OJ 11265

The Sultan's Problem

Prof. Edson Alves

Faculdade UnB Gama

OJ 11265 – The Sultan's Problem

Problema

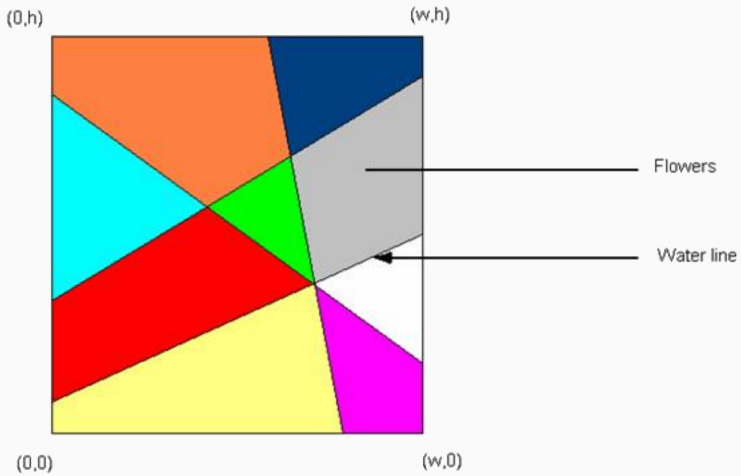
Once upon a time, there lived a great sultan, who was very much fond of his wife. He wanted to build a Tajmahal for his wife (ya, our sultan idolized Mughal emperor Shahjahan). But alas, due to budget cuts, loans, dues and many many things, he had no fund to build something so big. So, he decided to build a beautiful garden, inside his palace.

The garden is a rectangular piece of land. All the palaces water lines run through the garden, thus, dividing it into pieces of varying shapes. He wanted to cover each piece of the land with flowers of same kind.

The figure above shows a sample flower garden. All the lines are straight lines, with two end points on two different edge of the rectangle. Each piece of the garden is covered with the same kind of flowers.

The garden has a small fountain, located at position (x, y). You can assume that, it is not on any of the lines. He wants to fill that piece with her favourite flower. So, he asks you to find the area of that piece.

Problema



Entrada e saída

Input

Input contains around 500 test cases. Each case starts with 5 integers, N, W, H, x, y, the number of lines, width and height of the garden and the location of the fountain. Next N lines each contain 4 integers, $x_1\ y_1\ x_2\ y_2$, the two end points of the line. The end points are always on the boundary of the garden. You can assume that, the fountain is not located on any line.

Constraints

- $1 \leq W, H \leq 200,000$
- $1 \leq N \leq 500$

Output

For each test case, output the case number, with the area of the piece covered with her favourite flower. Print 3 digits after the decimal point.

Exemplo de entradas e saídas

Sample Input

```
3 100 100 20 30
0 0 100 100
100 0 0 100
0 40 40 100
```

Sample Output

```
Case #1: 1780.000
```

Solução

- A área que contém a fonte F deve ser obtida através de sucessivos cortes do polígono P que representa o jardim

Solução

- A área que contém a fonte F deve ser obtida através de sucessivos cortes do polígono P que representa o jardim
- Inicialmente P é um retângulo

Solução

- A área que contém a fonte F deve ser obtida através de sucessivos cortes do polígono P que representa o jardim
- Inicialmente P é um retângulo
- Cada nova reta fará um corte em P, separando a área que contém F do restante

Solução

- A área que contém a fonte F deve ser obtida através de sucessivos cortes do polígono P que representa o jardim
- Inicialmente P é um retângulo
- Cada nova reta fará um corte em P, separando a área que contém F do restante
- A orientação do corte é importante: se F está à esquerda dos pontos A e B da reta, o corte é feito na orientação AB

Solução

- A área que contém a fonte F deve ser obtida através de sucessivos cortes do polígono P que representa o jardim
- Inicialmente P é um retângulo
- Cada nova reta fará um corte em P, separando a área que contém F do restante
- A orientação do corte é importante: se F está à esquerda dos pontos A e B da reta, o corte é feito na orientação AB
- Se estiver à direita, a orientação é feita no sentido oposto BA

Solução

- A área que contém a fonte F deve ser obtida através de sucessivos cortes do polígono P que representa o jardim
- Inicialmente P é um retângulo
- Cada nova reta fará um corte em P, separando a área que contém F do restante
- A orientação do corte é importante: se F está à esquerda dos pontos A e B da reta, o corte é feito na orientação AB
- Se estiver à direita, a orientação é feita no sentido oposto BA
- Cada corte tem complexidade $O(n)$, onde n é o número de vértices do polígono

Solução

- A área que contém a fonte F deve ser obtida através de sucessivos cortes do polígono P que representa o jardim
- Inicialmente P é um retângulo
- Cada nova reta fará um corte em P, separando a área que contém F do restante
- A orientação do corte é importante: se F está à esquerda dos pontos A e B da reta, o corte é feito na orientação AB
- Se estiver à direita, a orientação é feita no sentido oposto BA
- Cada corte tem complexidade $O(n)$, onde n é o número de vértices do polígono
- No pior caso, a solução tem complexidade $O(N^2)$, onde N é o número retas

Solução AC

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

using namespace std;

const double EPS { 1e-6 };

struct Point {
    double x, y;

    Point(double xv = 0, double yv = 0) : x(xv), y(yv) {}

    double distance(const Point& P) const { return hypot(x - P.x, y - P.y); }

    bool operator==(const Point& P) const {
        return fabs(x - P.x) < EPS and fabs(y - P.y) < EPS;
    }
};
```

Solução AC

```
19 struct Polygon {
20     vector<Point> vs;
21     int n;
22 
23     Polygon(const vector<Point>& v) : vs(v), n(v.size()) {
24         vs.push_back(v[0]);
25     }
26 
27     double area() const {
28         double a = 0;
29 
30         for (int i = 0; i < n; ++i)
31         {
32             a += vs[i].x * vs[i+1].y;
33             a -= vs[i+1].x * vs[i].y;
34         }
35 
36         return 0.5 * fabs(a);
37     }
38 };
```

Solução AC

```
40 Point intersection(const Point& P, const Point& Q,
41     const Point& A, const Point& B)
42 {
43     auto a = B.y - A.y;
44     auto b = A.x - B.x;
45     auto c = B.x * A.y - A.x * B.y;
46     auto u = fabs(a * P.x + b * P.y + c);
47     auto v = fabs(a * Q.x + b * Q.y + c);
48
49     return Point((P.x*v + Q.x*u)/(u + v), (P.y*v + Q.y*u)/(u + v));
50 }
51
52 double D(const Point& P, const Point& Q, const Point& R)
53 {
54     return (P.x * Q.y + P.y * R.x + Q.x * R.y)
55         - (R.x * Q.y + R.y * P.x + Q.x * P.y);
56 }
57
58 Polygon cut_polygon(const Polygon& P, const Point& A, const Point& B) {
59     vector<Point> points;
```

Solução AC

```
61      for (int i = 0; i < P.n; ++i)
62      {
63          auto d1 = D(A, B, P.vs[i]);
64          auto d2 = D(A, B, P.vs[i + 1]);
65
66
67          if (d1 > -EPS)
68              points.push_back(P.vs[i]);
69
70          if (d1 * d2 < -EPS)
71              points.push_back(intersection(P.vs[i], P.vs[i+1], A, B));
72      }
73
74      return Polygon(points);
75  }
76
77  int main() {
78      int N, W, H, x, y, test = 0;
```

Solução AC

```
80      while(cin >> N >> W >> H >> x >> y)
81      {
82          Polygon p({ Point(0, 0), Point(W, 0), Point(W, H), Point(0, H) });
83          Point F(x, y);
84
85          while (N--) {
86              Point Q, R;
87              cin >> Q.x >> Q.y >> R.x >> R.y;
88
89              if (D(Q, R, F) > 0)
90                  p = cut_polygon(p, Q, R);
91              else
92                  p = cut_polygon(p, R, Q);
93          }
94
95          printf("Case #%d: %.3f\n", ++test, p.area());
96      }
97      return 0;
98  }
```