Geometria Computacional

Retas – Algoritmos: problemas resolvidos

Prof. Edson Alves

2018

Faculdade UnB Gama

Sumário

URI 1834 – Vogons!

Problema

Os vogons são uma raça alienígena que habita a Vogosfera, segundo o Guia do Mochileiro das Galáxias, escrito por Douglas Adams. Nas palavras do próprio autor:

"Here is what to do if you want to get a lift from a Vogon: forget it. They are one of the most unpleasant races in the Galaxy. Not actually evil, but bad-tempered, bureaucratic, officious and callous. They wouldn't even lift a finger to save their own grandmothers from the Ravenous Bugblatter Beast of Traal without orders - signed in triplicate, sent in, sent back, queried, lost, found, subjected to public inquiry, lost again, and finally buried in soft peat for three months and recycled as firelighters. The best way to get a drink out of a Vogon is to stick your finger down his throat, and the best way to irritate him is to feed his grandmother to the Ravenous Bugblatter Beast of Traal. On no account should you allow a Vogon to read poetry at you".

Problema

No romance, os vogons foram os responsáveis pela destruição da Terra, pois ela ficava na rota de construção de uma autoestrada intergalática. Este é o típico modo de trabalho vogon: muitas raças já foram exterminadas e planetas inteiros destruídos para que o trânsito entre as galáxias ficasse menos congestionado.

Dados dois pontos de referência, pelos quais a nova autoestrada intergalática passará em linha reta, e as coordenadas e habitantes dos planetas do setor espacial, escreva um programa que gere um relatório para os vogons.

Entrada e saída

Entrada

A primeira linha da entrada contém as coordenadas X_1, Y_1, X_2, Y_2 $(-10.000 \leq X_i, Y_i \leq 1.000)$ dos pontos de referência P_1 e P_2 pelos quais a autoestrada passará em linha reta, separadas por um espaço em branco. As coordenadas são números inteiros e a unidade de distância é o ano-luz.

A segunda linha da entrada contém o número $N(1 \leq N \leq 1.000)$ de planetas que fazem parte do setor espacial onde a estrada passará. As próximas N linhas contém, cada uma, as coordenadas X e Y $(-10.000 \leq X, Y \leq 10.000)$ do planeta e o número H $(1 \leq H \leq 100.000)$ de habitantes, em bilhões. Estes valores são números inteiros separados por espaços em branco.

Entrada e saída

Saída

O relatório a ser impresso contém várias linhas. A primeira delas deverá conter a mensagem "Relatorio Vogon #35987-2". Em seguida, deve ser impressa, em uma linha, a mensagem "Distancia entre referencias: d anos-luz", onde d é a distância entre os dois pontos de referência pelos quais a autoestrada deve passar, em anos-luz, com duas casas decimais de precisão.

Na linha seguinte deve ser impressa a mensagem "Setor Oeste:" e, nas duas linhas seguintes, as mensagens "P planeta(s)" e "H bilhao(oes) de habitante(s)", onde P é o número de planetas que ficaram à esquerda da autoestrada, quando se viaja no sentido do primeiro ponto de referência ao segundo, e H é o total de habitantes destes planetas. De modo semelhante, devem ser produzidas três mensagens equivalentes para o Setor Leste, que fica à direita da autoestrada.

Entrada e saída

Por fim, deve ser impressa a mensagem: "Casualidades: P planeta(s)", onde P é o número de planetas que estavam na rota da construção da autoestrada e, naturalmente, tiveram que ser dizimados.

Exemplo de entradas e saídas

Sample Input	Sample Output
-10 -10 30 30 5 1 10 6 5 5 8 2 0 4 -3 -3 30 -2 5 3	Relatorio Vogon #35987-2 Distancia entre referencias: 56.57 anos-luz Setor Oeste: - 2 planeta(s) - 9 bilhao(oes) de habitante(s) Setor Leste: - 1 planeta(s) - 4 bilhao(oes) de habitante(s) Casualidades: 2 planeta(s)

Observações sobre o problema

- A distância entre os dois pontos pode ser computada diretamente usando a função hypot
- Os planetas R que serão destruídos são colineares com os pontos de referência P e Q (isto é, $D(P, Q, R) = 0$)
- O discriminante também é usado para determinar à localização dos pontos: se $D(P, Q, R) < 0$, o planeta está à esquerda do sentido PQ; se $D(P, Q, R) > 0$, o planeta está à direita
- Exceto pela questão da distância, todo o restante do problema pode ser codificado com aritmética inteira
- A complexidade da solução é $O(N)$

Solução AC com complexidade $O(N)$

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

using namespace std;

struct Point {
    int x, y;
};

struct Planet {
    Point P;
    int h;
};

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

Solução AC com complexidade $O(N)$

```
22
23 int D(const Point& P, const Point& Q, const Point& R)
24 {
25     return (P.x * Q.y + P.y * R.x + Q.x * R.y) -
26         (R.x * Q.y + R.y * P.x + Q.x * P.y);
27 }
28
29 int main()
30 {
31     ios::sync_with_stdio(false);
32
33     Point P, Q;
34
35     cin >> P.x >> P.y >> Q.x >> Q.y;
36
37     cout << "Relatorio Vogon #35987-2\n";
38
39     cout.precision(2);
40     cout << "Distancia entre referencias: " << fixed <<
41         distance(P, Q) << " anos-luz\n";
42
```

Solução AC com complexidade $O(N)$

```
43      int N;
44      cin >> N;
45
46      vector<Planet> planets;
47
48      for (int i = 0; i < N; i++)
49      {
50          int x, y, h;
51          cin >> x >> y >> h;
52
53          planets.push_back(Planet { Point { x, y }, h });
54      }
55
56      int left = 0, right = 0, casualities = 0;
57      int hleft = 0, hright = 0;
58
59      for (const auto& p : planets)
60      {
61          int d = D(P, Q, p.P);
62
```

Solução AC com complexidade $O(N)$

```
63          d == 0 ?  casualities++ :
64              (d > 0 ? (left++, hleft += p.h) : (right++, hright += p.h));
65
66      }
67
68      cout << "Setor Oeste:\n";
69      cout << "- " << left << " planeta(s)\n";
70      cout << "- " << hleft << " bilhao(oes) de habitante(s)\n";
71
72      cout << "Setor Leste:\n";
73      cout << "- " << right << " planeta(s)\n";
74      cout << "- " << hright << " bilhao(oes) de habitante(s)\n";
75
76      cout << "Casualidades: " << casualities << " planeta(s)\n";
77
78      return 0;
79  }
```

UVA 11343 – Isolated Segments

Problema

You're given n segments in the rectangular coordinate system. The segments are defined by start and end points (X_i, Y_i) and (X_j, Y_j) $(1 \leq i, j \leq n)$. Coordinates of these points are integer numbers with real value smaller then 1000. Length of each segment is positive.

When 2 segments don't have a common point then it is said that segments don't collide. In any other case segments collide. Be aware that segments collide even if they have only one point in common.

Segment is said to be isolated if it doesn't collide with all the other segments that are given, i.e. segment i is isolated when for each $1 \leq j \leq n$, $(i \neq j)$, segments i and j don't collide. You are asked to find number T – how many segments are isolated.

Entrada e saída

Input First line of input contains number N ($N \leq 50$), then tests follow. First line of each test case contains number M ($M \leq 100$) — the number of segments for this test case to be considered. For this particular test case M lines follow each containing a description of one segment. Segment is described by 2 points: start point (X_{pi}, Y_{pi}) and end point (X_{ei}, Y_{ei}). They are given in such order: $X_{pi}\ Y_{pi}\ X_{ei}\ Y_{ei}$

Entrada e saída

Output

For each test case output one line containing number T.

Exemplo de entradas e saídas

Sample Input

```
6
3
0 0 2 0
1 -1 1 1
2 2 3 3
2
0 0 1 1
1 0 0 1
2
0 0 0 1
0 2 0 3
2
0 0 1 0
1 0 2 0
2
0 0 2 2
1 0 1 1
2
1 3 1 5
1 0 1 6
```

Sample Output

```
1
0
2
0
0
0
```

Observações sobre o problema

- O problema consiste na identificação da interseção entre segmentos
- A verificação de interseção entre segmentos consiste em duas etapas
- A primeira etapa é checar se as retas r e s que contém os segmentos se interceptam
- Em caso positivo é preciso ainda ver se o ponto de interseção está contido nos segmentos ou não
- É preciso atentar aos *corner cases*
- Como é preciso, para cada segmento i, testar todos os segmentos j, a solução tem complexidade $O(M^2)$
- Atenção ao produto dos discriminantes, que pode exceder a capacidade de um inteiro, de modo que é preciso usar o tipo `long long` em seu retorno

Solução AC com complexidade $O(N^2)$

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

using namespace std;
using ll = long long;

struct Point
{
    ll x, y;
};

ll D(const Point& P, const Point& Q, const Point& R)
{
    return (P.x * Q.y + P.y * R.x + Q.x * R.y) -
           (R.x * Q.y + R.y * P.x + Q.x * P.y);
}

struct Segment
{
    Point A, B;

```

Solução AC com complexidade $O(N^2)$

```
22      bool contains(const Point& p) const
23      {
24          return (A.x == B.x) ? min(A.y, B.y) <= p.y && p.y <= max(A.y, B.y)
25              : min(A.x, B.x) <= p.x && p.x <= max(A.x, B.x);
26      }
27
28      bool intersect(const Segment& s) const
29      {
30          auto d1 = D(A, B, s.A), d2 = D(A, B, s.B);
31
32          if ((d1 == 0 && contains(s.A)) || (d2 == 0 && contains(s.B)))
33              return true;
34
35          auto d3 = D(s.A, s.B, A), d4 = D(s.A, s.B, B);
36
37          if ((d3 == 0 && s.contains(A)) || (d4 == 0 && s.contains(B)))
38              return true;
39
40          return (d1 * d2 < 0) && (d3 * d4 < 0);
41      }
42  };
```

Solução AC com complexidade $O(N^2)$

```
44 int solve(const vector<Segment>& vs, int M)
45 {
46     auto ans = 0;
47 
48     for (int i = 0; i < M; ++i)
49     {
50         int hits = 0;
51 
52         for (int j = 0; j < M; ++j)
53         {
54             if (i == j)
55                 continue;
56 
57             hits += vs[i].intersect(vs[j]) ? 1 : 0;
58         }
59 
60         ans += (hits == 0) ? 1 : 0;
61     }
62 
63     return ans;
64 }
```

Solução AC com complexidade $O(N^2)$

```
66 int main()
67 {
68     ios::sync_with_stdio(false);
69
70     int T;
71     cin >> T;
72
73     while (T--)
74     {
75         int M;
76         cin >> M;
77
78         vector<Segment> vs;
79
80         for (int i = 0; i < M; ++i)
81         {
82             int a, b, c, d;
83             cin >> a >> b >> c >> d;
84
85             vs.push_back(Segment { Point {a, b}, Point{c, d} });
86         }
```

Solução AC com complexidade $O(N^2)$

```
87
88          auto ans = solve(vs, M);
89
90          cout << ans << '\n';
91      }
92
93      return 0;
94  }
```

Referências

1. URI 1843 – Vogons!
2. UVA 11343 – Isolated Segments