Geometria Computacional

Retas e Vetores: problemas resolvidos

Prof. Edson Alves

2018

Faculdade UnB Gama

Sumário

Codeforces Beta Round #7 – Problem C: Line

Problema

A line on the plane is described by an equation $Ax + By + C = 0$. You are to find any point on this line, whose coordinates are integer numbers from $-5 \cdot 10^{18}$ to $5 \cdot 10^{18}$ inclusive, or to find out that such points do not exist.

Entrada e saída

Input

The first line contains three integers A, B and C ($-2 \cdot 10^9 \leq A, B, C \leq 2 \cdot 10^9$) – corresponding coefficients of the line equation. It is guaranteed that $A^2 + B^2 > 0$.

Output

If the required point exists, output its coordinates, otherwise output -1.

Exemplo de entradas e saídas

Sample Input

```
2 5 3
```

Sample Output

```
6 -3
```

Observações sobre o problema

- A condição $A^2 + B^2 > 0$ indica que ambos coeficientes não são ambos nulos, de modo que as retas da entrada não são degeneradas
- Os limites do problema impedem uma solução por busca completa (seriam, no pior caso, mais de 10^{19} candidatos para o valor de x
- A equação geral da reta pode ser reescrita como

$$Ax + By = -C$$

- Ainda assim, são duas variáveis para uma única equação. Como proceder neste caso?
- Esta é, na verdade, uma equação diofantina
- Equações diofantinas são equações cujas soluções deve ser inteiras

Equações Diofantinas Lineares

- A equação diofantina linear, com duas variáveis x e y, são as mais comuns, e já foram amplamente estudadas
- Para que tal equação tenha solução, o maior divisor comum $d = (A, B)$ de A e B deve dividir também o coeficiente C
- Para encontrar uma solução, caso exista, deve ser utilizado o algoritmo de Euclides estendido
- Ele decorre do fato de que se $A = Bq + r$, com $0 \leq r < B$, então $d = (A, B) = (B, r)$, que $(A, 0) = |A|$, e que existem x_0, y_0 inteiros tais que $d = Ax_0 + By_0$
- No caso base, $d = |A|, x_0 = \pm 1, y_0 = 0$, onde o sinal de x_0 é igual ao sinal de A
- No caso geral, $Ax_0 + By_0 = Bx_1 + ry_1$, o que nos dá

$$x_0 = y_1, \quad y_0 = x_1 - qy_1,$$

  pois $r = A - Bq$
- Daí $x = kx_0, y = ky_0$, onde $k = -C/d$

Solução AC com complexidade $O(\log(A+B))$

```cpp
#include <iostream>

using ll = long long;

ll ext_gcd(ll a, ll b, ll& x, ll& y)
{
    if (b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }

    ll x1, y1, d = ext_gcd(b, a % b, x1, y1);

    x = y1;
    y = x1 - y1*(a / b);

    return d;
}

```

Solução AC com complexidade $O(\log(A+B))$

```cpp
int main()
{
    ll A, B, C;
    std::cin >> A >> B >> C;

    ll x, y, d = ext_gcd(A, B, x, y);

    if (C % d)
    {
        std::cout << -1 << '\n';
        return 0;
    }

    ll k = -C / d;
    x *= k;
    y *= k;

    std::cout << x << " " << y << '\n';

    return 0;
}
```

Educational Codeforces Round 1 – Problem C: Nearest vectors

Problema

You are given the set of vectors on the plane, each of them starting at the origin. Your task is to find a pair of vectors with the minimal non-oriented angle between them.

Non-oriented angle is non-negative value, minimal between clockwise and counterclockwise direction angles. Non-oriented angle is always between 0 and π. For example, opposite directions vectors have angle equals to π.

Entrada e saída

Input

First line of the input contains a single integer n $(2 \leq n \leq 100000)$ – the number of vectors.

The i-th of the following n lines contains two integers x_i and y_i $(|x|, |y| \leq 10000, x^2 + y^2 > 0)$ – the coordinates of the i-th vector. Vectors are numbered from 1 to n in order of appearing in the input. It is guaranteed that no two vectors in the input share the same direction (but they still can have opposite directions).

Output

If the required point exists, output its coordinates, otherwise output -1.

Exemplo de entradas e saídas

Sample Input

```
4
-1 0
0 -1
1 0
1 1

6
-1 0
0 -1
1 0
1 1
-4 -5
-4 -6
```

Sample Output

```
3 4

6 5
```

Observações sobre o problema

- O ângulo que um vetor faz com o eixo-x positivo pode ser computado com a função `atan2()` da biblioteca de matemática do C/C++
- É preciso observa que o retorno da função está no intervalo $[-\pi, \pi]$
- Uma vez ordenados os vetores por ângulo, basta computar o ângulo entre dois vetores consecutivos usando a diferença
- Por conta do retorno da função `atan2()`, esta diferença pode ser negativa
- Caso isto aconteça, basta somar 2π ao resultado
- O valor de π pode ser computado através da expressão `acos(-1.0)`
- Por fim, é preciso usar o tipo **`long double`**, caso contrário o veredito será WA, por conta da precisão

Solução AC com complexidade $O(n \log n)$

```cpp
#include <algorithm>
#include <iostream>
#include <vector>
#include <cmath>

using ii = std::pair<int, int>;

struct Vector
{
    int x, y, idx;
    long double angle;

    Vector(int xv, int yv, int i)
        : x(xv), y(yv), idx(i), angle(atan2l(y, x)) {}

    bool operator<(const Vector& v) const
    {
        return angle < v.angle;
    }
};

```

Solução AC com complexidade $O(n \log n)$

```
22 ii solve(std::vector<Vector>& vs, int N)
23 {
24     sort(vs.begin(), vs.end());
25     vs.push_back(vs.front());
26
27     long double min_angle = 10.0;
28     ii ans;
29
30     for (int i = 0; i < N; ++i) {
31         auto angle = vs[i + 1].angle - vs[i].angle;
32
33         if (angle < 0) angle += 2*acosl(-1.0);
34
35         if (angle < min_angle) {
36             min_angle = angle;
37             ans = ii(vs[i].idx, vs[i+1].idx);
38         }
39     }
40
41     return ans;
42 }
```

Solução AC com complexidade $O(n \log n)$

```
44 int main()
45 {
46     std::ios::sync_with_stdio(false);
47 
48     int N;
49     std::cin >> N;
50 
51     std::vector<Vector> vs;
52 
53     for (int i = 1; i <= N; ++i)
54     {
55         int x, y;
56         std::cin >> x >> y;
57         vs.push_back(Vector { x, y, i });
58     }
59 
60     auto ans = solve(vs, N);
61     std::cout << ans.first << " " << ans.second << '\n';
62 
63     return 0;
64 }
```

Solução AC usando aritmética inteira estendida

```
1  from math import *
2
3
4  class Vector:
5
6      def __init__(self, x, y, label):
7          self.x = x
8          self.y = y
9          self.label = label
10
11         if x >= 0 and y > 0:
12             self.quad = 1;
13         elif x < 0 and y >= 0:
14             self.quad = 2;
15         elif x <= 0 and y < 0:
16             self.quad = 3;
17         else:
18             self.quad = 4;
19
```

Solução AC usando aritmética inteira estendida

```
20      def __lt__(self, v):
21          if self.quad != v.quad:
22              return self.quad < v.quad
23
24          return self.y * v.x < self.x * v.y
25
26
27  def non_oriented_angle(a, b):
28
29      num = a.x * b.x + a.y * b.y
30      signal = 1
31
32      if num < 0:
33          signal = -1;
34
35      d1 = a.x * a.x + a.y * a.y
36      d2 = b.x * b.x + b.y * b.y
37
38      return (signal * num * num, d1 * d2)
39
40
```

Solução AC usando aritmética inteira estendida

```
41 def solve(vs, N):
42
43     vs.sort()
44     vs.append(vs[0])
45
46     min_angle = (-10, 1)
47     a = -1
48     b = -1
49
50     for i in xrange(N):
51         angle = non_oriented_angle(vs[i], vs[i + 1])
52
53         if angle[0] * min_angle[1] > angle[1] * min_angle[0]:
54             min_angle = angle
55             a = vs[i].label
56             b = vs[i + 1].label
57
58     print '{} {}'.format(a, b)
59
60
```

Solução AC usando aritmética inteira estendida

```
61 if __name__ == '__main__':
62
63     N = int(raw_input())
64
65     vs = []
66
67     for i in xrange(N):
68         x, y = [int(k) for k in raw_input().split()]
69         vs.append(Vector(x, y, i + 1))
70
71     solve(vs, N)
```

Referências

1. Codeforces Beta Round #7 – Problem C: Line
2. Educational Codeforces Round 1 – Problem C: Nearest vectors