Tutorial K

03-01-0v0

October 2021

1 Tóm tắt đề bài

Có 1 thiên hà diện tích vô hạn và **W** lãnh chúa, mỗi lãnh chúa cần có ít nhất một lãnh thổ cho vương quốc của mình nếu không sẽ xảy ra chiến tranh.

Theo thảo luận ban đầu, thì sẽ dùng **N** đường thẳng để chia diện tích thiên hà thành các phần cho các lãnh chúa nhưng có thể tập đường thẳng này vẫn chưa chia được thiên hà thành các phần phù hợp để các lãnh chúa không chiến tranh.

Các lãnh chúa đã quyết định sẽ có một sửa đổi bổ sung, cho phép thêm một số đường thẳng nữa. Câu hỏi đặt ra là phải thêm ít nhất bao nhiêu đường thẳng để các lãnh chúa sẽ không xảy ra chiến tranh ?

2 Chú ý

Vì diện tích thiên hà là vô hạn nên tổng 2 phần diện tích vô hạn bằng vô hạn: $S1_\infty + S2_\infty = S_\infty$.

3 Lời giải

Đầu tiên, chúng ta phải loại bỏ hết tất cả các đường thẳng trùng nhau, vì 2 đường thẳng trùng nhau sẽ không chia thiên hà thành nhiều phần hơn.

Cách kiểm tra 2 đường thẳng trùng nhau:

Gọi 2 điểm A, B tạo lên đường thẳng d_1.
Gọi 2 điểm C, D tạo lên đường thẳng d_2.
Gọi 2 điểm A, C tạo lên đường thẳng d_3.
Gọi 2 điểm A, D tạo lên đường thẳng d_4.
2 đường thẳng : $d_1 \equiv d_2 \iff d_1 \equiv d_3 \; and \; d_1 \equiv d_4$

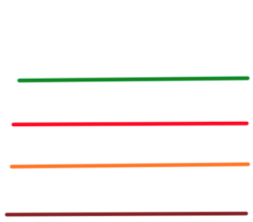
Cách viết phương trình đường thẳng đi qua 2 điểm $A(x_1, y_1)$, $B(x_2, y_2)$, ta có vector pháp tuyến $\overrightarrow{n}(y_1 - y_2, x_2 - x_1)$.

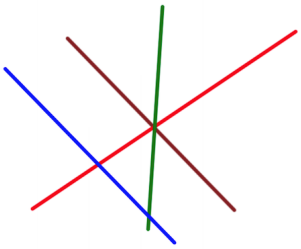
Phương trình đường thẳng có dạng : $(y_1-y_2)*(x-x_1)+(x_2-x_1)*(y-y_1) = 0$.

Giả sử có **x** đường thẳng phân biệt, đến đây sẽ có 2 trường hợp:

TH1: Tất cả các đường thẳng song song với nhau, ta sẽ x + 1 phần lãnh thổ.
TH2: Nếu không ta sẽ có 2 * x phần lãnh thổ.

Vậy ta sẽ kiểm tra liệu có tồn tại 1 cặp đường thẳng cắt nhau hay không.





Gọi ptdt d_1 có dạng : $a_1 * x + b_1 * y + c = 0$ d_2 có dạng : $a_2 * x + b_2 * y + c = 0$.
2 đường thẳng $d_1 d_2 \iff \mathrm{a}_1 * b_2 \neq a_2 * b_1$.
Tính toán số lượng đường thẳng cần thêm và đưa ra câu trả lời.

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

int W, N;

struct Line
{
    int a, b, c;
    Line() {}
    Line(int x1, int y1, int x2, int y2)
    {
        int u = y1 - y2;
        int v = x2 - x1;
        this -> a = u;
        this -> b = v;
        this -> c = -u * x1 - v * y1;
    }

    bool operator==(const Line &rhs) const {
        return a * rhs.b == rhs.a * b &&
               a * rhs.c == rhs.c * a;
    }

    bool operator!=(const Line &rhs) const {
        return !(rhs == *this);
    }

    bool is_parallel(const Line &rhs) const
```

```
    {
        return (a * rhs.b == b * rhs.a);
    }
};

signed main()
{
    ios::sync_with_stdio(0); cin.tie(0);

    cin >> W >> N;
    vector<array<int, 4>> v;
    bool all_parallel = true;
    for (int i = 0; i < N; i++)
    {
        int x1, y1, x2, y2;
        cin >> x1 >> y1 >> x2 >> y2;
        bool oke = true;
        for (int j = 0; j < v.size(); j++)
            if (Line(x1, y1, v[j][0], v[j][1])
            == Line(v[j][0], v[j][1], v[j][2], v[j][3])
            && Line(x2, y2, v[j][0], v[j][1])
            == Line(v[j][0], v[j][1], v[j][2], v[j][3]))
                oke = false;
        if (oke)
            v.push_back(array<int, 4> {x1, y1, x2, y2});
        if (!Line(x1, y1, x2, y2).is_parallel
        (Line(v[0][0], v[0][1], v[0][2], v[0][3])))
            all_parallel = false;
    }
    int cnt = 0, add = 0;
    if (all_parallel)
    {
        cnt = v.size() + 1;
        add = cnt;
    }
    else
    {
        cnt = 2 * v.size();
        add = 2;
    }
    int ans = 0;
    while (cnt < W)
    {
        ++ans;
        cnt += add;
        add = 2;
    };
    cout << ans << "\n";
    return 0;
}
```