E. Rectangle Land

PROBLEM DESCRIPTION

Given a list of rectangles, calculate the total area covered by these rectangles.

SOLUTION TECHINQUES

Sweeping Lines / Discrete Segment Tree

SOLUTION SKETCHES

This is a somewhat known application with Sweeping Lines algorithm.

The top-level idea is that we use the x axis as our time frame and use a data structure (Segment Tree in this case) on the y axis to maintain the area and number of layers covered.

Each rectangle is represented by 2 individual events.
Suppose the rectangle is denoted by (x1, y1) and (x2, y2),

1. Starting from x1, [y1, y2] is covered with 1 additional layer.
2. Ending in x2, [y1, y2] is covered with 1 less layer.

We increase x incrementally (ie. Sweep from left to right) and sum up the areas.

TIME COMPLEXITY

O($N \log N$)

SOLUTION PROGRAM FOR REFERENCE

```cpp
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <vector>
#include <map>
#include <queue>

using namespace std;
typedef long long ll;

const int N = 1e6 + 5;
const int X = 2e6 + 5;
const int S = 1e6;

vector<int> x, y;
int lx[X], ly[X];

struct node
{
    int mx, mxs, cov;
} s[N << 3];

struct event
{
    int x;
    int y1, y2;
    int u; // 1 for add, 0 for substract
    bool operator < (const event &rhs)const
    {
        if (x != rhs.x)
            return x < rhs.x;
        else
            return u > rhs.u;
    }
};
vector<event> a, m;

void build(int i, int l, int r)
{
    s[i].mx = s[i].cov = 0;
    s[i].mxs = y[r] - y[l]; // change
    if (r - l == 1) return ;
    int m = (l + r) >> 1;
    build(i << 1, l, m);
    build(i << 1 | 1, m, r);
}

int upd(int i, int l, int r, int ql, int qr, int u)
{
    if (qr <= l || ql >= r) return s[i].mx + s[i].cov;
    if (ql <= l && r <= qr)
    {
        s[i].cov += u;
        return s[i].mx + s[i].cov;
    }
    int m = (l + r) >> 1;
    s[i].mx = max(upd(i << 1, l, m, ql, qr, u), upd(i << 1 | 1, m, r, ql, qr, u));
```

```
58.     s[i].mxs = s[i << 1].mx + s[i << 1].cov == s[i].mx ? s[i << 1].mxs : 0;
59.     s[i].mxs += s[i<<1|1].mx + s[i<<1|1].cov == s[i].mx ? s[i<<1|1].mxs : 0;
60.     return s[i].mx + s[i].cov;
61. }
62.
63. int main()
64. {
65.     int i, tt, n, x1, y1, x2, y2, c;
66.     int xx, ai, mi;
67.     ll ans, anss;
68.     scanf("%d", &tt);
69.     while (tt--)
70.     {
71.         scanf("%d", &n);
72.         x.clear(); x.reserve(n << 1);
73.         y.clear(); y.reserve(n << 1);
74.         a.clear(); a.reserve(n << 1);
75.         m.clear(); m.reserve(n << 1);
76.         ai = mi = 0;
77.         ans = 0;
78.         for (i = 0; i < n; i++)
79.         {
80.             scanf("%d%d%d%d%d", &x1, &y1, &x2, &y2, &c);
81.             x.push_back(x1); x.push_back(x2);
82.             y.push_back(y1); y.push_back(y2);
83.             if (x1 > x2) swap(x1, x2);
84.             if (y1 > y2) swap(y1, y2);
85.             a.push_back( (event){x1, y1, y2, c} );
86.             m.push_back( (event){x2, y1, y2, -c} );
87.         }
88.         sort(x.begin(), x.end());
89.         sort(y.begin(), y.end());
90.         unique(x.begin(), x.end());
91.         unique(y.begin(), y.end());
92.         for (i = 0; i < x.size(); i++)
93.             lx[x[i] + S] = i;
94.         for (i = 0; i < y.size(); i++)
95.             ly[y[i] + S] = i;
96.         sort(a.begin(), a.end());
97.         sort(m.begin(), m.end());
98.         build(1, 0, y.size() - 1);
99.         for (xx = 1; xx < x.size(); xx++)
100.                {
101.                    while (ai < a.size() && lx[a[ai].x + S] < xx)
102.                    {
103.                        upd(1, 0, y.size() - 1, ly[a[ai].y1 + S], ly[a[ai].y2 + S], a[ai
    ].u);
104.                        ai++;
105.                    }
106.                    while (mi < m.size() && lx[m[mi].x + S] < xx)
107.                    {
108.                        upd(1, 0, y.size() - 1, ly[m[mi].y1 + S], ly[m[mi].y2 + S], m[mi
    ].u);
109.                        mi++;
110.                    }
111.                    if (s[1].mx + s[1].cov > ans)
112.                    {
113.                        ans = s[1].mx + s[1].cov;
114.                        anss = (ll)s[1].mxs * (x[xx] - x[xx - 1]);
115.                    }
116.                    else if (s[1].mx + s[1].cov == ans)
```

```
                    anss += (ll)s[1].mxs * (x[xx] - x[xx - 1]);
                }
                printf("%lld %lld\n", ans, anss);
            }
            return 0;
        }
```