

Лунная прогулка. 11 классы

На Луне, куда прилетел Незнайка, поверхность покрыта лунными горами. Каждая гора имеет три характеристики: 1) x – координата долготы (восток-запад), 2) y – координата широты (север-юг), 3) h – высота горы над лунной равниной.

Незнайка любит покорять вершины, но так как гор много, он придумал себе специальные ограничения, чтобы не запутаться. Он может стартовать с любой горы. Дальше он хочет прыгать с горы на гору, но только если выполняются такие правила:

1. Новая гора должна быть правее (восточнее) текущей: $x_2 > x_1$.
2. Новая гора должна быть выше на карте (севернее) текущей: $y_2 > y_1$.
3. Новая гора должна быть ниже той, с которой прыгает Незнайка: $h_2 < h_1$.

После прыжка Незнайка останавливается на новой горе и снова может прыгать дальше, если есть подходящая цель. Помогите Незнайке узнать, какое максимальное количество гор он сможет посетить за одну такую прогулку?

Формат ввода: Первая строка содержит одно целое число n ($1 \leq n \leq 5000$) – количество гор. В следующих n строках содержатся по три целых числа в каждой строке: $0 \leq x_i, y_i, h_i \leq 10^9$ – координаты и высота i горы.

Формат вывода: Выводится одно целое число – максимальное количество гор, которые Незнайка может посетить за одну прогулку.

Ввод примера №1:

```
5
0 0 10
1 2 9
2 1 8
3 3 7
4 4 6
```

Вывод примера №1:

```
4
```

Ввод примера №2:

```
6
0 0 5
1 1 6
2 2 4
3 3 3
4 4 2
5 5 1
```

Вывод примера №2:

```
5
```

Ввод примера №3:

```
10
0 0 5
1 1 4
2 2 3
3 3 10
4 4 9
5 5 8
0 0 4
0 100 15
6 6 7
7 7 7
```

Вывод примера №3:

```
4
```

Решение

Любой допустимый прыжок во время прогулки всегда увеличивает x , значит после сортировки по возрастанию x любая ранее посещенная гора обязательно стоит раньше текущей горы. Отсортируем все горы по возрастанию x . Пусть $dp[i]$ – максимальное число гор в маршруте, оканчивающемся в i -й горе в этом порядке. Тогда $dp[i] = 1 + \max(dp[j])$ по всем $j < i$, для которых одновременно $(x[j] < x[i]) \text{ and } (y[j] < y[i]) \text{ and } (h[j] > h[i])$. Ответом тогда будет $\max(dp[i])$.

Код возможного решения

```
#include <bits/stdc++.h>
using namespace std;

struct Mountain {
    long long x, y, h;
};

bool cmp(const Mountain &A, const Mountain &B) {
    if (A.x != B.x) return A.x < B.x;
    if (A.y != B.y) return A.y < B.y;
    return A.h < B.h;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<Mountain> a(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i].x >> a[i].y >> a[i].h;
    }

    sort(a.begin(), a.end(), cmp);

    vector<int> dp(n, 1);
    int ans = 1;

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < i; ++j) {
            if (a[j].x < a[i].x &&
                a[j].y < a[i].y &&
                a[j].h > a[i].h) {
                dp[i] = max(dp[i], dp[j] + 1);
            }
        }
        ans = max(ans, dp[i]);
    }

    cout << ans << '\n';
    return 0;
}
```