK | 0.015 | 10170368 | 12 | 18 |
| 161 | OK | 0.031 | 10178560 | 12 | 18 |
| 162 | OK | 0.031 | 10153984 | 11 | 17 |
| 163 | OK | 0.015 | 10190848 | 11 | 18 |
| 164 | OK | 0.078 | 10207232 | 12 | 18 |
| 165 | OK | 0.031 | 10149888 | 12 | 17 |
| 166 | OK | 0.031 | 10182656 | 12 | 18 |
| 167 | OK | 0.046 | 10223616 | 12 | 18 |
| 168 | OK | 0.031 | 10199040 | 12 | 18 |
| 169 | OK | 0.015 | 10162176 | 12 | 18 |
| 170 | OK | 0.031 | 10223616 | 12 | 18 |
| 171 | OK | 0.015 | 10215424 | 12 | 18 |
| 172 | OK | 0.031 | 10186752 | 12 | 17 |
| 173 | OK | 0.031 | 10149888 | 12 | 18 |
| 174 | OK | 0.031 | 10199040 | 12 | 18 |
| 175 | OK | 0.031 | 10145792 | 12 | 18 |
| 176 | OK | 0.015 | 10166272 | 12 | 18 |
| 177 | OK | 0.031 | 10145792 | 12 | 18 |
| 178 | OK | 0.031 | 10207232 | 12 | 18 |
| 179 | OK | 0.015 | 10215424 | 12 | 17 |
| 180 | OK | 0.015 | 10133504 | 12 | 18 |
| 181 | OK | 0.031 | 10223616 | 12 | 17 |
| 182 | OK | 0.031 | 10227712 | 12 | 18 |
| 183 | OK | 0.015 | 10194944 | 12 | 18 |
| 184 | OK | 0.031 | 10199040 | 12 | 18 |
| 185 | OK | 0.015 | 10186752 | 12 | 18 |
| 186 | OK | 0.015 | 10190848 | 11 | 18 |
| 187 | OK | 0.031 | 10162176 | 12 | 17 |
| 188 | OK | 0.031 | 10186752 | 9 | 22 |
| 189 | OK | 0.031 | 10199040 | 11 | 18 |
| 190 | OK | 0.015 | 10231808 | 12 | 18 |
| 191 | OK | 0.031 | 10186752 | 12 | 18 |
| 192 | OK | 0.015 | 10223616 | 12 | 18 |
| 193 | OK | 0.031 | 10145792 | 12 | 18 |
| 194 | OK | 0.031 | 10170368 | 12 | 18 |
| 195 | OK | 0.015 | 10207232 | 12 | 18 |
| 196 | OK | 0.015 | 10170368 | 12 | 17 |
| 197 | OK | 0.031 | 10256384 | 12 | 18 |
| 198 | OK | 0.015 | 10170368 | 12 | 18 |
| 199 | OK | 0.031 | 10256384 | 12 | 18 |
| 200 | OK | 0.015 | 10272768 | 11 | 18 |
| 201 | OK | 0.031 | 10227712 | 12 | 18 |
| 202 | OK | 0.031 | 10166272 | 12 | 17 |
| 203 | OK | 0.015 | 10190848 | 12 | 18 |
| 204 | OK | 0.015 | 10141696 | 12 | 18 |
| 205 | OK | 0.031 | 10190848 | 12 | 18 |
| 206 | OK | 0.015 | 10231808 | 12 | 18 |
| 207 | OK | 0.031 | 10178560 | 12 | 18 |
| 208 | OK | 0.015 | 10170368 | 11 | 18 |
| 209 | OK | 0.031 | 10158080 | 12 | 17 |
| 210 | OK | 0.031 | 10174464 | 11 | 18 |
| 211 | OK | 0.015 | 10215424 | 11 | 18 |
| 212 | OK | 0.031 | 10207232 | 11 | 18 |
| 213 | OK | 0.046 | 10162176 | 10 | 18 |
| 214 | OK | 0.015 | 10223616 | 12 | 17 |
| 215 | OK | 0.031 | 10158080 | 12 | 18 |
| 216 | OK | 0.031 | 10166272 | 12 | 18 |
| 217 | OK | 0.031 | 10215424 | 12 | 18 |
| 218 | OK | 0.015 | 10145792 | 11 | 18 |
| 219 | OK | 0.015 | 10149888 | 12 | 18 |
| 220 | OK | 0.031 | 10207232 | 11 | 18 |
| 221 | OK | 0.015 | 10174464 | 12 | 18 |
| 222 | OK | 0.015 | 10223616 | 12 | 18 |
| 223 | OK | 0.015 | 10293248 | 11 | 18 |
| 224 | OK | 0.031 | 10149888 | 11 | 18 |
| 225 | OK | 0.031 | 10215424 | 12 | 18 |
| 226 | OK | 0.031 | 10158080 | 12 | 18 |
| 227 | OK | 0.015 | 10203136 | 12 | 18 |
| 228 | OK | 0.015 | 10178560 | 12 | 18 |
| 229 | OK | 0.031 | 10199040 | 12 | 18 |
| 230 | OK | 0.031 | 10199040 | 12 | 18 |
| 231 | OK | 0.015 | 10153984 | 12 | 18 |
| 232 | OK | 0.031 | 10240000 | 12 | 18 |
| 233 | OK | 0.015 | 10219520 | 12 | 18 |
| 234 | OK | 0.031 | 10231808 | 12 | 17 |
| 235 | OK | 0.031 | 10162176 | 12 | 18 |
| 236 | OK | 0.031 | 10211328 | 11 | 18 |
| 237 | OK | 0.015 | 10133504 | 11 | 17 |
| 238 | OK | 0.031 | 10219520 | 11 | 18 |
| 239 | OK | 0.015 | 10162176 | 12 | 18 |
| 240 | OK | 0.031 | 10194944 | 12 | 18 |
| 241 | OK | 0.031 | 10174464 | 12 | 18 |
| 242 | OK | 0.031 | 10190848 | 12 | 18 |
| 243 | OK | 0.015 | 10153984 | 12 | 18 |
| 244 | OK | 0.015 | 10223616 | 12 | 18 |
| 245 | OK | 0.031 | 10203136 | 12 | 18 |
| 246 | OK | 0.031 | 10182656 | 10 | 22 |
| 247 | OK | 0.031 | 10219520 | 11 | 18 |
| 248 | OK | 0.015 | 10231808 | 12 | 18 |
| 249 | OK | 0.015 | 10149888 | 12 | 18 |
| 250 | OK | 0.031 | 10190848 | 12 | 18 |

# Задача 3 Высота дерева

|  |  |
| --- | --- |
| 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 . Для каждой вершины дерева выполняется следующее условие:

* все ключи вершин из левого поддерева меньше ключа вершины
* все ключи вершин из правого поддерева больше ключа вершины

Найдите высоту данного дерева.

#### Формат входного файла

Входной файл содержит описание двоичного дерева. В первой строке файла находится число   — число вершин в дереве. В последующих N строках файла находятся описания вершин дерева. В (i + 1)-ой строке файла (1 ) находится описание i-ой вершины, состоящее из трех чисел Ki, Li, Ri, разделенных пробелами — ключа в i-ой вершине -, номера левого ребенка i-ой вершины ( i < Li < N или Li = 0 , если левого ребенка нет) и номера правого ребенка i-ой вершины ( I < Ri < 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 |

## Исходный код к задаче 3

class Lab6\_3

{

class Tree

{

class Node

{

public int Key { get; set; }

public int Parent { get; set; }

public int Left { get; set; }

public int Right { get; set; }

}

public int Height { get { if (\_vertexes != null) return this.CalcHeight(); else return -1; } }

private int CalcHeight()

{

int heigth = 0;

for (int i = 0; i < \_rootIndexes.Count; i++)

{

int tempHeigth = 1;

int currentNode = \_rootIndexes[i];

while (\_vertexes[currentNode].Parent != 0)

{

tempHeigth++;

currentNode = \_vertexes[currentNode].Parent;

}

if (heigth < tempHeigth)

heigth = tempHeigth;

}

return heigth;

}

Node[] \_vertexes;

List<int> \_rootIndexes;

public void MakeTree(string[] stdin)

{

\_rootIndexes = new List<int>();

\_vertexes = new Node[stdin.Length];

for (int i = 1; i < stdin.Length; i++)

{

int[] temp = stdin[i].Split(' ').Select(x => int.Parse(x)).ToArray();

if (\_vertexes[i] == null)

\_vertexes[i] = new Node();

\_vertexes[i].Key = temp[0];

\_vertexes[i].Left = temp[1];

\_vertexes[i].Right = temp[2];

if (temp[1] != 0)

{

\_vertexes[temp[1]] = new Node();

\_vertexes[temp[1]].Parent = i;

}

if (temp[2] != 0)

{

\_vertexes[temp[2]] = new Node();

\_vertexes[temp[2]].Parent = i;

}

if (temp[1] == 0 && temp[2] == 0)

\_rootIndexes.Add(i);

}

}

}

public static void Main(string[] args)

{

var app = new Lab6\_3();

app.DoWork(args);

}

private void DoWork(string[] args)

{

using (var sw = new StreamWriter("output.txt"))

{

string[] stdin = File.ReadAllLines("input.txt");

if (int.Parse(stdin[0]) == 0)

sw.Write(0);

else

{

var tree = new Tree();

tree.MakeTree(stdin);

sw.WriteLine(tree.Height);

}

}

}

}

## Бенчмарк к задаче 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.375 | 45043712 | 3989144 | 8 |
| 1 | OK | 0.046 | 11120640 | 46 | 3 |
| 2 | OK | 0.031 | 10035200 | 3 | 1 |
| 3 | OK | 0.031 | 11104256 | 11 | 3 |
| 4 | OK | 0.046 | 11055104 | 18 | 3 |
| 5 | OK | 0.015 | 11083776 | 103 | 3 |
| 6 | OK | 0.031 | 11051008 | 76 | 4 |
| 7 | OK | 0.046 | 11173888 | 155 | 4 |
| 8 | OK | 0.078 | 11067392 | 163 | 4 |
| 9 | OK | 0.031 | 11030528 | 57 | 3 |
| 10 | OK | 0.031 | 11075584 | 161 | 3 |
| 11 | OK | 0.031 | 11108352 | 2099 | 3 |
| 12 | OK | 0.031 | 11206656 | 1197 | 5 |
| 13 | OK | 0.031 | 11169792 | 2073 | 5 |
| 14 | OK | 0.031 | 11145216 | 2139 | 5 |
| 15 | OK | 0.031 | 11141120 | 686 | 3 |
| 16 | OK | 0.031 | 11108352 | 2128 | 4 |
| 17 | OK | 0.015 | 11341824 | 8777 | 3 |
| 18 | OK | 0.046 | 11620352 | 10426 | 5 |
| 19 | OK | 0.031 | 11591680 | 16336 | 5 |
| 20 | OK | 0.031 | 11595776 | 16835 | 5 |
| 21 | OK | 0.031 | 11169792 | 3520 | 3 |
| 22 | OK | 0.031 | 11620352 | 16969 | 4 |
| 23 | OK | 0.031 | 12296192 | 36534 | 4 |
| 24 | OK | 0.031 | 12218368 | 38820 | 6 |
| 25 | OK | 0.046 | 12259328 | 55707 | 6 |
| 26 | OK | 0.031 | 12238848 | 57235 | 6 |
| 27 | OK | 0.031 | 11300864 | 7784 | 4 |
| 28 | OK | 0.062 | 12283904 | 56607 | 4 |
| 29 | OK | 0.046 | 15933440 | 149518 | 4 |
| 30 | OK | 0.046 | 15974400 | 117171 | 6 |
| 31 | OK | 0.046 | 16408576 | 164193 | 6 |
| 32 | OK | 0.046 | 16461824 | 168789 | 6 |
| 33 | OK | 0.031 | 12046336 | 29385 | 4 |
| 34 | OK | 0.031 | 16519168 | 171161 | 4 |
| 35 | OK | 0.093 | 21315584 | 624213 | 4 |
| 36 | OK | 0.093 | 21442560 | 489475 | 7 |
| 37 | OK | 0.078 | 21659648 | 637029 | 7 |
| 38 | OK | 0.093 | 21651456 | 654072 | 7 |
| 39 | OK | 0.031 | 12435456 | 62037 | 4 |
| 40 | OK | 0.078 | 21602304 | 666913 | 4 |
| 41 | OK | 0.125 | 27365376 | 1259549 | 4 |
| 42 | OK | 0.218 | 36003840 | 1788745 | 8 |
| 43 | OK | 0.218 | 32899072 | 2254723 | 8 |
| 44 | OK | 0.218 | 32886784 | 2313971 | 8 |
| 45 | OK | 0.046 | 15998976 | 152298 | 4 |
| 46 | OK | 0.234 | 32911360 | 2306482 | 4 |
| 47 | OK | 0.250 | 34537472 | 2561292 | 4 |
| 48 | OK | 0.328 | 42254336 | 3177798 | 8 |
| 49 | OK | 0.375 | 44634112 | 3888903 | 8 |
| 50 | OK | 0.359 | 44683264 | 3989144 | 8 |
| 51 | OK | 0.031 | 17510400 | 200543 | 4 |
| 52 | OK | 0.343 | 45043712 | 3953465 | 4 |

# Задача 4 Удаление поддеревьев

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 этого дерева записаны ключи — целые числа, по модулю не превышающие . Для каждой вершины дерева выполняется следующее условие:

* все ключи вершин из левого поддерева меньше ключа вершины
* все ключи вершин из правого поддерева больше ключа вершины

**Высота дерева не превосходит 25**, таким образом, можно считать, что оно сбалансировано.

Найдите высоту данного дерева.

#### Формат входного файла

Входной файл содержит описание двоичного дерева. В первой строке файла находится число   — число вершин в дереве. В последующих N строках файла находятся описания вершин дерева. В (i + 1)-ой строке файла (1 ) находится описание i-ой вершины, состоящее из трех чисел Ki, Li, Ri, разделенных пробелами — ключа в i-ой вершине -, номера левого ребенка i-ой вершины ( i < Li < N или Li = 0 , если левого ребенка нет) и номера правого ребенка i-ой вершины ( I < Ri < 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 — 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 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.

## Исходный код к задаче 4

class Lab6\_4

{

public class Tree

{

public class Node

{

public int Key { get; set; }

public Node Parent { get; set; }

public Node Left { get; set; }

public Node Right { get; set; }

}

public long Size { get; private set; }

public Node[] Vertexes { get; private set; }

List<int> \_rootIndexes;

public Tree(long n)

{

this.Size = n;

}

public void MakeTree(string[] stdin)

{

\_rootIndexes = new List<int>();

this.Vertexes = new Node[this.Size];

for (int i = 1; i <= this.Size; i++)

{

int[] temp = stdin[i].Split(' ').Select(x => int.Parse(x)).ToArray();

if (this.Vertexes[i - 1] == null)

this.Vertexes[i - 1] = new Node();

this.Vertexes[i - 1].Key = temp[0];

if (temp[1] != 0)

{

//Left

if (temp[1] != 0)

{

if (this.Vertexes[temp[1] - 1] == null)

this.Vertexes[temp[1] - 1] = new Node() { Parent = this.Vertexes[i - 1] };

this.Vertexes[i - 1].Left = this.Vertexes[temp[1] - 1];

}

}

//Righ

if (temp[2] != 0)

{

if (temp[2] != 0 && this.Vertexes[temp[2] - 1] == null)

this.Vertexes[temp[2] - 1] = new Node() { Parent = this.Vertexes[i - 1] };

this.Vertexes[i - 1].Right = this.Vertexes[temp[2] - 1];

}

}

}

public long RemoveNode(long request, Node root)

{

if (root != null)

{

Node node = this.Search(root, request);

if (node != null)

{

if (node == root)

{

root = null;

return 0;

}

else

{

if (node.Parent.Right == node)

node.Parent.Right = null;

else

node.Parent.Left = null;

}

this.Size -= this.Count(node);

}

}

return this.Size;

}

public long Count(Node node)

{

long result = 1;

if (node.Left != null)

result += this.Count(node.Left);

if (node.Right != null)

result += this.Count(node.Right);

return result;

}

public Node Search(Node node, long value)

{

while (value != node.Key &&

(value < node.Key && node.Left != null) || (value > node.Key && node.Right != null))

{

if (value < node.Key && node.Left != null)

node = node.Left;

if (value > node.Key && node.Right != null)

node = node.Right;

}

if (value == node.Key)

return node;

else

return null;

}

}

public static void Main(string[] args)

{

var app = new Lab6\_4();

app.DoWork(args);

}

private void DoWork(string[] args)

{

using (var sw = new StreamWriter("output.txt"))

{

string[] stdin = File.ReadAllLines("input.txt");

long n = long.Parse(stdin[0]);

var tree = new Tree(n);

tree.MakeTree(stdin);

long[] requests = stdin[n + 2].Split(' ').Select(x => long.Parse(x)).ToArray();

var root = tree.Vertexes[0];

while (root.Parent != null)

root = root.Parent;

for (int i = 0; i < requests.Length; i++)

{

sw.WriteLine(tree.RemoveNode(requests[i], root));

}

}

}

}

## Бенчмарк к задаче 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.562 | 65310720 | 6029382 | 1077960 |
| 1 | OK | 0.046 | 11304960 | 58 | 12 |
| 2 | OK | 0.031 | 11210752 | 27 | 12 |
| 3 | OK | 0.031 | 11223040 | 34 | 15 |
| 4 | OK | 0.031 | 11259904 | 211 | 30 |
| 5 | OK | 0.046 | 11243520 | 246 | 30 |
| 6 | OK | 0.031 | 11431936 | 3437 | 457 |
| 7 | OK | 0.046 | 11460608 | 3363 | 483 |
| 8 | OK | 0.031 | 11739136 | 18842 | 4247 |
| 9 | OK | 0.046 | 11952128 | 25683 | 3739 |
| 10 | OK | 0.062 | 12492800 | 69351 | 14791 |
| 11 | OK | 0.046 | 13139968 | 88936 | 11629 |
| 12 | OK | 0.062 | 17719296 | 244892 | 40297 |
| 13 | OK | 0.046 | 18059264 | 255614 | 37596 |
| 14 | OK | 0.093 | 22204416 | 978616 | 141281 |
| 15 | OK | 0.109 | 22306816 | 992647 | 137802 |
| 16 | OK | 0.250 | 37384192 | 2488583 | 634135 |
| 17 | OK | 0.343 | 47448064 | 3489729 | 483105 |
| 18 | OK | 0.437 | 56176640 | 4639039 | 1077960 |
| 19 | OK | 0.562 | 65126400 | 6007604 | 931260 |
| 20 | OK | 0.546 | 65310720 | 6029382 | 916969 |