集训队互测 左蓝右红 解题报告

北大附中 谢梓涵

2023 年 10 月 18 日

目录

1 题目大意

1.1 题目描述

在平面直角坐标系上给定 n 个矩形，形成若干个交点，每个矩形的所有边都平行于坐标轴。**保证没有两个矩形的两条边共线。**

矩形之间会形成若干个交点，每一个矩形都被它上面的交点分成若干段。定义属于一个矩形上的**一段**是该矩形上相邻的两个交点之间的部分。也就是说一个矩形上如果有 n 个交点，就会被划分成 n 段。

现在将每一个矩形上的每一段染色为蓝色或红色。要求：

- 与同一个交点相邻的**同属于一个矩形的**两段不同色；
- 所有红线形成若干互不相交的封闭曲线；
- 所有蓝线形成若干互不相交的封闭曲线。

定义两个图形 R, B：

- R 为删去所有蓝线后，所有被**奇数个**红色封闭曲线包含的点组成的集合；
- B 为删去所有红线后，所有被**奇数个**蓝色封闭曲线包含的点组成的集合。

一个方案合法当且仅当它满足上述所有条件，并且**使得 $R \cap B$ 里面只有有限个点**。

求字典序最小的合法染色方案，和合法染色方案数对大质数取模的值。

1.2 样例 1

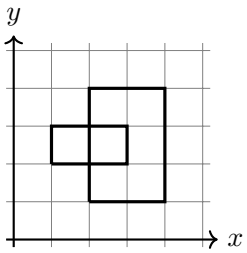
1.2.1 样例输入 1

```
2
1 2 3 3
2 1 4 4
```

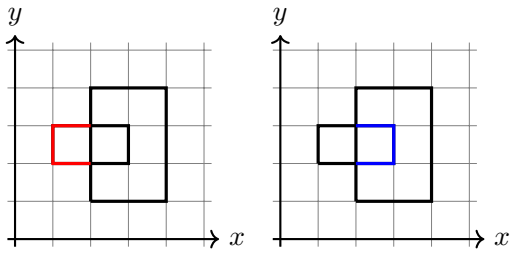
1.2.2 样例输出 1

```
01
2
```

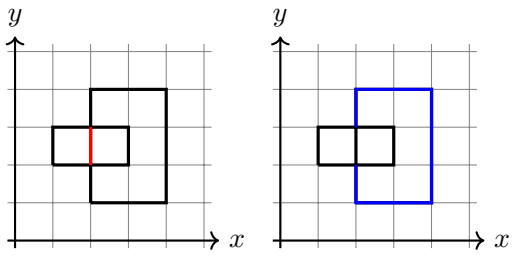
1.2.3 样例 1 解释



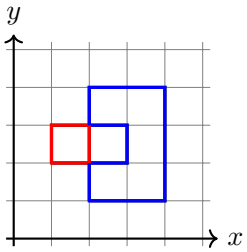
如图是给定的矩形在平面直角坐标系上的形态。



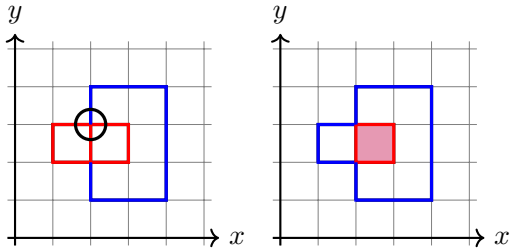
如图，标出颜色的地方是属于第 1 个矩形的两段。



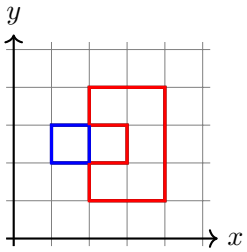
如图，标出颜色的地方是属于第 2 个矩形的两段。按照上述图片给出的颜色进行染色可以得到样例输出中给出的合法方案：



如下图，左侧方案是不合法的，因为与用圈标出的交点相邻的同属于一个矩形的两段同色；右侧方案也是不合法的，因为 R 与 B 交在图中的紫色正方形内，$R \cap B$ 包含无限多个点。



如下方案是合法的，但是对应字符串 10 不是字典序最小的。这种方案和样例里面给出的方案是全部的 2 种方案。



1.3 样例 2

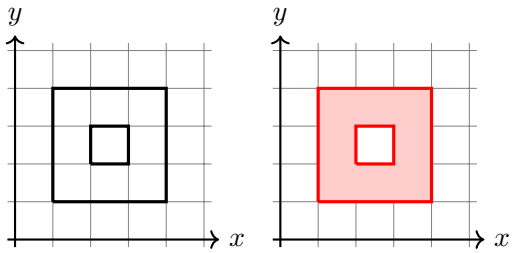
1.3.1 样例输入 2

```
2
1 1 4 4
2 2 3 3
```

1.3.2 样例输出 2

```
00
2
```

1.3.3 样例 2 解释



如图，该样例中，因为环内部的部分被 2 个非负整数包含，2 是偶数，所以环内部不属于 R，R 形成一个环。

1.4 数据范围

$1 \leq n \leq 2 \times 10^5$，矩形的两维坐标均处于 $[1, 2n]$ 范围内。

2 解题过程

2.1 结论和算法框架

以下为了方便，将属于 R 的部分称作是红色的，属于 B 的部分称作是蓝色的，既不属于 R 也不属于 B 的部分称作是无色的。

结论 1：「与同一个交点相邻的同属于一个矩形的两段不同色」是「所有红线形成若干互不相交的封闭曲线」和「所有红线形成若干互不相交的封闭曲线」的充分条件。

结论 1 证明：

考虑所有矩形之间的交点，矩形上的每**一段**连接两个交点。

由此我们可以建两个无向图 G_R, G_B，顶点对应矩形之间的交点，连接两个顶点的边对应连接两个交点的**一段**。其中 G_R 只包含红色段对应的边，G_B 只包含蓝色段对应的边。

显然一个交点恰好与 4 个**段**相邻，所以「与同一个交点相邻的同属于一个矩形的两段不同色」保证与之相邻的 4 **段**恰好有 2 **段**是红色的，2 **段**是蓝色的。

从而 G_R, G_B 中任意顶点的度数均为 2，即 G_R, G_B 分别构成若干相离的简单环。又由于任意两**段**只可能交在矩形之间的交点上，所以对应回平面图形，红线和蓝线各自形成若干条相离的封闭曲线。从而结论 1 得证。

在此基础上，我们可以对平面进行洪水填充，将其划分为块，如图所示：

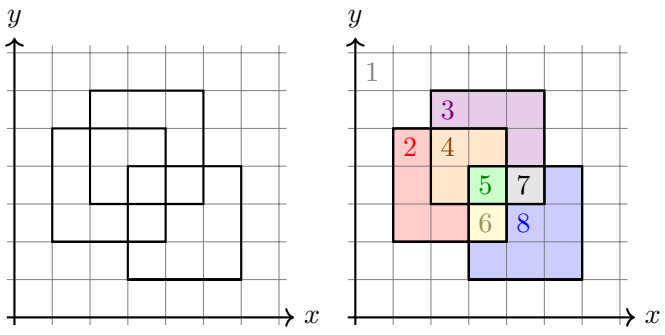


图 1. 左图为题目中的样例 4，右图为洪水填充的结果，共 8 个块。

结论 2：如果将整个平面用洪水填充的方式划分为块，那么点、边相邻的两块不可能同色（点相邻的两块可能都无色）。

结论 2 证明：

对点相邻的情况反证：如果有两个块点相邻并且同色，那么一定有一个交点的 4 条出边都是一个颜色，矛盾。故点相邻的情况得证。

如果两个块边相邻，那么包含这两个块中间的边的封闭曲线一定使得一侧被一个颜色的封闭曲线覆盖的次数比另一侧多 1。又由于所有矩形任意两条边不共线，所以两侧被该颜色覆盖次数奇偶性必然不同，因此按照定义，不可能都是同一个颜色，也不可能都无色。

结论 3：如果将整个平面用洪水填充的方式划分为块，那么边相邻的两块恰好有一个无色。

结论 3 证明：

反证，由结论 2，我们已经得知边相邻的两块不能同色也不能都无色，所以如果边相邻的两块不是恰好有一个无色，那么必然是一红一蓝。此时由定义，这条边同时是红色和蓝色，矛盾。从而结论 3 得证。

到这里，我们考虑扫描线算法。

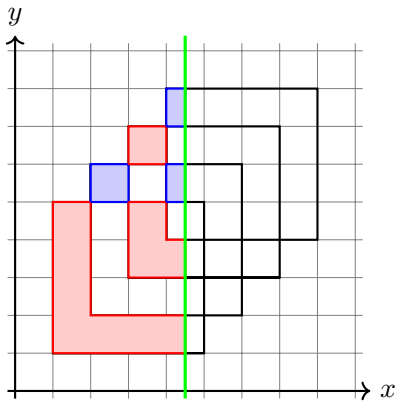


图 2. 扫描线的例子。

由结论 3 和扫描线本身的性质可知，与扫描线相交的部分一定是偶数条横线，且从上到下第 $2k-1$ 条横线和第 $2k$ 条横线中间的区域有颜色。

接下来我们考虑扫描线遇到一个矩形的左侧的边的时候应该如何处理。

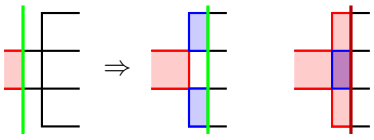


图 3. 扫描线遇到矩形左侧边的例子。以下使用绿色线表示状态合法，深红色线表示状态不合法。

由图 3 可以看出，如果这个矩形的左侧边截到 R，那么左侧边在对应的一段必须是红色以将 R 截断；否则 R 将会散开到相邻的三段区域，而中

间的一对线必然是蓝色，因此这相邻的三段区域里面必定会有一段属于 B，所以一定会交起来，出现无穷多个交点。同理，如果截到 B，那么对应的一段也必须是蓝色以将 B 截断。

同时我们发现加入左侧边时，两条颜色带之间一定间隔恰好一段线，所以两条颜色带的颜色必须是一样的。如图：

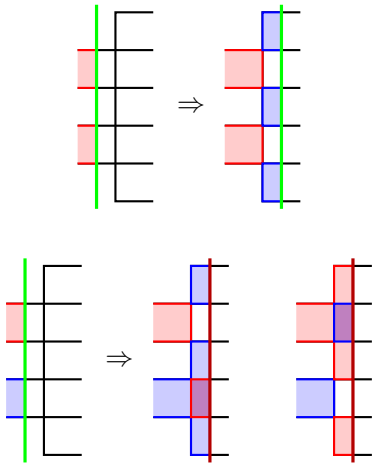


图 4. 矩形左侧边截到多条颜色带的例子。截到两条颜色不同的颜色带时不可能得到合法方案。

因此在截断若干条颜色带时，我们要求这些颜色带的颜色都是相同的。

考虑到在一些情况下我们无法在加入矩形的时候立刻确定其左下角颜色（比如矩形的左侧边完全处于无色区域内的情况），我们需要在扫描线过程中将每一个矩形的左下角颜色设为一个取值范围为 $\{0,1\}$ 的未知数，通过扫描线的过程可以得到若干个变量之间相等或不等关系，并在扫描线结束后通过这些关系解出所有变量的值。这一部分是一个经典问题，可以使用拓展域并查集解决。

根据上面的讨论，每次遇到矩形左侧边时需要将它截到的所有线的颜色全部改为相同的一个未知数。这相当于在一个序列上进行一个区间推平操作，我们可以在这里看到使用基于颜色段数均摊的暴力数据结构（即所谓的 Old Driver Tree）维护的可能性。

扫描线遇到矩形右侧边的情况和左侧边是几乎一致的，区别仅在于右侧边的两个端点附近的分类讨论。这一部分较为平凡，这里不再赘述。最终的形式也是将矩形右侧边所截到的横线的颜色修改为 $O(1)$ 段相同的未知数。

从而我们可以在扫描线的过程中使用 Old Driver Tree 维护所有横线的颜色所对应的未知数，并使用拓展域并查集维护在扫描线过程中产生的未

知数之间的关系。我们在 Old Driver Tree 上推平一个区间 $[l,r]$ 的时候按照经典用法操作：分裂区间两端的连续段，暴力遍历这个区间内所有的连续段，将这些段对应的未知数与当前矩形对应的未知数在并查集上合并，然后删除这些段，加入 $[l,r]$ 一整段，对应当前矩形对应的未知数。这样由颜色段数均摊的结论，每一次操作只会增加 1 段，总段数是 $O(n)$ 量级，使用 C++ 中的 std::set 等平衡二叉搜索树维护连续段即可在 $O(n\log n)$ 时间内完成扫描线过程。

如果在扫描线过程中未知数之间的关系出现矛盾，则直接输出无解。否则在完成扫描线之后，可以通过在拓展域并查集上从前往后贪心来构造出字典序最小的方案。染色的方案数也可以简单求出：设拓展域并查集最后形成了 c 个连通块，则染色方案数就是 $2^{c/2}$。

总时间复杂度 $O(n\log n)$，空间复杂度 $O(n)$。

2.2 参考代码实现

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

const int N = 200005;
const long long mod = 20051131;

int n, xl[N], xr[N], yl[N], yr[N], idx[2 * N], f[2 * N], rev[2 * N], c[2 * N],
    vis[2 * N];

inline long long Power(long long x, long long y) {
    long long ans = 1;
    while (y) {
        if (y & 1) ans = ans * x % mod;
        x = x * x % mod;
        y >>= 1;
    }
    return ans;
}

inline void Quit() {
    cout << "-1" << endl << "0" << endl;
    exit(0);
}

inline int Lowbit(int x) {
    return x & -x;
}

inline void Update(int i, int x) {
    for (int j = i;j <= 2 * n;j += Lowbit(j)) c[j] += x;
}

```

```
32 inline int Query(int i) {
33     int ans = 0;
34     for (int j = i;j >= 1;j -= Lowbit(j)) ans += c[j];
35     return ans;
36 }
37
38 inline void Init() {
39     for (int i = 1;i <= 2 * n;i++) f[i] = i;
40 }
41
42 inline int GetRoot(int v) {
43     return (f[v] == v ? v : f[v] = GetRoot(f[v]));
44 }
45
46 inline void Merge(int x, int y) {
47     int u = GetRoot(x), v = GetRoot(y);
48     f[v] = u;
49 }
50
51 inline void Join(int x, int y, bool eq) {
52     if (!x || !y) return;
53     if (eq) {
54         Merge(x, y);
55         Merge(rev[x], rev[y]);
56     } else {
57         Merge(x, rev[y]);
58         Merge(rev[x], y);
59     }
60     if (GetRoot(x) == GetRoot(rev[x]) || GetRoot(y) == GetRoot(rev[y])) Quit();
61 }
62
63 struct Node {
64     mutable int l, r, idx;
65     bool operator < (const Node& b) const {return l < b.l;}
```

```
66      Node() {}
67      Node(int l, int r, int idx) : l(l), r(r), idx(idx) {}
68  };
69
70  #define ITR set<Node>::iterator
71
72  set <Node> cst;
73  set <int> pst;
74
75  inline ITR Split(int p) {
76      ITR it = --(cst.upper_bound(Node(p, 0, 0))), res;
77      if (it->l == p) res = it;
78      else {
79          int l = it->l, r = it->r, v = it->idx;
80          cst.erase(it); cst.insert(Node(l, p, v)); res = cst.insert(Node(p, r, v)
                ).first;
81      }
82      return res;
83  }
84
85  inline ITR Single(int p) {
86      int nxt = *(pst.upper_bound(p));
87      if (nxt <= 2 * n) Split(nxt); return Split(p);
88  }
89
90  inline ITR Ins(int p, int col) {
91      pst.insert(p); ITR it = Single(p); it->idx = col;
92      Update(p, 1); return it;
93  }
94
95  inline void addLine(int yl, int yr, int cidx) {
96      int cnt1 = Query(yl), cnt2 = Query(yr);
97      int coll = cidx, colm = (cnt1 & 1 ? rev[cidx] : cidx), colr = (cnt2 - cnt1 &
            1 ? rev[cidx] : cidx);
```

```
98      ITR il = Ins(yl, coll), ir = Ins(yr, colr), tmp; tmp = il = Split(yl);
99      for (ITR it = ++tmp;it != ir;it++) Join(it->idx, colm, 0);
100     tmp = il;
101     if (cnt1 & 1) Join((--tmp)->idx, cidx, 1);
102     int l = il->r, r = ir->l;
103     if (il != ir) {
104         cst.erase(++il, ir);
105         cst.insert(Node(l, r, colm));
106     }
107 }
108
109 inline void rmvLine(int yl, int yr) {
110     int cnt1 = Query(yl) - 1 & 1, cnt2 = Query(yr) & 1; Update(yl, -1); Update(
            yr, -1);
111     ITR il = Single(yl), ir = Single(yr), tmp; tmp = il = Split(yl); tmp++;
112     int coll = il->idx, colm = (cnt1 & 1 ? coll : rev[coll]);
113     for (ITR it = tmp;it != ir;it++) Join(it->idx, colm, 0);
114     int l = il->r, r = ir->r;
115     if (ir->l != l) {
116         Node exl = *(--il); exl.r = l;
117         cst.erase(il, ++ir);
118         cst.insert(Node(l, r, colm));
119         cst.insert(exl);
120     } else {
121         Node exl = *(--il); exl.r = r;
122         cst.erase(il, ++ir);
123         cst.insert(exl);
124     }
125     pst.erase(yl); pst.erase(yr);
126 }
127
128 inline void Read() {
129     cin >> n;
130     for (int i = 1;i <= n;i++) cin >> xl[i] >> yl[i] >> xr[i] >> yr[i];
```

```
}

inline void Solve() {
    Init();
    for (int i = 1;i <= n;i++) {
        rev[i] = i + n; rev[i + n] = i;
    }
    for (int i = 1;i <= 2 * n;i++) idx[i] = i;
    sort(idx + 1, idx + 2 * n + 1, [&](const int &x, const int &y) {
        return (x <= n ? xl[x] : xr[x - n]) < (y <= n ? xl[y] : xr[y - n]);
    });
    pst.insert(0); pst.insert(2 * n + 1); pst.insert(2 * n + 2); cst.insert(Node
        (0, 2 * n + 1, 0)); cst.insert(Node(2 * n + 1, 2 * n + 2, 0));
    for (int i = 1;i <= 2 * n;i++) {
        if (idx[i] <= n) addLine(yl[idx[i]], yr[idx[i]], idx[i]);
        else rmvLine(yl[idx[i] - n], yr[idx[i] - n]);
    }
    for (int i = 1;i <= n;i++) {
        if (!vis[GetRoot(i)]) {
            vis[GetRoot(i)] = 1;
            vis[GetRoot(i + n)] = -1;
            cout << 0;
        } else if (vis[GetRoot(i)] == 1) cout << 0;
        else cout << 1;
    }
    cout << endl;
    int tot = 0;
    for (int i = 1;i <= 2 * n;i++) tot += (GetRoot(i) == i);
    cout << (Power(2, tot / 2) % mod + mod) % mod << endl;
}

int main() {
    std::ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    Read();
```

```
164     Solve();
165     return 0;
166 }
```

3 参考资料

本题部分改编自 USACO 2022 February Contest, Platinum 的第 1 题 *Paint by Rectangles*。