Geometria Computacional

Retas: Algoritmos

Prof. Edson Alves

2018

Faculdade UnB Gama

Sumário

Classificação de retas

Retas paralelas, concorrentes e coincidentes

- Em relação às possíveis interseções entre duas retas, há três cenários possíveis:
    1. nenhum ponto em comum (retas paralelas)
    2. um único ponto em comum (retas concorrentes)
    3. todos os pontos em comum (retas coincidentes)
- O coeficiente angular é a chave para tal classificação: retas com coeficientes angulares distintos são concorrentes
- Caso duas retas tenham coeficientes angulares iguais, é necessário verificar também o coeficiente linear: se iguais, as retas são coincidentes
- Retas com coeficientes angulares iguais e coeficientes lineares distintos são paralelas
- A implementação destas verificações é trivial na representação baseada na equação reduzida, sendo necessário apenas o cuidado no trato do caso das retas verticais

Exemplo de implementação de classificação de retas em C++

```cpp
// Definição da função equals()

template<typename T>
struct Line {
    // Membros e construtores (equação reduzida)

    bool operator==(const Line& r) const    // Verdadeiro se coincidentes
    {
        if (vertical != r.vertical || !equals(m, r.m)) return false;

        return equals(b, r.b);
    }

    bool parallel(const Line& r) const // Verdadeiro se paralelas
    {
        if (vertical && r.vertical) return b != r.b;
        if (vertical || r.vertical) return false;

        return equals(m, r.m) && !equals(b, r.b);
    }
};
```

Exemplo de implementação de classificação de retas em C++

```cpp
// Definição da função equals()

template<typename T>
struct Line {
    // Membros e construtores (equação geral}

    bool operator==(const Line& r) const
    {
        auto k = a ? a : b;
        auto s = r.a ? r.a : r.b;

        return equals(a*s, r.a*k) && equals(b*s, r.b*k)
            && equals(c*s, r.c*k);
    }

    bool parallel(const Line& r) const
    {
        auto det = a*r.b - b*r.a;
        return det == 0 and !(*this == r);
    }
};
```

Retas perpendiculares

- Duas retas são perpendiculares se o produto de seus coeficientes angulares for igual a -1
- Outra maneira de checar se duas retas são perpendiculares é escolher dois pontos pertencentes a cada reta e montar dois vetores $\vec{u}$ e $\vec{v}$
- Estes pontos podem ser escolhidos de forma eficiente, fazendo $x = 0$ e $y = 0$ (caso a reta não passe na origem)
- Se o produto interno dos dois vetores for igual a zero, as retas são perpendiculares
- Importante notar, porém, é que os coeficientes a e b da equação geral de uma reta formam um vetor $\vec{v} = (a, b)$ perpendicular à reta
- Tais vetores, denominados normais, podem ser utilizados na comparação descrita anteriormente

Exemplo de verificação de retas perpendiculares em C++

```cpp
// Definição da função equals()

template<typename T>
struct Line
{
    // Membros e construtores (equação reduzida)

    bool orthogonal(const Line& r) const // Verdadeiro se perpendiculares
    {
        if (vertical && r.vertical)
            return false;

        if ((vertical && equals(r.m, 0)) || (equals(m, 0) && r.vertical))
            return true;

        if (vertical || r.vertical)
            return false;

        return equals(m * r.m, -1.0);
    }
};
```

Exemplo de verificação de retas perpendiculares em C++

```cpp
// Definição da função equals()

template<typename T>
struct Line
{
    // Membros e construtores (equação geral)

    bool orthogonal(const Line& r) const  // Verdadeiro se perpendiculares
    {
        return equals(a * r.a + b * r.b, 0);
    }
};
```

Relação entre retas

Interseção entre retas

- Dado um par de retas r e s, elas podem ser:
  1. coincidentes (infinitas interseções),
  2. paralelas (nenhuma interseção), ou
  3. concorrentes (um único ponto de interseção)
- Para encontrar o ponto de interseção, no caso de retas concorrentes, basta resolver o sistema linear resultante das equações gerais das duas retas:

$$\begin{cases} a_r x + b_r y + c_r = 0 \\ a_s x + b_s y + c_s = 0 \end{cases}$$

- As soluções são

$$x = (-c_r b_s + c_s b_r)/(a_r b_s - a_s b_r)$$

$$y = (-c_s a_r + c_r a_s)/(a_r b_s - a_s b_r)$$

Exemplo de implementação da interseção entre duas retas

```cpp
// Definição função equals(), das classes Point e Line

const int INF { -1 };

template<typename T>
std::pair<int, Point<T>> intersections(const Line<T>& r, const Line<T>& s)
{
    auto det = r.a * s.b - r.b * s.a;

    if (equals(det, 0))     // Coincidentes ou paralelas
    {
        int qtd = (r == s) ? INF : 0;
        return std::pair<int, Point<T>>(qtd, Point());
    } else                  // Concorrentes
    {
        auto x = (-r.c * s.b + s.c * r.b) / det;
        auto y = (-s.c * r.a + r.c * s.a) / det;

        return std::pair<int, Point<T>>(1, Point<T>(x, y));
    }
}
```

Ângulo entre retas

- Para mensurar o ângulo formado por duas retas (ou dois segmentos de reta), é preciso identificar os vetores $\vec{u}$ e $\vec{v}$ que estejam na mesma direção das duas retas e usar o produto interno
- Dados dois pontos distintos $P = (x_p, y_p)$ e $Q = (x_q, y_q)$, o vetor direção da reta que passa por P e Q é dado por $\vec{u} = (x_q - x_p, y_q - y_p)$
- De posse dos vetores de direção, o cosseno ângulo entre as retas é dado por

$$\cos\theta = \frac{u \cdot v}{|u||v|} = \frac{u_x v_x + u_y v_y}{\sqrt{u_x^2 + u_y^2}\sqrt{v_x^2 + v_y^2}}$$

- Para achar o ângulo, basta computar a função inversa do cosseno (`acos()`, na biblioteca de matemática padrão do C/C++) no lado direito da expressão acima

Exemplo de implementação do ângulo entre duas retas

```cpp
// Definição da classe Point

// Ângulo entre os segmentos de reta PQ e RS
template<typename T>
double angle(const Point<T>& P, const Point<T>& Q,
             const Point<T>& R, const Point<T>& S)
{
    auto ux = P.x - Q.x;
    auto uy = P.y - Q.y;

    auto vx = R.x - S.x;
    auto vy = R.y - S.y;

    auto num = ux * vx + uy * vy;
    auto den = hypot(ux, uy) * hypot(vx, vy);

    // Caso especial: se den == 0, algum dos vetores é degenerado: os dois
    // pontos são iguais. Neste caso, o ângulo não está definido

    return acos(num / den);
}
```

Interseção entre segmentos de reta

- Para determinar a interseçaõ entre dois segmentos de reta é preciso resolver o problema para as duas retas que contém os respectivos segmentos e verificar se as interseções, se existirem, pertencem a ambos intervalos
- Embora esta abordagem permita conhecer as coordenadas das possíveis interseções, ela traz alguns problemas em potencial:
    1. mesmo que as retas sejam coincidentes, não há garantias que os segmentos tenham interseção
    2. a concorrência também não garante interseção: ainda é preciso verificar se o ponto pertence a ambos intervalos
- Para identificar apenas se há interseção entre ambos segmentos, sem determinar as coordenadas de tal interseção, o problema fica simplificado, e será abordado mais adiante

Rotina que verifica se um ponto P pertence ao segmento AB

```cpp
// Definição da classe Point e da função de comparação equals()

// Verifica se o ponto P pertence ao segmento de reta AB
template<typename T>
bool contains(const Point<T>& A, const Point<T>& B, const Point<T>& P)
{
    if (P == A || P == B)
        return true;

    auto xmin = min(A.x, B.x);
    auto xmax = max(A.x, B.x);
    auto ymin = min(A.y, B.y);
    auto ymax = max(A.y, B.y);

    if (P.x < xmin || P.x > xmax || P.y < ymin || P.y > ymax)
        return false;

    return equals((P.y - A.y)*(P.x - B.x), (P.x - A.x)*(P.y - B.y));
}
```

Referências

1. **HALIM**, Felix; **HALIM**, Steve. *Competitive Programming 3*, 2010.
2. **LAAKSONEN**, Antti. *Competitive Programmer's Handbook*, 2018.
3. **De BERG**, Mark; **CHEONG**, Otfried. *Computational Geometry: Algorithms and Applications*, 2008.
4. David E. Joyce. *Euclid's Elements*. Acesso em 15/02/2019[1]
5. Wikipédia. *Geometria Euclidiana*. Acesso em 15/02/2019[2].

[1]https://mathcs.clarku.edu/ djoyce/elements/bookI/defI1.html
[2]https://pt.wikipedia.org/wiki/Geometria_euclidiana