题意

求矩形的面积并

思路

由于矩形的位置可以多变，因此矩形的面积一下子不好求

可以采用“分割”的思想，即把整块的矩形面积分割成几个小矩形的面积，然后求和就行了。

把每个矩形投影到y坐标轴上来，枚举矩形的 x 坐标，然后检测当前相邻x坐标上y方向的合法长度，两种相乘就是面积。

如何用线段树来维护那个 “合法长度” 线段树的节点这样定义
struct node {
int left,right,cov;
double len;
}
cov 表示当前节点区间是否被覆盖，len 是当前区间的合法长度

然后我们通过“扫描线”的方法来进行扫描：

- 枚举 x 的竖边，矩形的左边那条竖边就是入边，右边那条就是出边了。
- 把所有这些竖边按照 x 坐标递增排序，每次进行插入操作，由于坐标不一定为整数，因此需要进行离散化处理。
- 每次插入时如果当前区间被完全覆盖，那么就要对 cov 域进行更新。
- 入边 +1 出边 -1，更新完毕后判断当前节点的 cov 域是否大于 0 ，如果大于 0，那么当前节点的 len 域就是
  节点所覆盖的区间。否则，如果是叶子节点，则 len=0。 如果内部节点，则 len=左右儿子的 len 之和。

代码

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

```
using namespace std;
typedef long long LL;
#define L(x) (x<<1)
#define R(x) (x<<1|1)
double y[1000];
struct Line{
    double x,y1,y2;
    int flag;
}line[300];

struct Node{
    int l,r,cover;
    double lf,rf,len;
}node[1000];

bool cmp(Line a,Line b)
{
    return a.x<b.x;
}
void length(int u)
{
    if(node[u].cover>0)
    {
        node[u].len = node[u].rf - node[u].lf;
        return ;
    }
    else if(node[u].l +1 == node[u].r)
        node[u].len = 0;
    else
        node[u].len = node[L(u)].len + node[R(u)].len;
}

void build (int u,int l,int r)
{
    node[u].l =l;
    node[u].lf = y[l];node[u].rf = y[r];
    node[u].len = node[u].cover = 0;
    if(l+1 == r) return ;
    int mid = (l+r)/2;
    build(L(u),l,mid);
    build(R(u),mid,r);
}

void update(int u,Line e)
{
    if(e.y1 == node[u].lf && e.y2 == node[u].rf)
```

```
48      {
49          node[u] . cover += e.flag;
50          length(u);
51          return ;
52      }
53      if(e.y1>=node[R(u)].lf)
54          update(R(u),e);
55      else if(e.y2 <= node[L(u)].rf)
56          update(L(u),e);
57      else
58      {
59          Line temp = e;
60          temp.y2 = node[L(u)].rf;
61          update(L(u),temp);
62          temp = e;
63          temp.y1 = node[R(u)].lf;
64          update( R(u),temp);
65      }
66      length(u);
67  }
68  int main()
69  {
70      //ios::sync_with_stdio(false);
71      int n,t,i,Case = 0;
72      double x1,y1,x2,y2,ans;
73      while(scanf("%d",&n)&&n)
74      {
75          for(i = t = 1;i<=n;i++,t++)
76          {
77              scanf("%lf%lf%lf%lf",&x1,&y1,&x2,&y2);
78              line[t].x = x1;
79              line[t].y1 = y1;
80              line[t].y2 = y2;
81              line[t].flag = 1;
82              y[t] = y1;
83              t++;
84              line[t].x = x2;
85              line[t].y1 = y1;
86              line[t].y2 = y2;
87              line[t].flag = -1;
88              y[t] = y2;
89          }
90          sort(line+1,line+t,cmp);
91          sort(y+1,y+t);
92          build(1,1,t-1);
93          update(1,line[1]);
```

```
94          ans = 0;
95          for(i=2;i<t;i++)
96          {
97              ans += node[1].len * (line[i].x - line[i-1].x);
98              update(1,line[i]);
99          }
100         printf("Test case #%d\n",++Case);
101         printf("Total explored area: %.2f\n\n",ans);
102     }
103     return 0;
104 }
105
106
```