# OJ 10088

## Trees on My Island

Prof. Edson Alves

Faculdade UnB Gama

# OJ 10088 – Trees on My Island

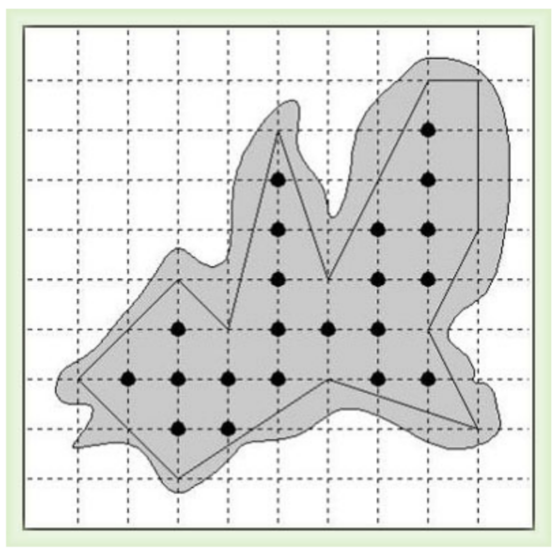
# Problema

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

# 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

# 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 $A$

# 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 $h = |Q_y - R_y|$

## 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

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

# 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 $O(TN)$

```
struct Point { ll x, y; };

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

ll area(int N, const vector<Point>& 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;
    }

    return llabs(A);
}
```

# Solução $O(TN)$

```
// Teorema de Pick: A = I + B/2 - 1
ll solve(int N, vector<Point>& ps)
{
    ps.push_back(ps.front());

    ll B = 0;

    for (int i = 0; i < N; ++i)
    {
        auto b = llabs(ps[i].x - ps[i + 1].x);
        auto h = llabs(ps[i].y - ps[i + 1].y);
        auto d = gcd(b, h);

        B += (d + 1);
    }
```

# Solução $O(TN)$

```
    // Desconta os vértices, contados em duplicidade
    B -= N;

    auto _2A = area(N, ps);
    auto I = (_2A - B + 2)/2;

    return I;
}
```