

## Очередь на Луну. 11 классы

Коротышки из Цветочного города готовятся отправиться в путешествие на Луну. Знайка поручил Незнайке, выстроить всех коротышек в очередь на посадку в космический корабль так, чтобы они были упорядочены по росту: первым должен стоять самый высокий, последним – самый низкий. Однако, пока Незайка витал в облаках, все перепутались и встали в очередь, кто как хотел.

Строгий Знайка решил навести порядок и наказать всех тех, кто встал в очередь не по порядку. Знайка решил выбрать самую длинную цепочку коротышек, которые уже стоят (не обязательно рядом) в правильном порядке (по невозрастанию роста). Всех остальных, кто не входит в эту цепочку, он будет наказывать, заставляя учить лунные правила. Помогите Знайке узнать, каково количество коротышек, которые окажутся вне цепочки и будут наказаны.

Требуется составить программу, которая считывает натуральное число  $N$ :  $0 < N < 10001$  – общее количество коротышек в очереди, а затем  $N$  натуральных чисел  $H[i]$ :  $0 < H[i] < 1000000001$ .  $H[i]$  – рост коротышки, стоящего на  $i$ -ом месте в очереди. Программа находит наименьшее целое число  $K$  ( $-1 < K < N$ ) – количество коротышек, которые не входят в цепочку из коротышек, стоящих в очереди на местах с номерами  $i_1, i_2, \dots, i_{N-K}$ , где  $i_1 < i_2 < \dots < i_{N-K}$ , и  $H[i_1], H[i_2], \dots, H[i_{N-K}]$  является невозрастающей, т. е.  $H[i_j] > H[i_{j+1}]$  или  $H[i_j] = H[i_{j+1}]$  для любого  $j$  ( $0 < j < N - K$ ).

**Формат ввода:** В первой строке задано число  $N$ :  $0 < N < 10001$ . Во второй строке заданы  $N$  натуральных чисел  $H[i]$ :  $0 < H[i] < 1000000001$ .

**Формат вывода:** Выводится искомое число  $K$ :  $-1 < K < N$ .

Ввод примера №1:		Ввод примера №2:		Ввод примера №3:
6		5		5
6 28 5 5 29 3		1 4 10 8 5		2 2 2 1 1
Вывод примера №1:		Вывод примера №2:		Вывод примера №3:
2		2		0
Пояснение:		Пояснение:		Пояснение:
Цепочка: 6 5 5 3		Цепочка: 10 8 5		Все выстроились верно.
Вне цепочки: 29 28		Вне цепочки: 1 4		

## Пояснения к решению

Вместо того чтобы искать наименьшее количество наказуемых, будем искать наибольшее количество тех, кого наказывать не нужно. Тогда ответ можно получить как  $ans = N - LnIS$ , где  $LnIS$  - это длина наибольшей невозрастающей подпоследовательности. Вместо того, чтобы вычислять длину наибольшей невозрастающей подпоследовательности можно развернуть массив и свести задачу к вычислению длины наибольшей неубывающей подпоследовательности  $LnDS$ . При данных ограничениях на  $N \leq 10^4$  задачу можно решить с помощью метода динамического программирования. Для этого определим «динамику»  $dp[i]$  как минимальное значение, на которое может заканчиваться неубывающая подпоследовательность длины  $i$ . Изначально  $dp[0] = -\infty$  а  $dp[1] = dp[2] = \dots = \infty$ . Рассматриваемая «динамика» является монотонной, а значит в ней можно бинарным поиском по  $H[i]$  находить точную верхнюю границу, куда можно внести элемент  $H[i]$ , чтобы образовать неубывающую подпоследовательность. Для каждого значения массива нужно запустить бинарный поиск по «динамике».

## Код возможного решения

```
#include <bits/stdc++.h>

using namespace std;

signed
main(void)
{
    int n;
    cin >> n;
```

```

if (n == 0) {
    cout << 0 << endl;
    return 0;
}

vector<int> a(n);
for (auto &i : a) {
    cin >> i;
}

reverse(a.begin(), a.end());

const int INF = 1e9 + 1e8;
vector<int> d(n + 1, INF);
d[0] = -INF;

for (int i = 0; i < n; i++) {
    int l = upper_bound(d.begin(), d.end(), a[i]) - d.begin();
    if (d[l - 1] <= a[i] && a[i] < d[l]) {
        d[l] = a[i];
    }
}

int ans = 0;
for (int l = 0; l <= n; l++) {
    if (d[l] < INF) {
        ans = l;
    }
}

int punish = n - ans;
cout << punish << endl;

return 0;
}

```