DCC207 – Algoritmos 2

Aula 04 – Introdução a Geometria Computacional (Parte 01)

Professor Renato Vimieiro

DCC/ICEx/UFMG

Introdução

- A área de geometria computacional é um ramo da computação que estuda soluções algorítmicas para problemas geométricos
- A solução para esses problemas surge como demanda de outras áreas:
  - Robótica (navegação)
  - Sistemas CAD
  - Sistemas de Informação Geográficos
  - Epidemiologia (proposta de John Snow para surto de cólera)

Operações com segmentos de reta

- Considere segmentos de reta no plano definidos por suas extremidades
- Os primeiros problemas que iremos explorar são:
  - Dados dois segmentos de reta orientados (vetores) p0p1, p0p2, como saber se o primeiro está em uma posição horária ou anti-horária do segundo com respeito ao ponto em comum p0?
  - Dados dois segmentos de reta p0p1 e p1p2, há mudança de direção (e em qual) quando passamos pelo ponto p1 ao seguir o caminho p0p1-p1p2?
  - Dois segmentos quaisquer p0p1 e p2p3 se interceptam em algum ponto?
- As soluções que veremos para esses problemas operam em tempo O(1) e utilizam somente operações elementares
  - Não serão usadas funções trigonométricas nem divisão, já que essas podem apresentar falhas de precisão em alguns casos (e.g. interseção de retas)

Orientação relativa de segmentos de reta

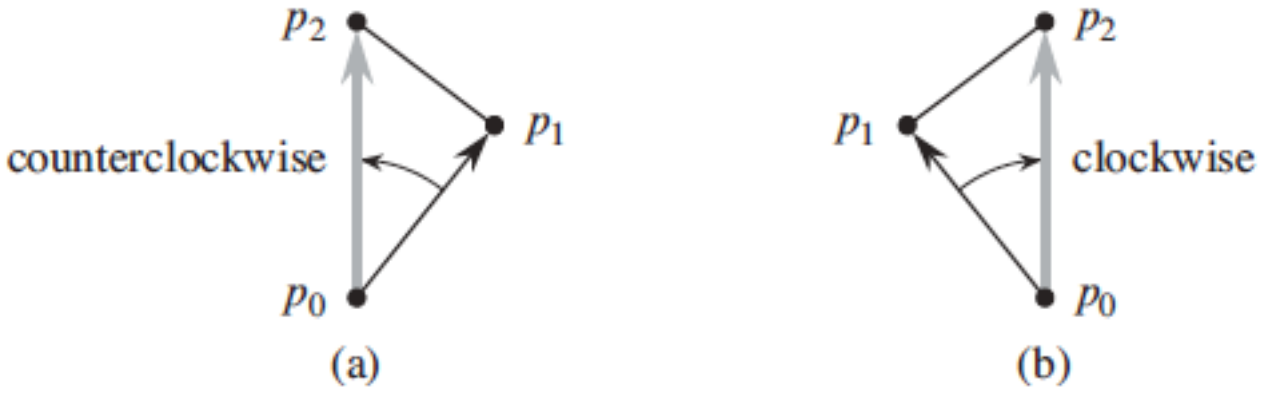
- Considere o primeiro problema listado anteriormente de se determinar a posição relativa de dois segmentos de reta
- O produto vetorial entre dois vetores **a** e **b**, é um vetor **c** = **a** x **b** perpendicular a ambos **a** e **b**.
- A magnitude de **c** é a área do paralelogramo formado pelos pontos (0,0), **a**, **b**, **a**+**b**
- Considerando **a** e **b** vetores no plano, a magnitude de **c** pode ser expressada por
  - $|\mathbf{c}| = |a1b2 - b1a2|$
  - **a** = (a1,a2), **b** = (b1,b2)
- Para simplificar a notação, vamos nos referir a $|\mathbf{c}|$ simplesmente como produto vetorial

Orientação relativa de segmentos de reta

- **Teorema**: Dados **a** e **b** dois vetores 2d. Se o sinal de $|\mathbf{a} \times \mathbf{b}| > 0$, então **a** está numa posição horária de **b**. Se $|\mathbf{a} \times \mathbf{b}| < 0$, então **a** está numa posição anti-horária de **b**. Se $|\mathbf{a} \times \mathbf{b}| = 0$, então **a** e **b** são colineares. (Assumindo a notação do slide anterior)
- **Prova:** Segue da regra da mão direita. Intuitivamente, se **a** está numa posição horária a **b**, então a inclinação da reta que passa por **a** deve ser menor que a que passa por **b**. Logo,
  - $a2/a1 < b2/b1 \leftrightarrow a1b2 > a2b1 \leftrightarrow a1b2 - a2b1 > 0 \leftrightarrow |\mathbf{a} \times \mathbf{b}| > 0$
- Usando o teorema acima, podemos decidir a posição relativa de dois segmentos p0p1 e p0p2
  - Basta tomar p0 como origem e o sinal da expressão (x1-x0)(y2-y0)-(x2-x0)(y1-y0) como orientação

Direção da rota por segmentos

- Considere o segundo problema de determinar se há mudança de direção ao seguir o caminho definido pelos pontos p0-p1-p2
- Esse problema é equivalente ao problema anterior, como podemos visualizar na figura abaixo



Determinar se dois segmentos se interceptam

- Podemos verificar se dois segmentos se interceptam, verificando se eles cruzam as retas que os sobrepõem
- Dois segmentos se interceptam sse uma das seguintes condições se aplica:
  - Cada segmento atravessa a reta que se sobrepõe ao outro segmento
  - Um dos terminais de um segmento se encontra sobre o outro segmento
- Podemos usar a orientação relativa de segmentos para verificar se um segmento atravessa a reta que se sobrepõe ao outro
  - Um segmento atravessa uma reta se seus pontos estão em lados opostos da reta

Determinar se dois segmentos se interceptam

- Considerando os segmentos p1p2 e p3p4, p1p2 atravessa a reta sobre p3p4 se:
  - p3p1 está à esquerda de p3p4, e p3p2 está à direita de p3p4; ou
  - p3p1 está à direita de p3p4, e p3p2 está à esquerda de p3p4
- O raciocínio acima se aplica ao caso complementar de p3p4 em relação à reta que se sobrepõe a p1p2
- Contudo, se verificarmos que algum dos pontos entre dois segmentos são colineares, então devemos verificar se o ponto avaliado se sobrepõe ao segmento testado
  - Basta verificar se as coordenadas estão dentro do intervalo definido pelos dois terminais do segmento

Determinar se dois segmentos se interceptam

```
SEGMENTS-INTERSECT(p1, p2, p3, p4)
 1  d1 = DIRECTION(p3, p4, p1)
 2  d2 = DIRECTION(p3, p4, p2)
 3  d3 = DIRECTION(p1, p2, p3)
 4  d4 = DIRECTION(p1, p2, p4)
 5  if ((d1 > 0 and d2 < 0) or (d1 < 0 and d2 > 0)) and
        ((d3 > 0 and d4 < 0) or (d3 < 0 and d4 > 0))
 6      return TRUE
 7  elseif d1 == 0 and ON-SEGMENT(p3, p4, p1)
 8      return TRUE
 9  elseif d2 == 0 and ON-SEGMENT(p3, p4, p2)
10      return TRUE
11  elseif d3 == 0 and ON-SEGMENT(p1, p2, p3)
12      return TRUE
13  elseif d4 == 0 and ON-SEGMENT(p1, p2, p4)
14      return TRUE
15  else return FALSE
```

Determinar se há interseção de segmentos em um conjunto

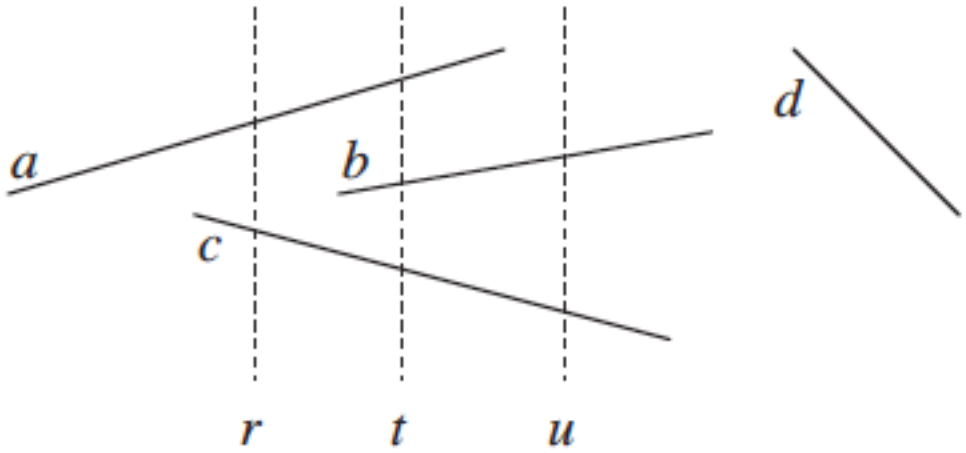
- No problema anterior, discutimos como verificar se dois dados segmentos se interceptavam
- Agora, dado um conjunto de segmentos, queremos determinar se há interseção entre quaisquer dois elementos desse conjunto
- Utilizaremos uma estratégia para resolver o problema muito comum em algoritmos de geometria computacional: a varredura linear (sweeping)
- Nessa abordagem, uma reta vertical imaginária é varrida sobre os objetos geométricos usualmente da esquerda para a direita

Determinar se há interseção de segmentos em um conjunto

- Para focar na compreensão geral do método, assumiremos duas restrições ao problema:
  - Três segmentos não se interceptam em um único ponto
  - Não há segmentos puramente verticais
- Processaremos os segmentos em ordem conforme a varredura dos pontos que os definem
- Como não há segmentos verticais, cada um intercepta a reta de varredura em um único ponto
- Definiremos uma ordem sobre os segmentos de acordo com a coordenada y de suas interseções com a reta de varredura

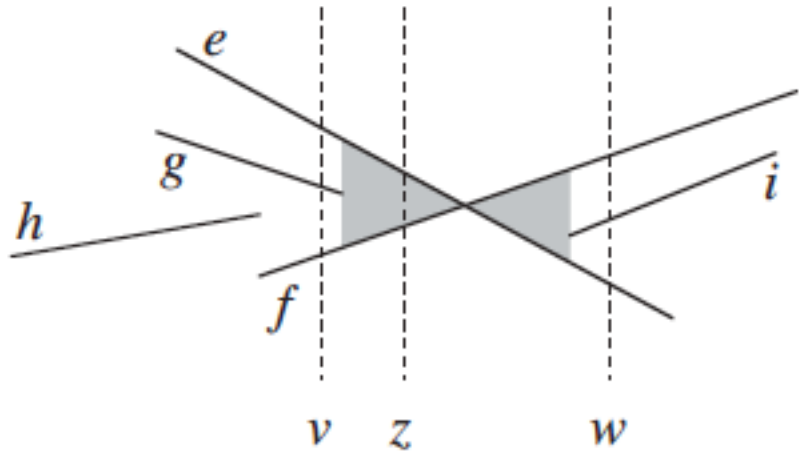
Determinar se há interseção de segmentos em um conjunto

- Dados dois segmentos s1 e s2 e a reta de varredura na coordenada x, s1 e s2 são **x-comparáveis** se a reta intercepta ambos os segmentos.
- O segmento s1 está **acima** de s2 conforme a reta x, s1 $\succcurlyeq_x$ s2, se s1 e s2 são x-comparáveis e a coordenada y da interseção de s1 com x é maior que de s2, ou ambos se interceptam em x.



Determinar se há interseção de segmentos em um conjunto

- Note que a ordem dos segmentos, segundo a relação, varia para diferentes valores de x
  - s1 $\succcurlyeq_x$ s2, mas pode ser que s2 $\succcurlyeq_{x+t}$ s1
- Os elementos entram e saem da relação à medida que a varredura avança
  - Um segmento entra na relação quando seu ponto inicial é lido, e sai quando o final é lido
- Há um valor de x para o qual s1 e s2 se tornam consecutivos
  - Nesse momento é avaliado se há interseção entre s1 e s2



Determinar se há interseção de segmentos em um conjunto

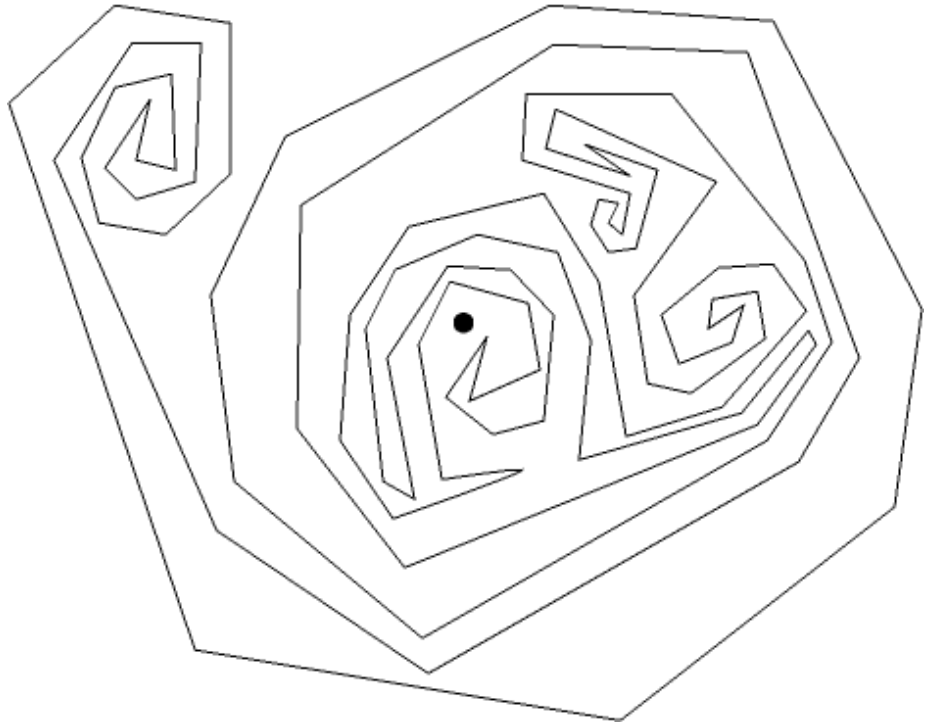
- Como a relação é dinâmica (varia durante a varredura), precisamos de uma estrutura de dados capaz de permitir a execução das seguintes operações de forma eficiente:
  - Inserir um novo segmento s
  - Remover um segmento s
  - Descobrir o sucessor de s conforme a relação de ordem
  - Descobrir o antecessor de s conforme a relação de ordem
- Essas operações podem ser executadas em tempo O(lg n) (n é o número de segmentos), usando árvores binárias balanceadas (e.g. rubro-negra)
- A varredura feita pelo algoritmo é baseada nos pontos terminais dos segmentos (chamados de eventos)
- Os eventos são ordenados pela coordenada x. Desempates são baseados se são pontos de início ou fim, e na coordenada y.

Determinar se há interseção de segmentos em um conjunto

```
ANY-SEGMENTS-INTERSECT(S)
 1  T = ∅
 2  sort the endpoints of the segments in S from left to right,
        breaking ties by putting left endpoints before right endpoints
        and breaking further ties by putting points with lower
        y-coordinates first
 3  for each point p in the sorted list of endpoints
 4      if p is the left endpoint of a segment s
 5          INSERT(T, s)
 6          if (ABOVE(T, s) exists and intersects s)
                or (BELOW(T, s) exists and intersects s)
 7              return TRUE
 8      if p is the right endpoint of a segment s
 9          if both ABOVE(T, s) and BELOW(T, s) exist
                and ABOVE(T, s) intersects BELOW(T, s)
10              return TRUE
11          DELETE(T, s)
12  return FALSE
```

Determinar se um ponto está dentro de um polígono

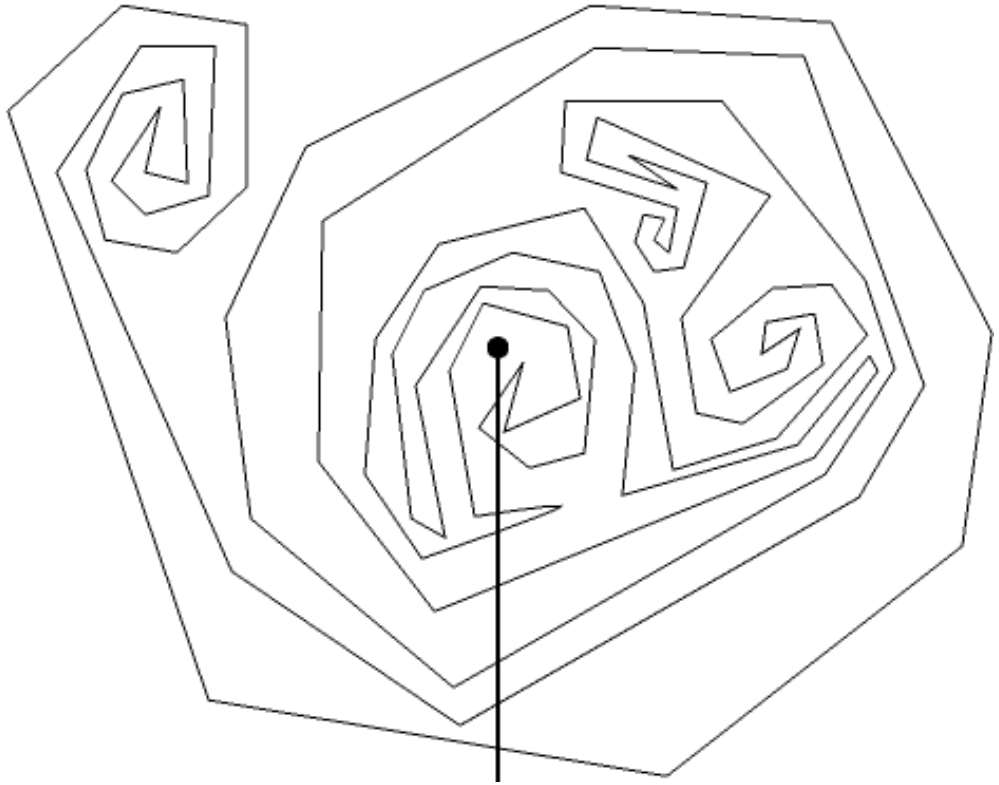
- Considere agora o problema ‘simples’ de determinar se um ponto está contido em um polígono
- Embora o problema seja muitas vezes visualmente trivial, computacionalmente ele é, de certa forma, desafiador
- A figura ao lado mostra um exemplo cuja solução requer atenção até mesmo visualmente



https://www.ics.uci.edu/~eppstein/161/960307.html

Determinar se um ponto está dentro de um polígono

- Esse problema surge em diversas aplicações, como:
  - Verificar a localização de um ponto no mapa (cidade, estado ou país)
  - Desenhar um polígono preenchido na tela
- Solução: projetar uma reta a partir do ponto e contar o número de interseções
  - Par = fora
  - Ímpar = dentro
- Casos particulares:
  - Reta toca vértice do polígono
  - Reta se sobrepõe a uma aresta do polígono



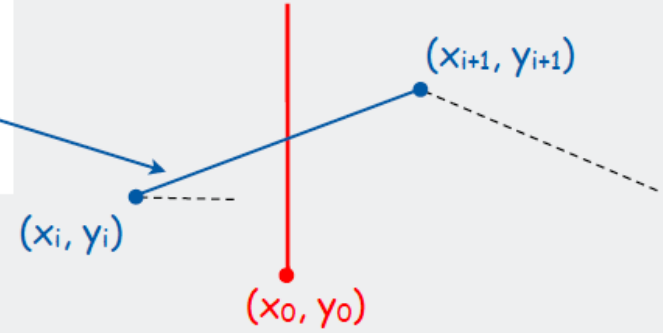
https://www.ics.uci.edu/~eppstein/161/960307.html

Determinar se um ponto está dentro de um polígono

- O uso da reta vertical simplifica a computação de interseções
- Deve-se ter cuidado com essa implementação por questões de precisão numérica
- Inclinações muito pequenas podem gerar underflow e falsos negativos

$$y_0 = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x_0 - x_i) + y_i$$

$$x_i \le x_0 \le x_{i+1}$$




```
public boolean contains(double x0, double y0)
{
    int crossings = 0;
    for (int i = 0; i < N; i++)
    {
       double  slope = (y[i+1] - y[i]) / (x[i+1] - x[i]);
       boolean cond1 = (x[i] <= x0) && (x0 < x[i+1]);
       boolean cond2 = (x[i+1] <= x0) && (x0 < x[i]);
       boolean above = (y0 < slope * (x0 - x[i]) + y[i]);
       if ((cond1 || cond2)  && above ) crossings++;
    }
    return ( crossings % 2 != 0 );
 }
```

https://www.cs.princeton.edu/~rs/AlgsDS07/16Geometric.pdf

Leitura

- Seção 33.1 e 33.2 (CLRS)
- Seção 22.1 (Goodrich e Tamassia)
- Seção 8.2 (Manber)

DCC207 – Algoritmos 2

Aula 04 – Introdução a Geometria Computacional (Parte 01)

Professor Renato Vimieiro

DCC/ICEx/UFMG