Geometria Computacional

Círculos: Algoritmos

Prof. Edson Alves

2018

Faculdade UnB Gama

Sumário

Relação entre pontos e círculos

Relação de pertinência de um ponto P

- Dado um ponto P e um círculo de centro C e raio r, uma (e apenas uma) das três afirmações abaixo será verdadeira:
    1. P está dentro do círculo
    2. P está sobre o círculo
    3. P está fora do círculo
- Para determinar qual é a relação válida, basta computar a distância entre o ponto P e o centro C do círculo
- Caso esta distância seja menor, igual ou maior que r, P estará dentro, sobre e fora do círculo, respectivamente
- O conjunto de pontos que estão dentro do círculo é denominado disco de raio r e centro C

Implementação da posição do ponto em um círculo em C++

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

template<typename T>
struct Circle {
    Point<T> C;
    T r;

    enum { IN, ON, OUT } PointPosition;

    PointPosition position(const Point& P) const
    {
        auto d = dist(P, C);

        return equals(d, r) ? ON : (d < r ? IN : OUT);
    }
};
```

Construção de círculos a partir de pontos

- É possível identificar o(s) círculo(s) que interceptam um conjunto de N pontos dados
- No caso $N = 1$, existem infinitos círculos (com infinitos raios possíveis) que passam por um dado ponto P
- O caso $N = 2$ se torna mais interessante se o raio r for pré-determinado
- Dados dois pontos P e Q e o um raio r, os cenários possíveis são:
  1. $P = Q$: esta situação é idêntica ao caso $N = 1$
  2. $\mathrm{dist}(P, Q) = 2r$: se a distância entre os dois pontos dados é igual ao diâmetro do círculo, existe um único círculo de raio r que passa por P e Q, cujo centro será o ponto médio do segmento PQ
  3. $\mathrm{dist}(P, Q) > 2r$: neste caso, nenhum círculo de r pode passar por ambos pontos simultaneamente
  4. $\mathrm{dist}(P, Q) < 2r$: neste caso, exatamente dois círculos passam por P e Q com raio r

Implementação da identificação de um círculo a partir de dois pontos e o raio

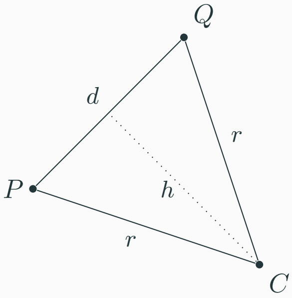
```cpp
// O código abaixo, adaptado do livro Competitive Programming 3.
// A função retorna um dos círculos possíveis: o outro pode ser
// encontrado invertendo os parâmetros P e Q na chamada da função

#include <optional>

// Definição da class Point

template<typename T>
struct Circle {
    // Membros e construtores

    static std::optional<Circle>
    from_2_points_and_r(const Point& P, const Point& Q, T r, Circle& c)
    {
        double d2 = (P.x - Q.x) * (P.x - Q.x) + (P.y - Q.y) * (P.y - Q.y);
        double det = r * r / d2 - 0.25;

        if (det < 0.0)
            return { };
```

Implementação da identificação de um círculo a partir de dois pontos e o raio

```

        double h = sqrt(det);

        auto x = (P.x + Q.x) * 0.5 + (P.y - Q.y) * h;
        auto y = (P.y + Q.y) * 0.5 + (Q.x - P.x) * h;

        return Circle { Point(x, y), r };
    }
}
```

Demonstração do algoritmo

- A implementação do algoritmo anterior, embora simples, é baseada em fatos não-triviais
- Sejam P e Q dois pontos distintos tais que $\mathrm{dist}(P, Q) < 2r$, onde r é o raio dado
- Neste cenário, o centro C do círculo não pertence ao segmento PQ
- Assim, é formado um triângulo PCQ



Demonstração do algoritmo

- A Lei dos Cossenos diz que, num triângulo de lados a, b, c cujo ângulo oposto a a é α, vale a igualdade

$$a^2 = b^2 + c^2 - 2ab\cos\alpha$$

- Aplicando a Lei dos Cossenos ao triângulo PCQ, e considerando θ o ângulo oposto ao lado d, têm-se que

$$d^2 = r^2 + r^2 - 2r^2\cos\theta = 2r^2(1 - \cos\theta)$$

- Como $|\cos\theta| \leq 1$, vale a desigualdade

$$d^2 \leq 4r^2,$$

a qual pode ser reescrita como

$$\frac{r^2}{d^2} \geq \frac{1}{4}$$

Demonstração do algoritmo

- Defina o discriminante
$$\Delta = \frac{r^2}{d^2} - \frac{1}{4}$$
- Para que exista um círculo que passe por P e Q é preciso que $\Delta \geq 0$
- Como PCQ é um triângulo isóceles, sua altura (cuma medida é h) coincide com a mediana
- Seja M o ponto médio de PQ. Aplicando o Teorema de Pitágoras ao triângulo PMC obtêm-se

$$r^2 = h^2 + \left(\frac{d}{2}\right)^2,$$

isto é,
$$h = \sqrt{r^2 - \frac{d^2}{4}} = d\sqrt{\frac{r^2}{d^2} - \frac{1}{4}} = d\sqrt{\Delta}$$

Demonstração do algoritmo

- Sejam $\vec{P}$ e $\vec{Q}$ os vetores-posição dos pontos P e Q. O vetor unitário $\vec{u}$ que parte de P em direção a Q é dado por

$$\vec{u} = \frac{\vec{Q} - \vec{P}}{\|\vec{Q} - \vec{P}\|} = \left(\frac{x_Q - x_P}{d}, \frac{y_Q - y_P}{d} \right)$$

- O vetor unitário $\vec{n}$, normal a $\vec{u}$, é dado por

$$\vec{n} = \left(\frac{y_P - y_Q}{d}, \frac{x_Q - x_P}{d} \right)$$

- Assim, o vetor posição do centro C do círculo pode ser encontrado pela soma vetorial

$$\vec{C} = \vec{P} + \frac{d}{2}\vec{u} + h\vec{n}$$

- As duas soluções possíveis diferem pelo sentido do vetor $\vec{n}$

Demonstração do algoritmo

- Portanto,

$$
\begin{aligned}
\vec{C} &= \vec{P} + \frac{d}{2}\vec{u} + h\vec{n} \\
&= (x_P, y_P) + \frac{d}{2}\left(\frac{x_Q - x_P}{d}, \frac{y_Q - y_P}{d}\right) + h\left(\frac{y_P - y_Q}{d}, \frac{x_Q - x_P}{d}\right) \\
&= (x_P, y_P) + \left(\frac{x_Q - x_P}{2}, \frac{y_Q - y_P}{2}\right) + d\sqrt{\Delta}\left(\frac{y_P - y_Q}{d}, \frac{x_Q - x_P}{d}\right) \\
&= (x_P, y_P) + \left(\frac{x_Q - x_P}{2}, \frac{y_Q - y_P}{2}\right) + \sqrt{\Delta}\,(y_P - y_Q, x_Q - x_P) \\
&= \left(\frac{x_P + x_Q}{2}, \frac{y_P + y_Q}{2}\right) + \sqrt{\Delta}\,(y_P - y_Q, x_Q - x_P)
\end{aligned}
$$

Relação entre círculo e reta

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.