Geometria Computacional

Polígonos: Treliças – problemas resolvidos

Prof. Edson Alves

2019

Faculdade UnB Gama

Sumário

UVA 10088 – Trees on My Island

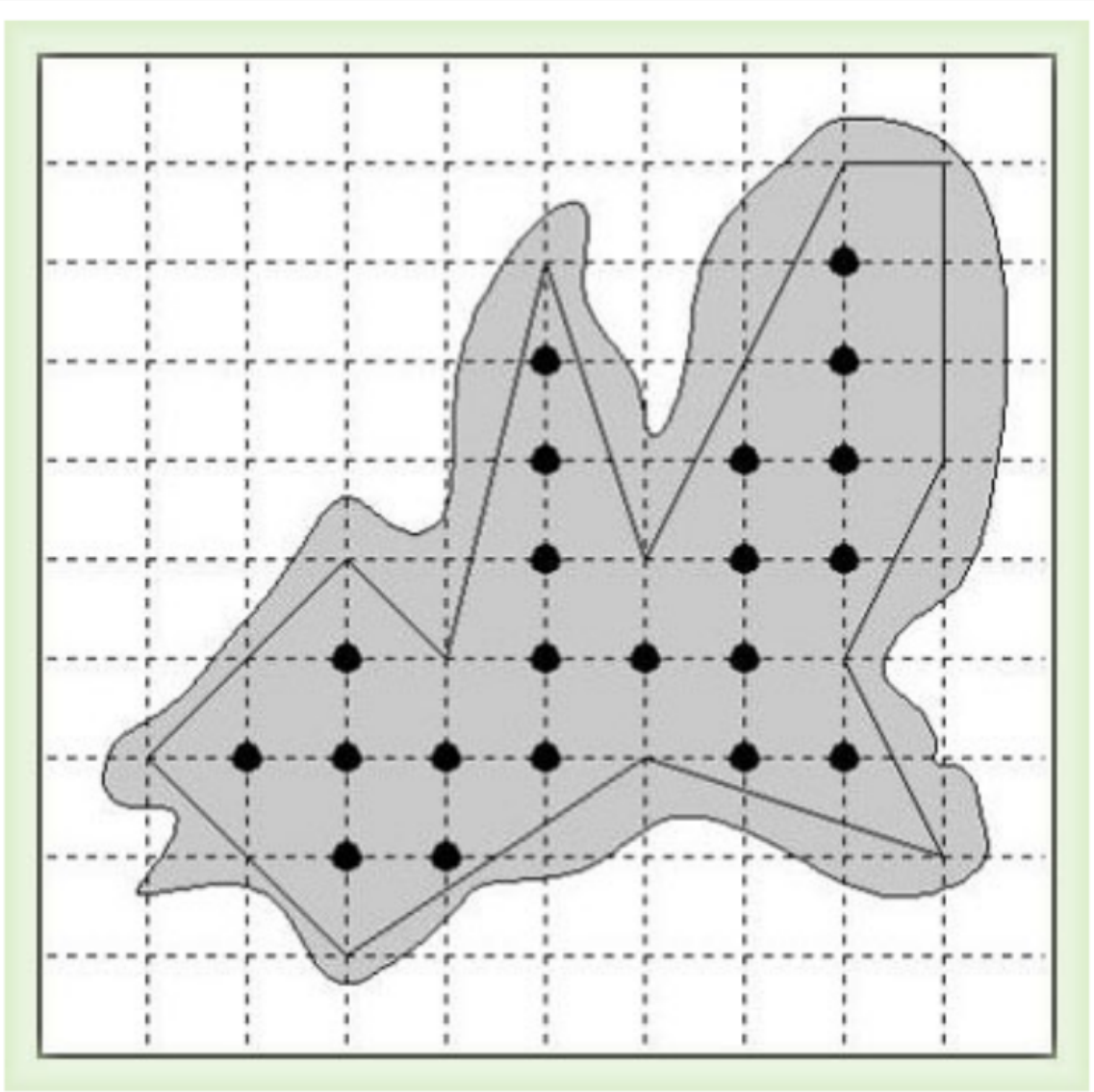
Problema

I have bought an island where I want to plant trees in rows and columns. So, the trees will form a rectangular grid and each of them can be thought of having integer coordinates by taking a suitable grid point as the origin.

But, the problem is that the island itself is not rectangular. So, I have identified a simple polygonal area inside the island with vertices on the grid points and have decided to plant trees on grid points lying strictly inside the polygon.

Now, I seek your help for calculating the number of trees that can be planted on my island.

Problema



Entrada e saída

Input

The input file may contain multiple test cases. Each test case begins with a line containing an integer N $(3 \leq N \leq 1,000)$ identifying the number of vertices of the polygon. The next N lines contain the vertices of the polygon either in clockwise or in anti-clockwise direction. Each of these N lines contains two integers identifying the x and y-coordinates of a vertex. You may assume that none of the coordinates will be larger than $1,000,000$ in absolute values.

A test case containing a zero for N in the first line terminates the input.

Output

For each test case in the input print a line containing the number of trees that can be planted inside the polygon.

Exemplo de entradas e saídas

Sample Input

```
12
3 1
6 3
9 2
8 4
9 6
9 9
8 9
6 5
5 8
4 4
3 5
1 3
12
1000 1000
2000 1000
4000 2000
6000 1000
8000 3000
8000 8000
7000 8000
5000 4000
4000 5000
3000 4000
3000 5000
1000 3000
0
```

Sample Output

```
21
25990001
```

Solução $O(TN)$

- A tentativa de testar todos os pontos dentro do retângulo que delimita o polígono gera um algoritmo $O(N^2)$ que leva ao TLE
- Contudo, este problema pode ser resolvido com complexidade $O(N)$ para cada um dos T casos de teste
- Basta utilizar o Teorema de Pick, que relaciona o número de pontos com coordenadas inteiras que estão no interior (I) e na borda (B) do polígono P com sua área A
- A área A pode ser computada em $O(N)$ a partir das coordenadas de seus vértices, utilizando a expressão

$$A = \frac{1}{2} \left| \sum_{i=1}^{N} x_i y_{i+1} - \sum_{j=1}^{N} y_j x_{j+1} \right|,$$

com $(x_N, y_N) = (x_0, y_0)$

Solução $O(TN)$

- Cada aresta QR de P intercepta $d+1$ pontos com coordenadas inteiras, onde $d = (b, h)$ é o maior divisor comum entre $b = |Q_x - R_x|$ e $h = |Q_y - R_y|$
- Como os vértices são contados duas vezes cada, segue que

$$B = -N + \sum_{i}^{N} \gcd(b_i, h_i),$$

- Assim, pelo Teorema de Pick,

$$2I = 2A - B + 2,$$

  o que permite computar o valor de I a partir de A e B

Solução $O(TN)$

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

using namespace std;
using ll = long long;
using ii = pair<ll, ll>;

#define x first
#define y second

ll gcd(ll a, ll b) { return b ? gcd(b, a % b) : a; }

ll area(int N, const vector<ii>& ps)
{
    ll A = 0;

    for (int i = 0; i < N; ++i)
    {
        A += ps[i].x * ps[i + 1].y;
        A -= ps[i].y * ps[i + 1].x;
    }

```

Solução $O(TN)$

```
22      return llabs(A);
23  }
24
25  // Teorema de Pick: A = I + B/2 - 1
26  ll solve(int N, vector<ii>& ps)
27  {
28      ps.push_back(ps.front());
29
30      ll B = 0;
31
32      for (int i = 0; i < N; ++i)
33      {
34          auto b = llabs(ps[i].x - ps[i + 1].x);
35          auto h = llabs(ps[i].y - ps[i + 1].y);
36          auto d = gcd(b, h);
37
38          B += (d + 1);
39      }
40
41      // Desconta os vértices, contados em duplicidade
42      B -= N;
```

Solução $O(TN)$

```
43
44      auto _2A = area(N, ps);
45      auto I = (_2A - B + 2)/2;
46
47      return I;
48  }
49
50  int main()
51  {
52      ios::sync_with_stdio(false);
53
54      int N;
55
56      while (cin >> N, N)
57      {
58          vector<ii> ps(N);
59
60          for (int i = 0; i < N; ++i)
61              cin >> ps[i].x >> ps[i].y;
62
63          auto ans = solve(N, ps);
```

Solução $O(TN)$

```
64
65          cout << ans << '\n';
66      }
67
68      return 0;
69  }
```

Codechef – Ant Colony

Problema

In an ants' colony spread over a flat surface, the ants can reside at integral coordinates as close as unit distance apart from each other (but no closer). The colony has the shape of a quadrilateral $ABCD$ where one ant must reside at each of A, B, C, and D, and all other ants can be on any integral coordinate on its perimeter and its interior. Given the integral positions of A, B, C, and D, print the maximum number of ants that can reside in the colony (including the ants that can reside on the four corners and the perimeter).

Entrada e saída

Input

The first line contains the number of test cases, N.

For each test case, a single line contains the x and y coordinates of the four corners of the quadrilateral in clockwise order starting with the left-most, bottom-most point.

Output

For each test case, print the case number, followed by a colon, followed by a single space, followed by a single integer indicating the maximum number of ants.

Constraints

$O < N \leq 3$

$|x|, |y| \leq 10,000$

Exemplo de entradas e saídas

Sample Input

```
2
1 1 3 4 5 3 6 1
1 3 5 6 6 3 3 1
```

Sample Output

```
Case 1: 14
Case 2: 16
```

Solução com complexidade $O(N)$

- Este problema é semelhante ao anterior, com pequenas alterações
- A primeira delas é que o polígono a ser avaliado é sempre um quadrilátero
- Outro ponto é a orientação do polígono, que passa a ser fixa
- Além disso, a resposta consiste na soma dos pontos interiores I e dos pontos da borda B
- A saída contém a descrição de cada caso de teste
- Como o número de operações é constante para cada um dos N casos de teste, a solução é $O(N)$

Solução $O(N)$

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

using namespace std;
using ll = long long;
using ii = pair<ll, ll>;

#define x first
#define y second

ll gcd(ll a, ll b) { return b ? gcd(b, a % b) : a; }

ll area(int N, const vector<ii>& ps)
{
    ll A = 0;

    for (int i = 0; i < N; ++i)
    {
        A += ps[i].x * ps[i + 1].y;
        A -= ps[i].y * ps[i + 1].x;
    }

```

Solução $O(N)$

```
22      return llabs(A);
23  }
24
25  // Teorema de Pick: A = I + B/2 - 1
26  ll solve(int N, vector<ii>& ps)
27  {
28      ps.push_back(ps.front());
29
30      ll B = 0;
31
32      for (int i = 0; i < N; ++i)
33      {
34          auto b = llabs(ps[i].x - ps[i + 1].x);
35          auto h = llabs(ps[i].y - ps[i + 1].y);
36          auto d = gcd(b, h);
37
38          B += (d + 1);
39      }
40
41      // Desconta os vértices, contados em duplicidade
42      B -= N;
```

Solução $O(N)$

```
43
44      auto _2A = area(N, ps);
45      auto I = (_2A - B + 2)/2;
46
47      return I + B;
48  }
49
50  int main()
51  {
52      ios::sync_with_stdio(false);
53
54      int N, test = 0;
55      cin >> N;
56
57      while (N--)
58      {
59          vector<ii> ps(4);
60
61          for (int i = 0; i < 4; ++i)
62              cin >> ps[i].x >> ps[i].y;
63
```

Solução $O(N)$

```
64          auto ans = solve(4, ps);
65
66          cout << "Case " << ++test << ": " << ans << '\n';
67      }
68
69      return 0;
70  }
```

Referências

1. Codechef – Ant Colony
2. UVA 10088 – Trees on My Island