```cpp

#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<string>
#include<queue>
#include<algorithm>
#include<map>
#include<iomanip>
#define INF 99999999
using namespace std;

const int MAX = 200 + 10;
int mark[MAX << 2];//记录某个区间的下底边个数
double sum[MAX << 2];//记录某个区间的下底边总长度
double Hash[MAX];//对x进行离散化,否则x为浮点数且很大无法进行线段树

//以横坐标作为线段(区间),对横坐标线段进行扫描
//扫描的作用是每次更新下底边总长度和下底边个数,增加新面积
struct seg {//线段
    double l, r, h;
    int d;
    seg() {}
    seg(double x1, double x2, double H, int c) :l(x1), r(x2), h(H), d(c) {}
    bool operator<(const seg& a)const {
        return h < a.h;
    }
}s[MAX];

void Upfather(int n, int left, int right) {
    if (mark[n])sum[n] = Hash[right + 1] - Hash[left];//表示该区间整个线段长度可以作为底边
    else if (left == right)sum[n] = 0;//叶子结点则底边长度为0(区间内线段长度为0)
    else sum[n] = sum[n << 1] + sum[n << 1 | 1];
}

void Update(int L, int R, int d, int n, int left, int right) {
    if (L <= left && right <= R) {//该区间是当前扫描线段的一部分,则该区间下底边总长以及上下底边个数差更新
        mark[n] += d;//更新底边相差差个数
        Upfather(n, left, right);//更新底边长
        return;
    }
    int mid = left + right >> 1;
    if (L <= mid)Update(L, R, d, n << 1, left, mid);
    if (R > mid)Update(L, R, d, n << 1 | 1, mid + 1, right);
    Upfather(n, left, right);
}

int search(double key, double* x, int n) {
    int left = 0, right = n - 1;
    while (left <= right) {
        int mid = left + right >> 1;
        if (x[mid] == key)return mid;
```

```
54          if (x[mid] > key)right = mid - 1;
55          else left = mid + 1;
56      }
57      return -1;
58  }
59
60  int main() {
61      int n, num = 0;
62      double x1, x2, y1, y2;
63      while (cin >> n, n) {
64          int k = 0;
65          for (int i = 0; i < n; ++i) {
66              cin >> x1 >> y1 >> x2 >> y2;
67              Hash[k] = x1;
68              s[k++] = seg(x1, x2, y1, 1);
69              Hash[k] = x2;
70              s[k++] = seg(x1, x2, y2, -1);
71          }
72          sort(Hash, Hash + k);
73          sort(s, s + k);
74          int m = 1;
75          for (int i = 1; i < k; ++i)//去重复端点
76              if (Hash[i] != Hash[i - 1])Hash[m++] = Hash[i];
77          double ans = 0;
78          //memset(mark,0,sizeof mark);
79          //memset(sum,0,sizeof sum);如果下面是i<k-1则要初始化,因为如果对第k-1条
              线段扫描时会使得mark,sum为0才不用初始化的
80          for (int i = 0; i < k; ++i) {//扫描线段
81              int L = search(s[i].l, Hash, m);
82              int R = search(s[i].r, Hash, m) - 1;
83              Update(L, R, s[i].d, 1, 0, m - 1);//扫描线段时更新底边长度和底边相
                  差个数
84              ans += sum[1] * (s[i + 1].h - s[i].h);//新增加面积
85          }
86          printf("Test case #%d\nTotal explored area: %.2lf\n\n", ++num, ans);
87      }
88      return 0;
89  }
90  /*
91  这里注意下扫描线段时r-1:int R=search(s[i].l,hash,m)-1;
92  计算底边长时r+1:if(mark[n])sum[n]=hash[right+1]-hash[left];
93  解释:假设现在有一个线段左端点是l=0,右端点是r=m-1
94  则我们去更新的时候,会算到sum[1]=hash[mid]-hash[left]+hash[right]-hash[mid+1]
95  这样的到的底边长sum是错误的,why?因为少算了mid~mid+1的距离,由于我们这利用了
96  离散化且区间表示线段,所以mid~mid+1之间是有长度的,比如hash[3]=1.2,hash[4]
      =5.6,mid=3
97  所以这里用r-1,r+1就很好理解了
98  */
```