Geometria Computacional

Polígonos

Prof. Edson Alves

Faculdade UnB Gama

Sumário

Definição

Definição de polígono

- Polígonos são figuras planas delimitadas por caminhos fechados (o vértice de partida é o vértice de chegada), compostos por segmentos de retas que unem vértices consecutivos

Definição de polígono

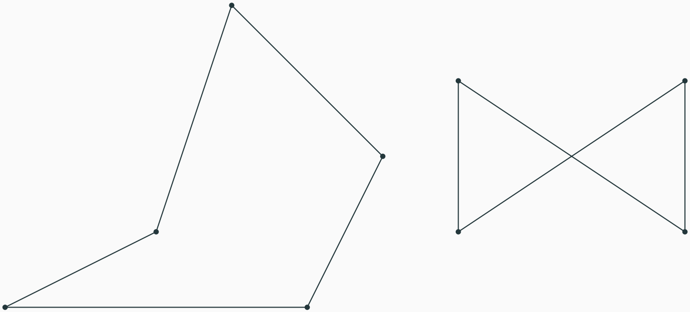
- Polígonos são figuras planas delimitadas por caminhos fechados (o vértice de partida é o vértice de chegada), compostos por segmentos de retas que unem vértices consecutivos
- Os segmentos que unem os vértices são denominados arestas

Definição de polígono

- Polígonos são figuras planas delimitadas por caminhos fechados (o vértice de partida é o vértice de chegada), compostos por segmentos de retas que unem vértices consecutivos
- Os segmentos que unem os vértices são denominados arestas
- Embora alguns polígonos especiais (triângulos, quadriláteros) possam ter tratamento especial, os algoritmos de polígonos podem ser aplicados igualmente a estes entes geométricos

Definição de polígono

- Polígonos são figuras planas delimitadas por caminhos fechados (o vértice de partida é o vértice de chegada), compostos por segmentos de retas que unem vértices consecutivos
- Os segmentos que unem os vértices são denominados arestas
- Embora alguns polígonos especiais (triângulos, quadriláteros) possam ter tratamento especial, os algoritmos de polígonos podem ser aplicados igualmente a estes entes geométricos



Representação de polígonos

- A representação mais comum de um polígono é a listagem de seus vértices, sendo que as arestas ficam subentendidas (há sempre uma aresta unindo dois vértice consecutivos)

Representação de polígonos

- A representação mais comum de um polígono é a listagem de seus vértices, sendo que as arestas ficam subentendidas (há sempre uma aresta unindo dois vértice consecutivos)
- Para facilitar a implementação de algumas rotinas, pode ser conveniente inserir, ao final da lista, o ponto de partida

Representação de polígonos

- A representação mais comum de um polígono é a listagem de seus vértices, sendo que as arestas ficam subentendidas (há sempre uma aresta unindo dois vértice consecutivos)
- Para facilitar a implementação de algumas rotinas, pode ser conveniente inserir, ao final da lista, o ponto de partida
- É preciso tomar cuidado: ao fazer isso, o número de vértices do polígono passa a ser o número de elementos da lista subtraído de uma unidade

Representação de polígonos

- A representação mais comum de um polígono é a listagem de seus vértices, sendo que as arestas ficam subentendidas (há sempre uma aresta unindo dois vértice consecutivos)
- Para facilitar a implementação de algumas rotinas, pode ser conveniente inserir, ao final da lista, o ponto de partida
- É preciso tomar cuidado: ao fazer isso, o número de vértices do polígono passa a ser o número de elementos da lista subtraído de uma unidade

```cpp
template<typename T>
using Polygon = vector<Point<T>>;
```

Representação de polígonos

- A representação mais comum de um polígono é a listagem de seus vértices, sendo que as arestas ficam subentendidas (há sempre uma aresta unindo dois vértice consecutivos)
- Para facilitar a implementação de algumas rotinas, pode ser conveniente inserir, ao final da lista, o ponto de partida
- É preciso tomar cuidado: ao fazer isso, o número de vértices do polígono passa a ser o número de elementos da lista subtraído de uma unidade

```cpp
template<typename T>
using Polygon = vector<Point<T>>;
```

- Esta implementação é a mais compacta possível, mas requer atenção a questão do número de vértices, conforme já comentado

Representação de polígonos

- A representação mais comum de um polígono é a listagem de seus vértices, sendo que as arestas ficam subentendidas (há sempre uma aresta unindo dois vértice consecutivos)
- Para facilitar a implementação de algumas rotinas, pode ser conveniente inserir, ao final da lista, o ponto de partida
- É preciso tomar cuidado: ao fazer isso, o número de vértices do polígono passa a ser o número de elementos da lista subtraído de uma unidade

```cpp
template<typename T>
using Polygon = vector<Point<T>>;
```

- Esta implementação é a mais compacta possível, mas requer atenção a questão do número de vértices, conforme já comentado
- Uma implementação mais extensa evita os problemas já mencionados

Exemplo de implementação de um polígono em C++

```cpp
#include <bits/stdc++.h>

using namespace std;

template<typename T>
struct Point { T x, y; };

template<typename T>
class Polygon {
private:
    vector<Point<T>> vs;
    int n;

public:
    // O parâmetro deve conter os n vértices do polígono
    Polygon(const vector<Point<T>>& ps) : vs(ps), n(vs.size())
    {
        vs.push_back(vs.front());
    }
```

Polígonos côncavos e convexos

- Um polígono é dito convexo se, para quaisquer dois pontos P e Q localizados no interior do polígono, o segmento de reta PQ não intercepta nenhuma das arestas do polígono

Polígonos côncavos e convexos

- Um polígono é dito convexo se, para quaisquer dois pontos P e Q localizados no interior do polígono, o segmento de reta PQ não intercepta nenhuma das arestas do polígono
- Caso contrário, o polígono é dito côncavo

Polígonos côncavos e convexos

- Um polígono é dito convexo se, para quaisquer dois pontos P e Q localizados no interior do polígono, o segmento de reta PQ não intercepta nenhuma das arestas do polígono
- Caso contrário, o polígono é dito côncavo
- É possível determinar se um polígono é ou não convexo sem recorrer à busca completa, isto é, testar todos os possíveis pares de pontos interiores ao polígono

Polígonos côncavos e convexos

- Um polígono é dito convexo se, para quaisquer dois pontos P e Q localizados no interior do polígono, o segmento de reta PQ não intercepta nenhuma das arestas do polígono
- Caso contrário, o polígono é dito côncavo
- É possível determinar se um polígono é ou não convexo sem recorrer à busca completa, isto é, testar todos os possíveis pares de pontos interiores ao polígono
- A orientação D, entre pontos e reta, pode ser utilizada para tal fim

Polígonos côncavos e convexos

- Um polígono é dito convexo se, para quaisquer dois pontos P e Q localizados no interior do polígono, o segmento de reta PQ não intercepta nenhuma das arestas do polígono
- Caso contrário, o polígono é dito côncavo
- É possível determinar se um polígono é ou não convexo sem recorrer à busca completa, isto é, testar todos os possíveis pares de pontos interiores ao polígono
- A orientação D, entre pontos e reta, pode ser utilizada para tal fim
- Basta checar se, para quaisquer três pontos consecutivos do polígono, eles tem a mesma orientação: ou sempre à esquerda, ou sempre à direita

Implementação da rotina de verificação de convexidade

```
private:
    T D(const Point<T>& P, const Point<T>& Q, const Point<T>& R) const
    {
        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);
    }

public:
    bool convex() const {
        // Um polígono deve ter, no minimo, 3 vértices
        if (n < 3) return false;

        int P = 0, N = 0, Z = 0;

        for (int i = 0; i < n; ++i) {
            auto d = D(vs[i], vs[(i + 1) % n], vs[(i + 2) % n]);
            d ? (d > 0 ? ++P : ++N) : ++Z;
        }

        return not ((P and N) or (P == 0 and N == 0));
    }
```

Algoritmos envolvendo polígonos

Perímetro

- O perímetro de um polígono consiste na medida de seu contorno, isto é, a soma dos comprimentos de suas aresta

Perímetro

- O perímetro de um polígono consiste na medida de seu contorno, isto é, a soma dos comprimentos de suas aresta
- Ele pode ser calculado diretamente a partir da representação do polígono por meio de seus vértices

Perímetro

- O perímetro de um polígono consiste na medida de seu contorno, isto é, a soma dos comprimentos de suas aresta
- Ele pode ser calculado diretamente a partir da representação do polígono por meio de seus vértices

```cpp
    double distance(const Point<T>&P, const Point<T>& Q)
    {
        return hypot(P.x - Q.x, P.y - Q.y);
    }
public:
    double perimeter() const
    {
        auto p = 0.0;

        for (int i = 0; i < n; ++i)
            p += distance(vs[i], vs[i + 1]);

        return p;
    }
```

Área

- A área delimitada por um polígono pode ser também determinada diretamente a partir de seus vértices

Área

- A área delimitada por um polígono pode ser também determinada diretamente a partir de seus vértices
- Ela corresponde à metade do valor absoluto do "determinante" abaixo (as aspas significam que a notação remete a um determinante, mas não é um determinante de fato, uma vez que a matriz não é quadrada)

$$
\begin{aligned}
A &= \frac{1}{2}\begin{vmatrix} x_0 & y_0 \\ x_1 & y_1 \\ x_2 & y_2 \\ \dots & \dots \\ x_{n-1} & y_{n-1} \end{vmatrix} \\
&= \frac{1}{2}\left|x_0y_1 + x_1y_2 + \ldots + x_{n-1}y_0 - x_1y_0 - x_2y_1 - \ldots - x_0y_{n-1}\right|
\end{aligned}
$$

Implementação da área do polígono

```cpp
double area() const
{
    auto a = 0.0;

    for (int i = 0; i < n; ++i)
    {
        a += vs[i].x * vs[i + 1].y;
        a -= vs[i + 1].x * vs[i].y;
    }

    return 0.5 * fabs(a);
}
```

Área de polígonos regulares

- Um polígono é dito regular se todos os seus lados têm a mesma medida

Área de polígonos regulares

- Um polígono é dito regular se todos os seus lados têm a mesma medida
- A área também pode ser computada através do conhecimento do número de lados n e um dos três valores abaixo:

Área de polígonos regulares

- Um polígono é dito regular se todos os seus lados têm a mesma medida
- A área também pode ser computada através do conhecimento do número de lados n e um dos três valores abaixo:
    1. o comprimento de um dos lados (s)

Área de polígonos regulares

- Um polígono é dito regular se todos os seus lados têm a mesma medida
- A área também pode ser computada através do conhecimento do número de lados n e um dos três valores abaixo:
    1. o comprimento de um dos lados (s)
    2. a apótema, isto é, o raio do círculo inscrito (r)

Área de polígonos regulares

- Um polígono é dito regular se todos os seus lados têm a mesma medida
- A área também pode ser computada através do conhecimento do número de lados n e um dos três valores abaixo:
  1. o comprimento de um dos lados (s)
  2. a apótema, isto é, o raio do círculo inscrito (r)
  3. o raio do círculo circunscrito (R)

Área de polígonos regulares

- Um polígono é dito regular se todos os seus lados têm a mesma medida
- A área também pode ser computada através do conhecimento do número de lados n e um dos três valores abaixo:
  1. o comprimento de um dos lados (s)
  2. a apótema, isto é, o raio do círculo inscrito (r)
  3. o raio do círculo circunscrito (R)
- As expressões abaixo relacionam a área do polígono regular com as medidas supracitadas:

$$A = \frac{1}{2}nrs = \frac{1}{4}ns^2 \cot \frac{\pi}{n} = nr^2 \tan \frac{\pi}{n} = \frac{1}{2}nR^2 \sin \frac{2\pi}{n}$$

Relação entre pontos e polígonos

- Para se verificar se um ponto P está localizado, ou não, no interior de um polígono, basta computar a soma dos ângulos formados por P e cada par de vértices do polígono

Relação entre pontos e polígonos

- Para se verificar se um ponto P está localizado, ou não, no interior de um polígono, basta computar a soma dos ângulos formados por P e cada par de vértices do polígono
- Esta soma deve adicionar o ângulo se o ponto está na mesma orientação do polígono, e subtrair em caso contrário

Relação entre pontos e polígonos

- Para se verificar se um ponto P está localizado, ou não, no interior de um polígono, basta computar a soma dos ângulos formados por P e cada par de vértices do polígono
- Esta soma deve adicionar o ângulo se o ponto está na mesma orientação do polígono, e subtrair em caso contrário
- Se o total for igual a 2π, o ponto está no interior do polígono

Relação entre pontos e polígonos

- Para se verificar se um ponto P está localizado, ou não, no interior de um polígono, basta computar a soma dos ângulos formados por P e cada par de vértices do polígono
- Esta soma deve adicionar o ângulo se o ponto está na mesma orientação do polígono, e subtrair em caso contrário
- Se o total for igual a 2π, o ponto está no interior do polígono
- Esta verificação vale tanto para polígonos convexos quanto côncavos

Implementação da relação entre pontos e polígonos

```cpp
// Ângulo APB, em radianos
double angle(const Point<T>& P, const Point<T>& A, const Point<T>& B)
{
    auto ux = P.x - A.x;
    auto uy = P.y - A.y;

    auto vx = P.x - B.x;
    auto vy = P.y - B.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);
}
```

Implementação da relação entre pontos e polígonos

```cpp
    bool equals(double x, double y) {
        static const double EPS { 1e-6 };
        return fabs(x - y) < EPS;
    }
public:
    bool contains(const Point<T>& P) const
    {
        if (n < 3) return false;

        auto sum = 0.0;

        for (int i = 0; i < n - 1; ++i) {
            auto d = D(P, vs[i], vs[i + 1]);
            auto a = angle(P, vs[i], vs[i + 1]);
            sum += d > 0 ? a : -a;
        }

        static const double PI = acos(-1.0);
        return equals(fabs(sum), 2*PI);
    }
```

Relação entre polígonos e retas

- Considere uma reta r, que passa pelos pontos A e B, e um polígono convexo P, com n vértices

Relação entre polígonos e retas

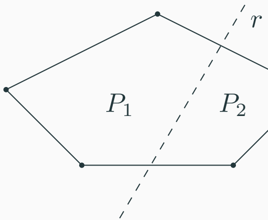
- Considere uma reta r, que passa pelos pontos A e B, e um polígono convexo P, com n vértices
- A reta r secciona o polígono em duas regiões, esquerda e direita, que podem ser ou uma vazias e outra contendo P integralmente, ou serem compostas de dois polígonos convexos P_1 e P_2, resultantes do corte de P por r

Relação entre polígonos e retas

- Considere uma reta r, que passa pelos pontos A e B, e um polígono convexo P, com n vértices
- A reta r secciona o polígono em duas regiões, esquerda e direita, que podem ser ou uma vazias e outra contendo P integralmente, ou serem compostas de dois polígonos convexos P_1 e P_2, resultantes do corte de P por r
- A rotina `cut_polygon()`, apresentada a seguir e adaptada de Competitive Programming 3, retorna a região a esquerda do corte, considerando que P está descrito no sentido anti-horário

Relação entre polígonos e retas

- Considere uma reta r, que passa pelos pontos A e B, e um polígono convexo P, com n vértices
- A reta r secciona o polígono em duas regiões, esquerda e direita, que podem ser ou uma vazias e outra contendo P integralmente, ou serem compostas de dois polígonos convexos P_1 e P_2, resultantes do corte de P por r
- A rotina `cut_polygon()`, apresentada a seguir e adaptada de Competitive Programming 3, retorna a região a esquerda do corte, considerando que P está descrito no sentido anti-horário



Implementação da relação entre pontos e retas

```
    // Interseção entre a reta AB e o segmento de reta PQ
    Point<T> intersection(const Point<T>& P, const Point<T>& Q,
                          const Point<T>& A, const Point<T>& B)
    {
        auto a = B.y - A.y;
        auto b = A.x - B.x;
        auto c = B.x * A.y - A.x * B.y;
        auto u = fabs(a * P.x + b * P.y + c);
        auto v = fabs(a * Q.x + b * Q.y + c);

        // Média ponderada pelas distâncias de P e Q até a reta AB
        return {(P.x * v + Q.x * u)/(u + v), (P.y * v + Q.y * u)/(u + v)};
    }

public:
    // Corta o polígono com a reta r que passa por A e B
    Polygon cut_polygon(const Point<T>& A, const Point<T>& B) const
    {
        vector<Point<T>> points;
        const double EPS { 1e-6 };
```

Implementação da relação entre pontos e retas

```cpp

    for (int i = 0; i < n; ++i)
    {
        auto d1 = D(A, B, vs[i]);
        auto d2 = D(A, B, vs[i + 1]);

        // Vértice à esquerda da reta
        if (d1 > -EPS)
            points.push_back(vs[i]);

        // A aresta cruza a reta
        if (d1 * d2 < -EPS)
            points.push_back(intersection(vs[i], vs[i + 1], A, B));
    }

    return Polygon(points);
}
```

Círculo circunscrito

- Um polígono regular (medidas dos lados iguais) de n lados possui um círculo circunscrito (cujos vértices pertencem ao círculo) e um círculo inscrito (cujos lados são tangentes ao círculo)

Círculo circunscrito

- Um polígono regular (medidas dos lados iguais) de n lados possui um círculo circunscrito (cujos vértices pertencem ao círculo) e um círculo inscrito (cujos lados são tangentes ao círculo)
- O raio R do círculo circunscrito é igual ao raio do polígono: a distância entre o seu centro e um de seus vértices

Círculo circunscrito

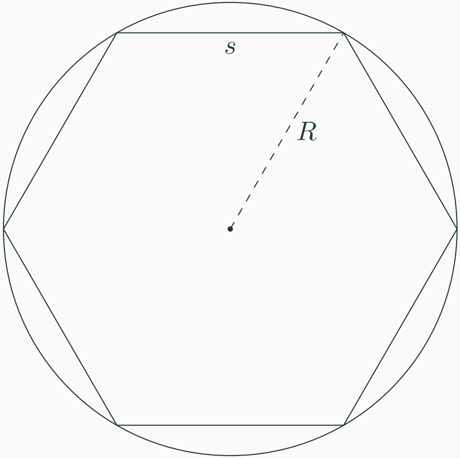
- Um polígono regular (medidas dos lados iguais) de n lados possui um círculo circunscrito (cujos vértices pertencem ao círculo) e um círculo inscrito (cujos lados são tangentes ao círculo)
- O raio R do círculo circunscrito é igual ao raio do polígono: a distância entre o seu centro e um de seus vértices
- Se s é a medida do lado do polígono, então

$$\sin \frac{\pi}{n} = \frac{(s/2)}{R},$$

isto é,

$$R = \frac{s}{2} \csc \frac{\pi}{n}$$

Visualização do círculo circunscrito



Implementação do cálculo do raio R do círculo circunscrito

```
double circumradius() const
{
    auto s = distance(vs[0], vs[1]);
    const double PI { acos(-1.0) };

    return (s/2.0)*(1.0/sin(PI/n));
}
```

Círculo inscrito

- O raio r do círculo inscrito pode ser determinado a partir da medida s de um dos lados do polígono regular, através da relação

$$\tan \frac{\pi}{n} = \frac{(s/2)}{r},$$

isto é,

$$r = \frac{s}{2} \cot \frac{\pi}{n}$$

Círculo inscrito

- O raio r do círculo inscrito pode ser determinado a partir da medida s de um dos lados do polígono regular, através da relação

$$\tan \frac{\pi}{n} = \frac{(s/2)}{r},$$

isto é,

$$r = \frac{s}{2} \cot \frac{\pi}{n}$$

- O raio r também é denominado apótema do polígono regular

Círculo inscrito

- O raio r do círculo inscrito pode ser determinado a partir da medida s de um dos lados do polígono regular, através da relação

$$\tan \frac{\pi}{n} = \frac{(s/2)}{r},$$

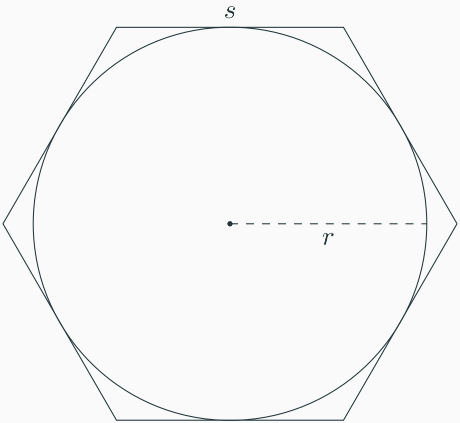
isto é,

$$r = \frac{s}{2} \cot \frac{\pi}{n}$$

- O raio r também é denominado apótema do polígono regular
- Os raios R e r se relacionam de modo que

$$r = R \cos \frac{\pi}{n}$$

Visualização do círculo inscrito



Implementação do cálculo do raio r do círculo inscrito

```cpp
    double apothem() const
    {
        auto s = distance(vs[0], vs[1]);
        const double PI { acos(-1.0) };

        return (s/2.0)*(1.0/tan(PI/n));
    }
```

Referências

1. **HALIM**, Felix; **HALIM**, Steve. *Competitive Programming 3*, 2010.
2. Math Open Reference. Incircle of a Polygon, acesso em 18/08/2016.
3. Mathwords. Area of a Regular Polygon, acesso em 20/09/2016.
4. Wikipédia. Regular Polygon, acesso em 18/08/2016.