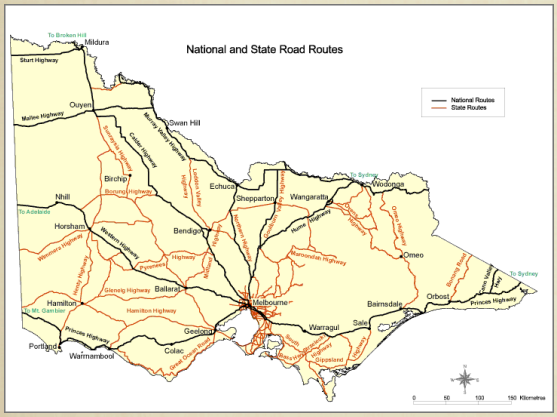
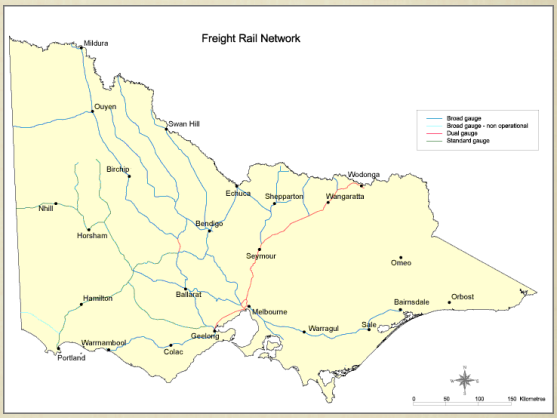
Interseções de Segmentos de Linha

João Comba

Sobreposição de Mapas Temáticos

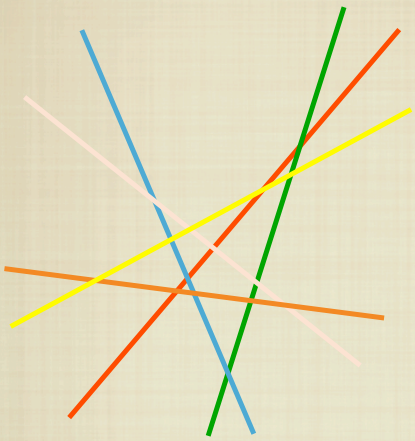




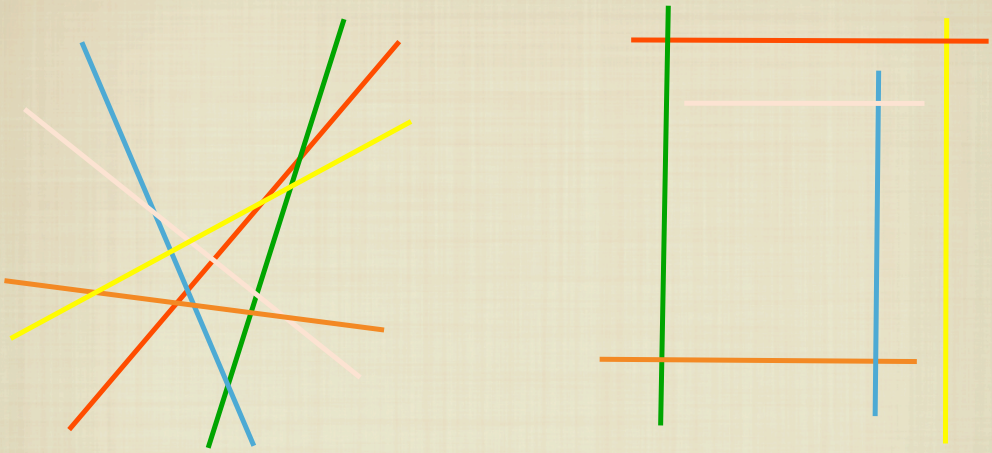
Mapas temáticos

- **Segmentos curvos aproximados por vários segmentos de linha**
- **Problema de Sobreposição (Overlay):**
  - Dados dois conjuntos de segmentos de linha, calcular todas interseções entre o segmento de um conjunto e um segmento do outro conjunto
    - $O(n^2)$?
  - Colocar todos segmentos em um mesmo conjunto (simplificar a descrição da solução)

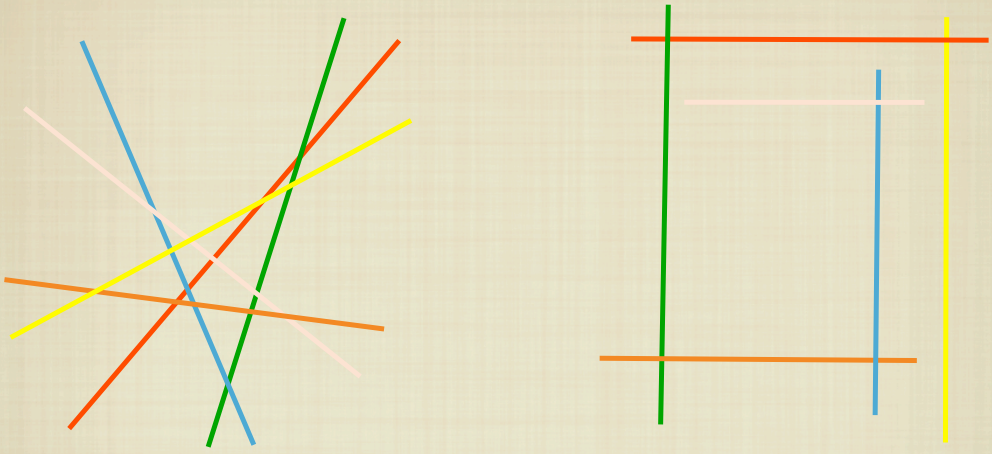
Casos Extremos



Casos Extremos



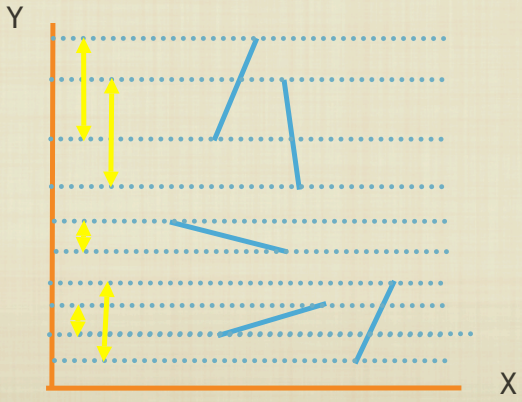
Casos Extremos



Algoritmo sensível a saida (OUTPUT sensitive)

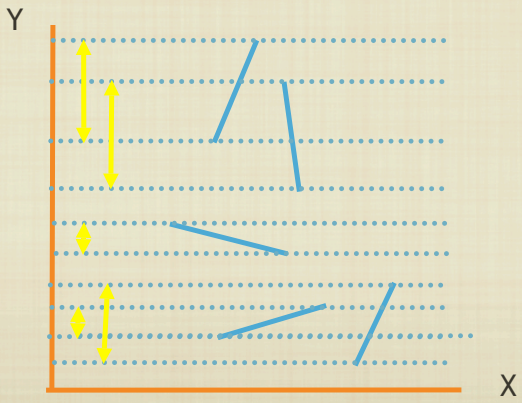
Como evitar testar todos os pares ?

- Segmentos próximos são candidatos mais fortes a interseção



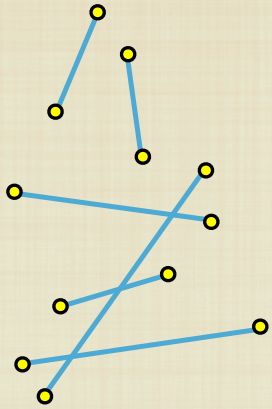
Como evitar testar todos os pares ?

- Segmentos próximos são candidatos mais fortes a interseção



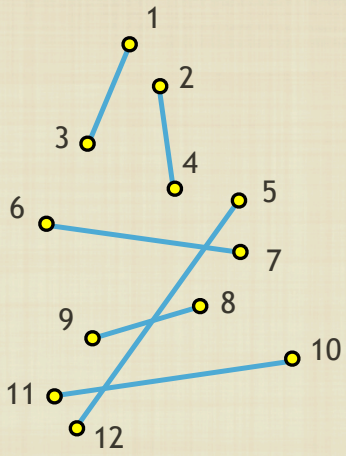
Problema: Como enumerar todos pares candidatos a interseção ?

Plane Sweep



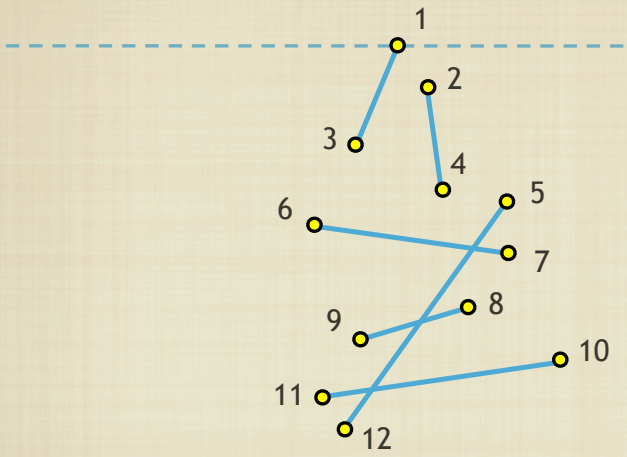
Ordenar os pontos na coordenada Y

Plane Sweep

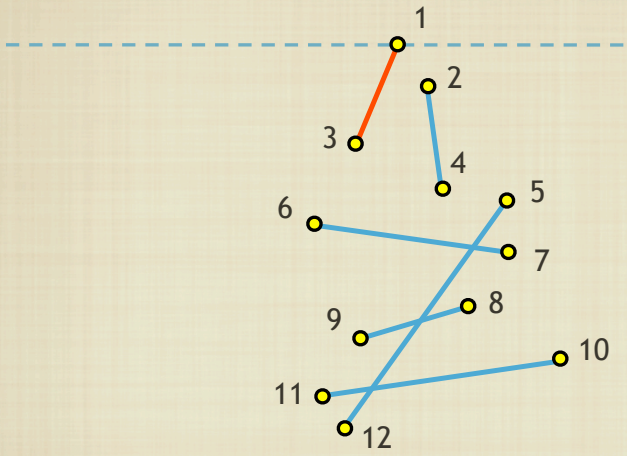


Ordenar os pontos na coordenada Y

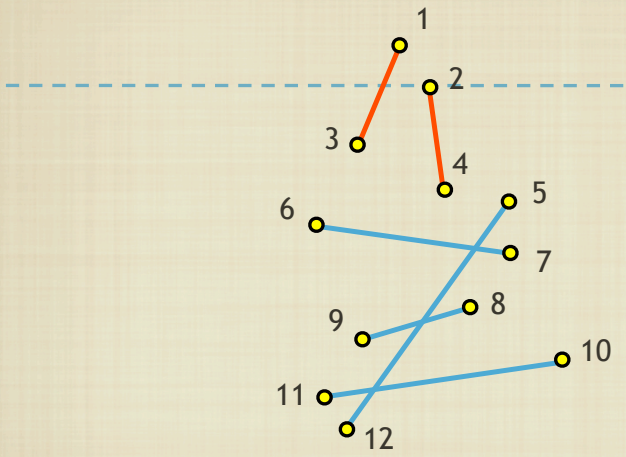
Plane Sweep



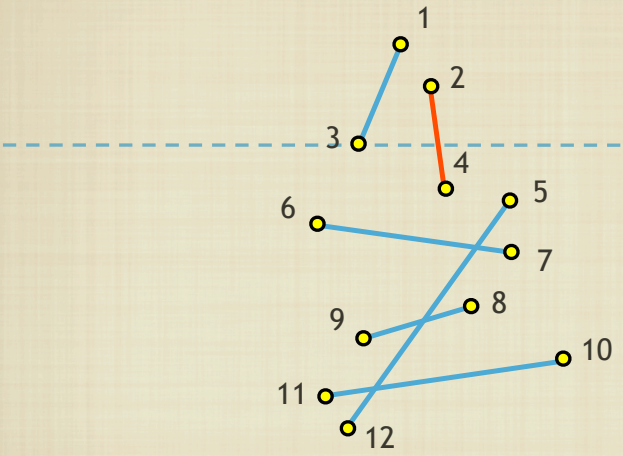
Plane Sweep



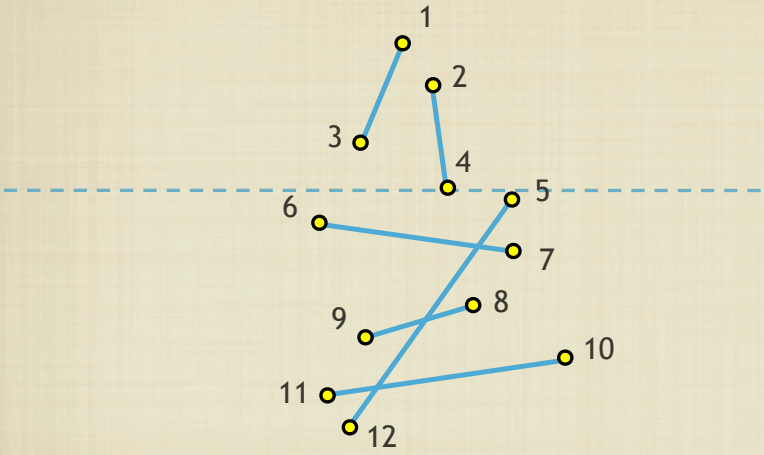
Plane Sweep



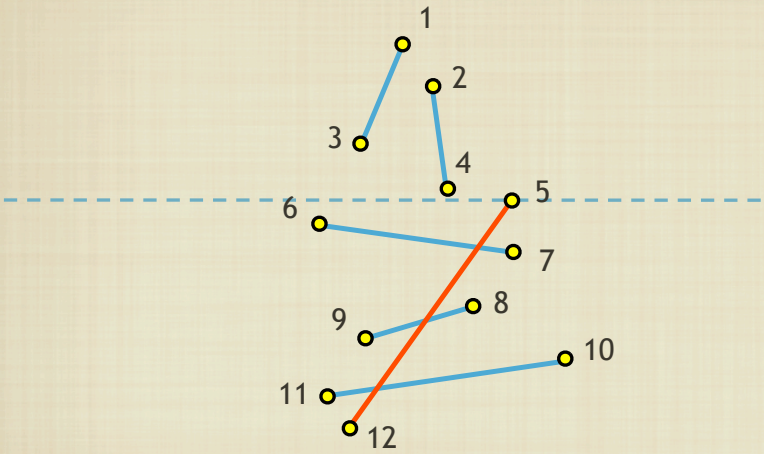
Plane Sweep



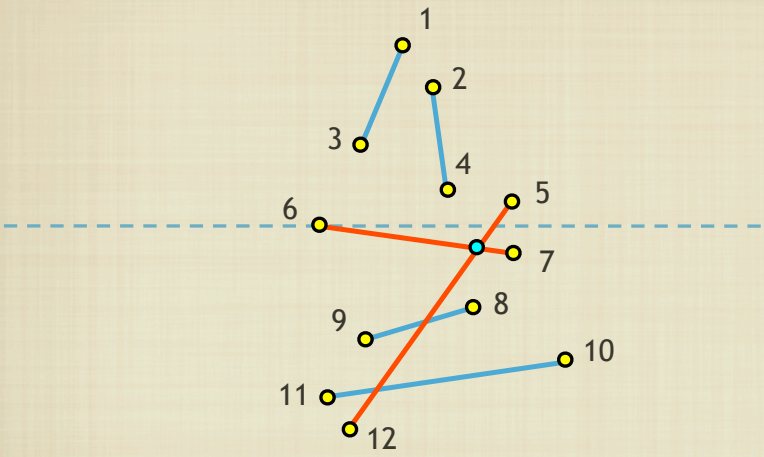
Plane Sweep



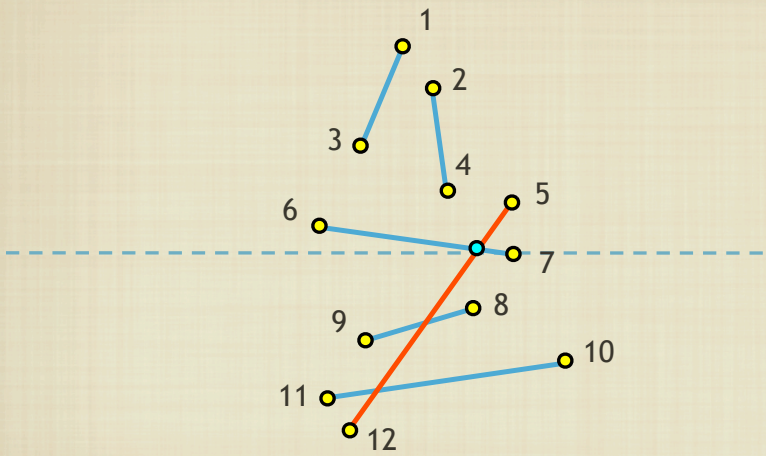
Plane Sweep



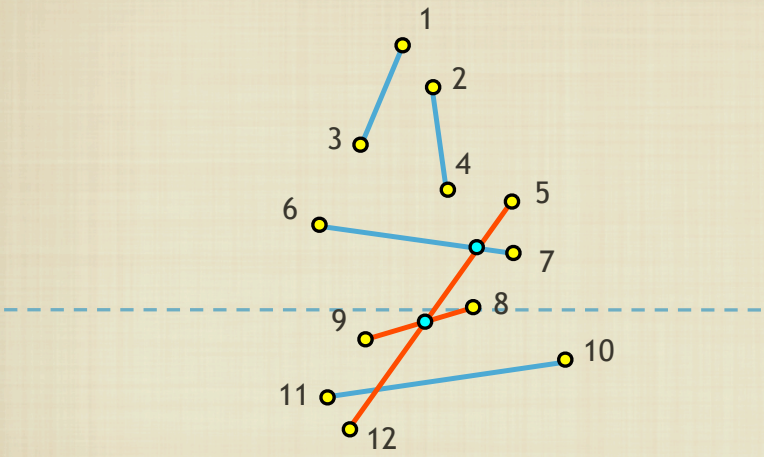
Plane Sweep



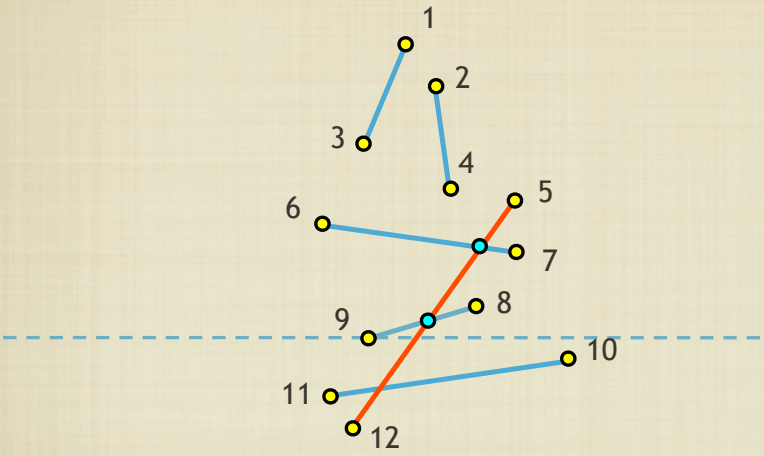
Plane Sweep



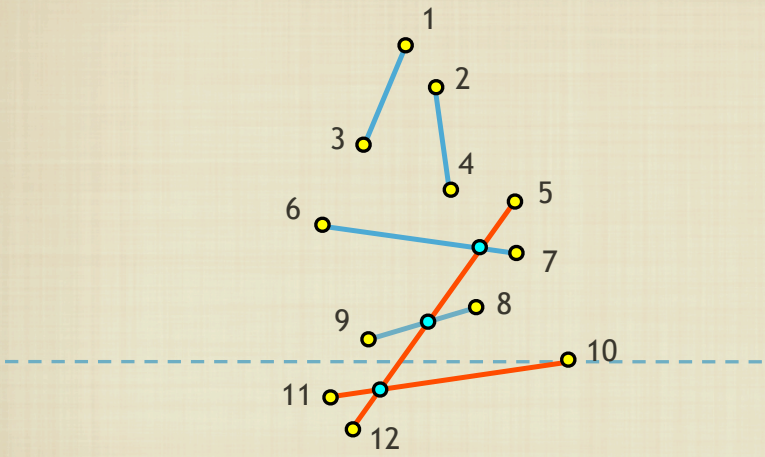
Plane Sweep



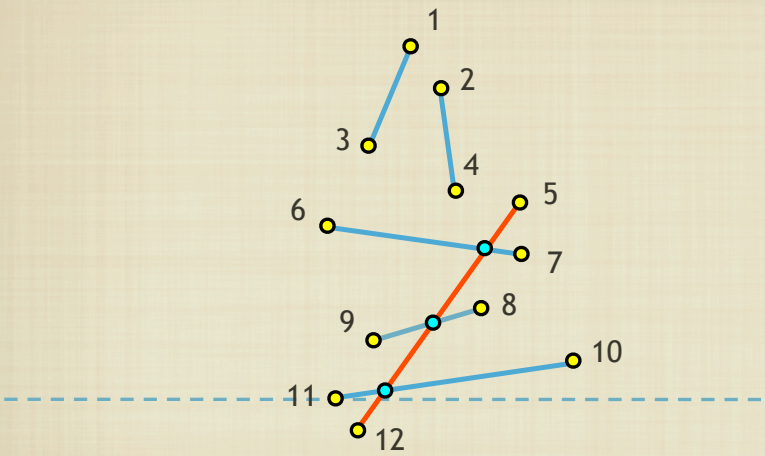
Plane Sweep



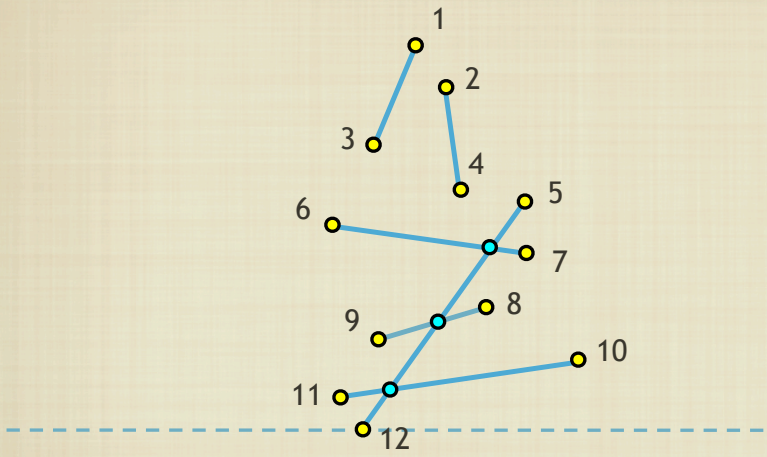
Plane Sweep



Plane Sweep

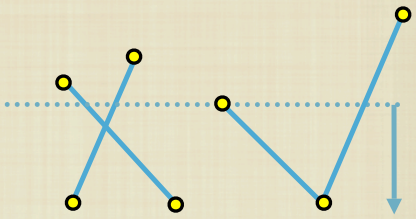


Plane Sweep



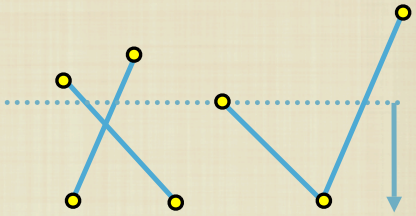
Plane Sweep

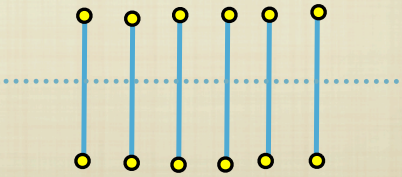
- Linha de Sweep
- Status da linha de sweep
- Eventos:
  - quando ?
    - pontos superior/inferior



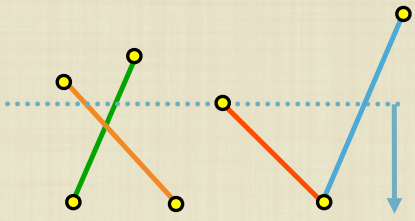
Plane Sweep

- Linha de Sweep
- Status da linha de sweep
- Eventos:
  - quando ?
    - pontos superior/inferior
  - interseção ? testar todos segmentos
  - atualizar linha de sweep
- Complexidade ?

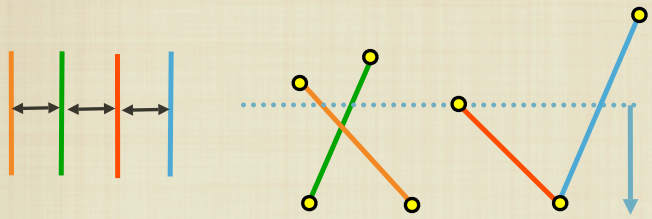




Ordernar segmentos na sweep line

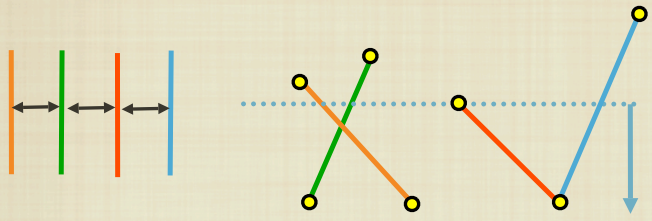


Ordernar segmentos na sweep line



L

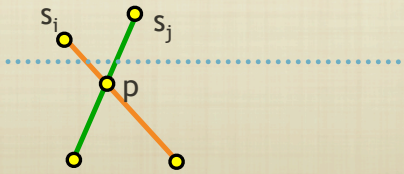
Ordernar segmentos na sweep line



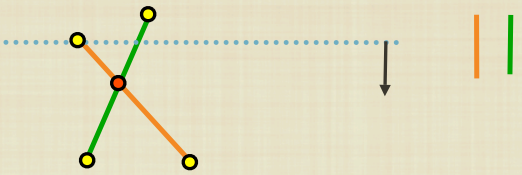
L

Solução é correta ?

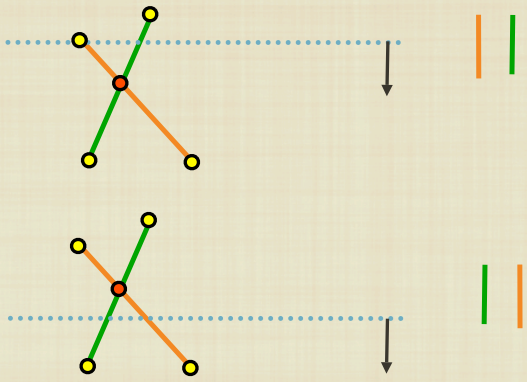
- Seja s_i e s_j dois segmentos não horizontais que se interceptam em um ponto p
- Assuma que não haja um terceiro segmento passando por p.
- LEMA: Existe um evento acima de p onde s_i e s_j se tornam adjacente e sua interseção é testada



Processando

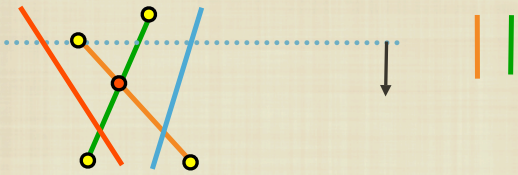


Processando Interseções

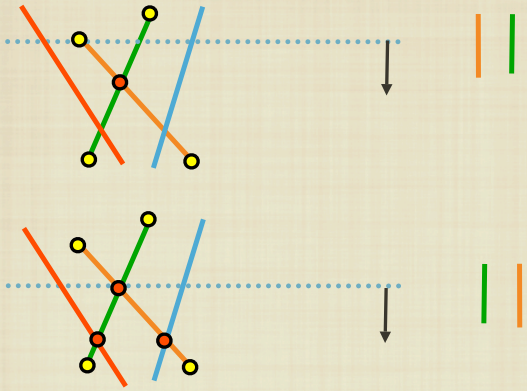


Trocar Ordem requer processar eventos em pontos de interseção

Processando Interseções

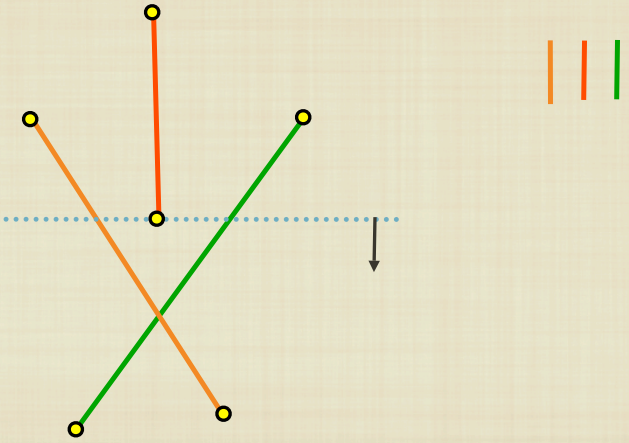


Processando Interseções

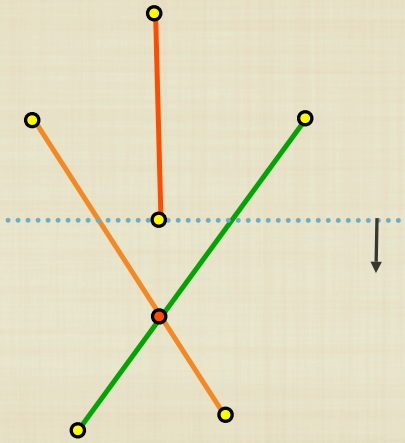


Novas interseções podem ser criadas pois segmentos que antes não eram adjacentes se tornam após o ponto de interseção

Processando Interseções



Processando Interseções



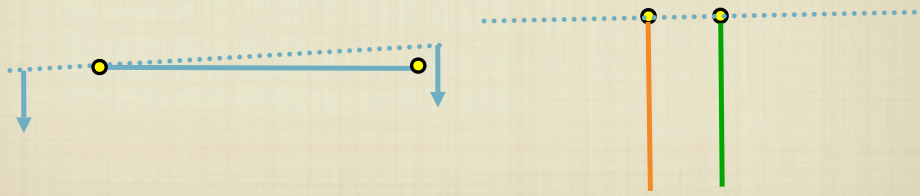
Novas Interseções

Casos Especiais

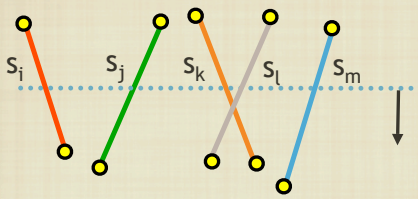
- Três segmentos interceptando em um ponto:
  - Reportar para cada ponto de interseção os segmentos passando por ele
- Dois segmentos parcialmente sobrepostos

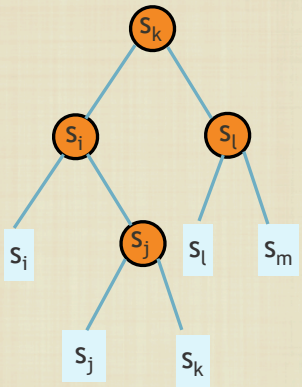
Estruturas de Dados

- **Fila de eventos (event queue) Q:**
  - Escolher evento com maior coordenada Y, coordenada X como desempate

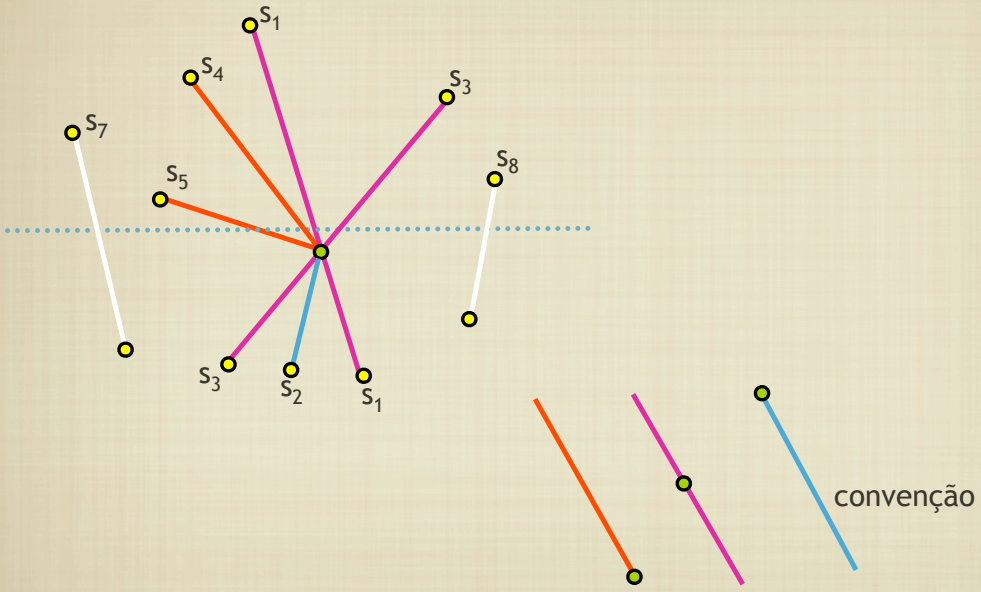


Estruturas de Dados





Exemplo



FindIntersections(S)

Algoritmo FindIntersections(S):
Entrada: Conjunto de segmentos de linha S no plano
Saída: Conjunto de pontos de interseção, contendo para cada ponto o segmento que o contém

1. Inicializar uma fila de eventos Q, inserir os vértices dos segmentos em Q, associar o segmento ao vértice superior
2. Inicializar como vazia uma árvore de status T
3. WHILE Q não está vazia
4. DO Determinar o próximo evento p em Q e deletar de Q
   HandleEventPoint(p)

HandleEventPoint

Algoritmo HandleEventPoint(p)
Entrada: Conjunto de segmentos de linha S no plano
Saída: Conjunto de pontos de interseção

1. Seja U(p) segmentos que p é vértice superior (para segmentos horizontais é o ponto mais a esquerda). Eles são armazenados com p na fila de eventos
2. Buscar em T o conjunto S(p) de todos segmentos que contém p (eles são adjacentes em T).
   Seja L(p) C S(p) o conjunto de segmentos nos quais p é o vértice inferior
   Seja C(p) C S(p) o conjunto de segmentos que contém p no seu interior

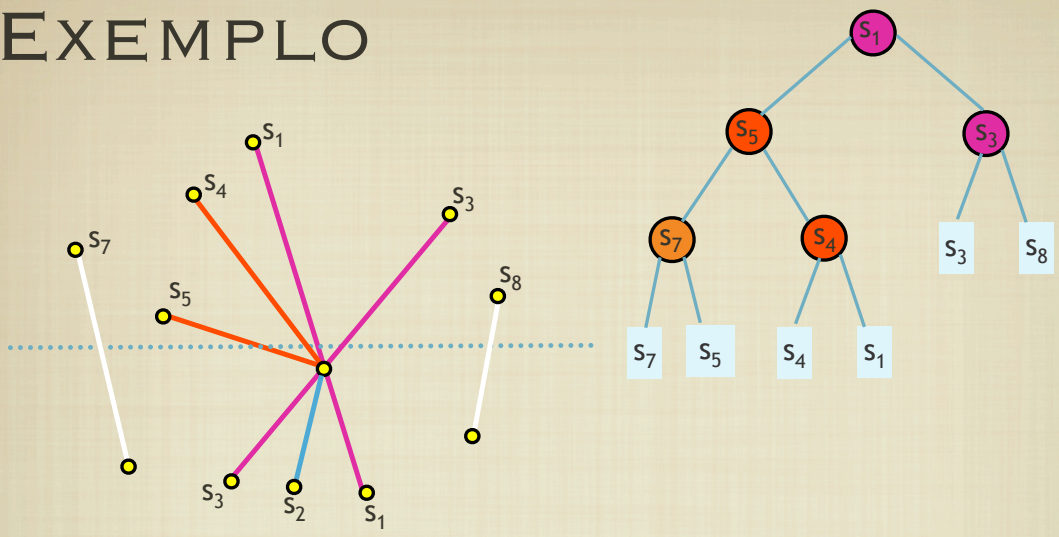
HandleEventPoint

3. IF L(p) U U(p) U C(p) contém mais de um segmento
   THEN Reportar p junto com L(p), U(p) e C(p)
4. Deletar os segmentos em L(p) U C(p) de T
5. Inserir os segmentos em U(p) U C(p) em T
6. // Deletar e re-inserir C(p) reverte sua ordem
7. IF U(p) U C(p) = Ø
8. THEN Seja s_l e s_r os segmentos vizinhos a p em T
9. FindNewEvent(s_l, s_r, p)
10. ELSE Seja s' o segmento mais a esquerda de U(p)UC(p)
11. Seja sl o vizinho esquerdo de s'em T
12. FindNewEvent(s_l, s', p)
13. Seja s'' o segmento mais a direita de U(p)UC(p)
14. Seja s_r o vizinho direito de s' em T
15. FindNewEvent(s'', s_r, p)

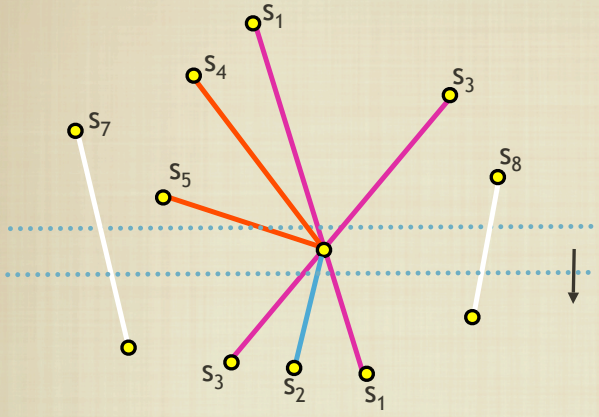
FindNewEvent(S_L, S_R, P)

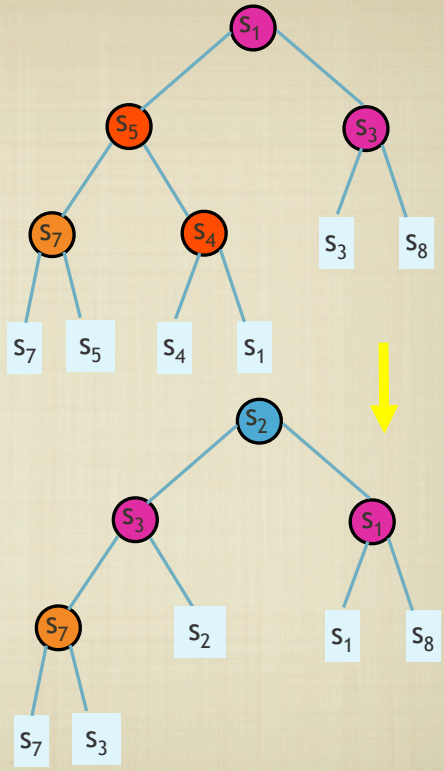
1. IF s_l e s_r se interceptam abaixo da linha de sweep (ou sobre ela e a direita do ponto evento p), e a interseção ainda não está presente em Q
2. THEN Inserir o ponto de interseção como evento em Q

Exemplo

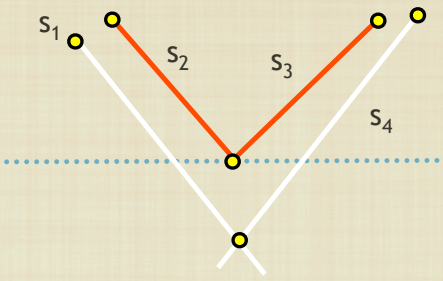


Exemplo



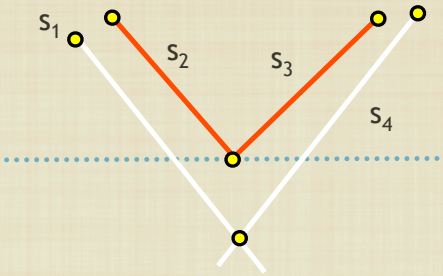


Exemplo



(s1,s2,s3,s4)

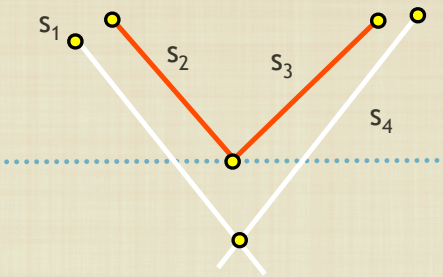
Exemplo



(s1,s2,s3,s4) → (s1, s4)

IF U(p) U C(p) = Ø
THEN Seja sl e sr os segmentos vizinhos a p em T
FindNewEvent(sl, sr, p)

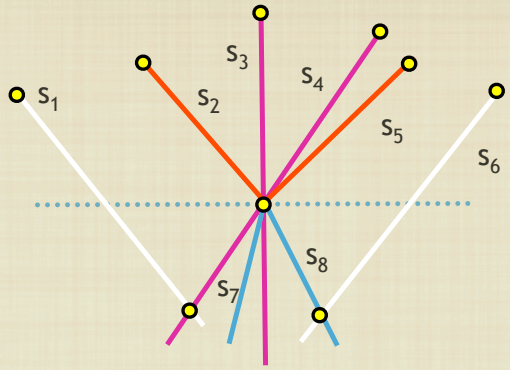
Exemplo



(s1,s2,s3,s4) → (s1, s4)

IF U(p) U C(p) = Ø
THEN Seja sl e sr os segmentos vizinhos a p em T
FindNewEvent(sl, sr, p)

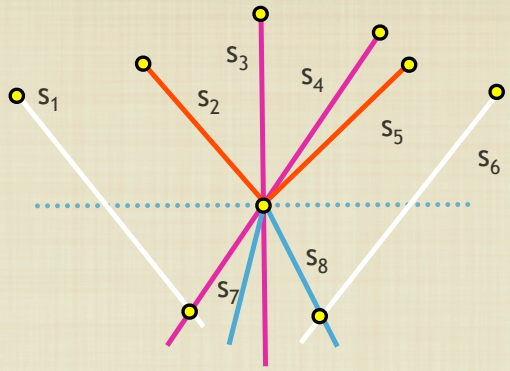
Exemplo



(s1,s2,s3,s4,s5,s6)

ELSE Seja s' o segmento mais a esquerda de U(p)UC(p)
Seja sl o vizinho esquerdo de s'em T
FindNewEvent(sl, s', p)
Seja s'' o segmento mais a direita de U(p)UC(p)
Seja sr o vizinho direito de s' em T
FindNewEvent(s'', sr, p)

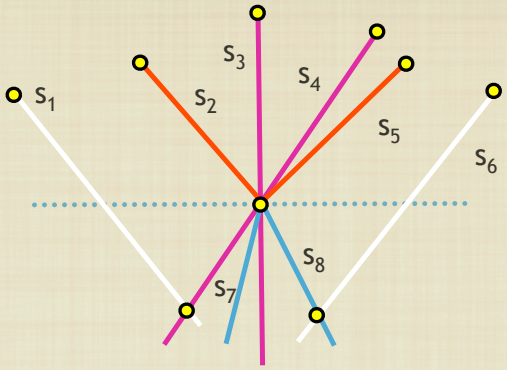
Exemplo



(s1,s2,s3,s4,s5,s6) → (s1,s4,s7,s3,s8,s6)

ELSE Seja s' o segmento mais a esquerda de U(p)UC(p)
Seja sl o vizinho esquerdo de s'em T
FindNewEvent(sl, s', p)
Seja s'' o segmento mais a direita de U(p)UC(p)
Seja sr o vizinho direito de s' em T
FindNewEvent(s'', sr, p)

Exemplo



(s1,s2,s3,s4,s5,s6) → (s1,s4,s7,s3,s8,s6)

ELSE Seja s' o segmento mais a esquerda de U(p)UC(p)
Seja sl o vizinho esquerdo de s'em T
FindNewEvent(sl, s', p)
Seja s'' o segmento mais a direita de U(p)UC(p)
Seja sr o vizinho direito de s' em T
FindNewEvent(s'', sr, p)

Complexidade

- LEMA. A complexidade do algoritmo FindIntersection para um conjunto n de segmentos de linha S no plano é :
- O((n+I)logn)

onde I é o número de pontos de interseção em S

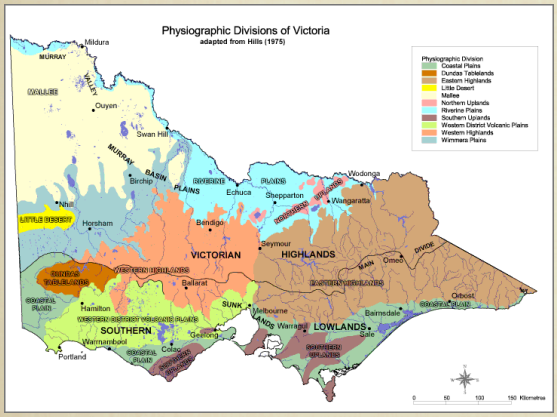
Prova

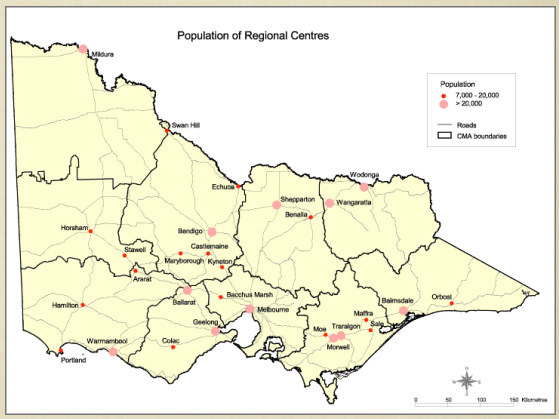
- O(nlogn) para construir uma árvore binária balanceada com os eventos (fila Q)
- Processando um evento:
  - deletar o evento de Q O(log n)
  - 1 ou 2 chamadas para FindNewEvent (podem inserir até 2 eventos em Q) O(log n)
  - Número de inserções/deleções:
    - linear em m(p), grau do vertice, = card(L(p)U U(p) U C(p))
    - $m = \sum m(p)$
  - Complexidade = número de ins/del * O(logn) → O(mlogn)
  - Provar que m = O(n+I) → O((n+I)logn) → O(nlogn + Ilogn)

Prova

- Considere segmentos de linha subdivisões do plano
- Considere um evento p, m(p) é o grau do vértice
  - m é limitado pela soma dos graus de todos os vértices
- Cada aresta aumenta em 1 o grau de 2 vértices
  - m é limitado por $2N_E$
- Limitar N_E em termos de n e I
  - N_V é 2N + I (por definição)
- Cada aresta limita duas faces, e precisam de pelo menos três para formar uma face:
  - $N_F \leq 2N_E/3$
- EULER: $N_V - N_E + N_F \geq 2$ (= se o grafo é conectado)
  - $2 \leq (2N+I) - N_E + 2N_E/3$
  - $2 \leq (2N+I) - N_E/3$
  - $N_E \leq 6N+3I - 6$
  - $m \leq 12N + 6I - 12$ → O(n+I)

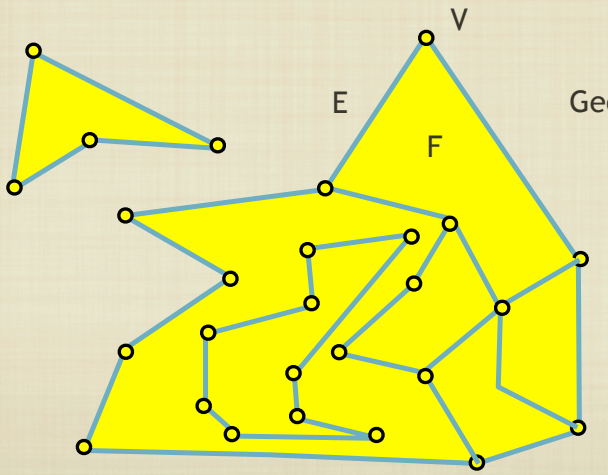
Sobreposição de Regiões





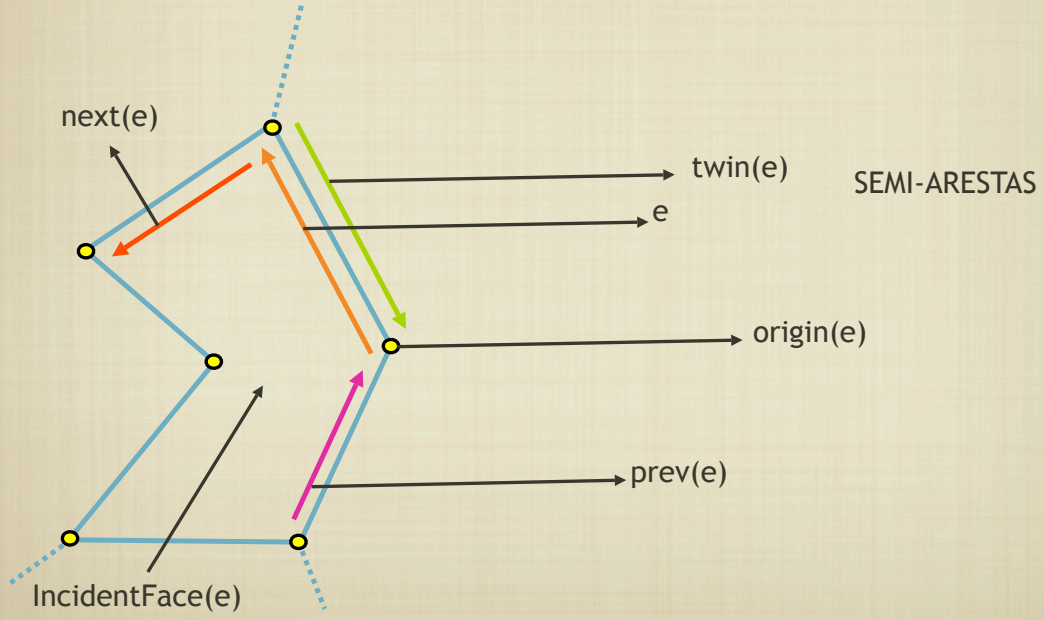
Subdivisões Planares

- **Mapas: representados por grafos planares**



Geometria x Topologia

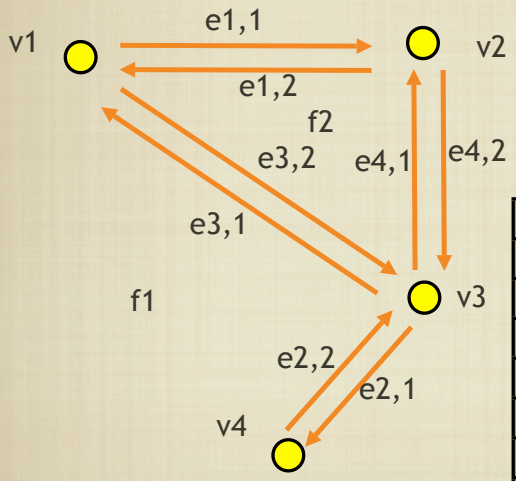
Lista de Arestas



Lista de Arestas Duplamente Conectadas

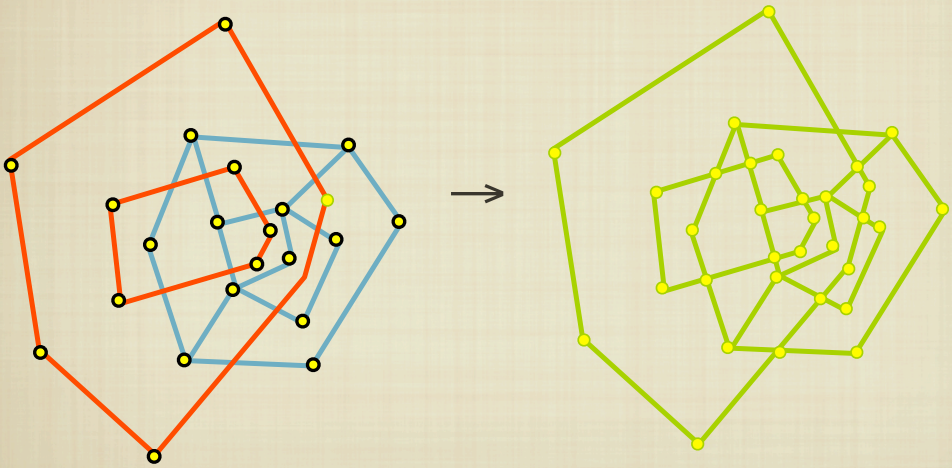
Vértice	Coord	Aresta
v1	(0,4)	e1,1
v2	(2,4)	e4,2
v3	(2,2)	e2,1
v4	(1,1)	e2,2

Face	Out	In
f1	NULL	e1,1
f2	e4,1	NULL



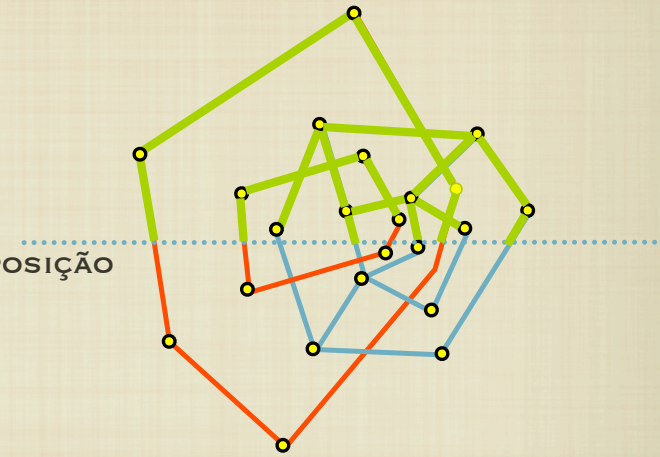
Aresta	Orig	Twin	Incid	Next	Prev
e1,1	v1	e1,2	f1	e4,2	e3,1
e1,2	v2	e1,1	f2	e3,2	e4,2
e2,1	v3	e2,2	f1	e2,2	e4,2
e2,2	v4	e2,1	f1	e3,1	e2,1
e3,1	v3	e3,2	f1	e1,1	e2,2
e3,2	v1	e3,1	f2	e4,1	e1,2
e4,1	v3	e4,2	f2	e1,2	e3,2
e4,2	v2	e4,1	f1	e2,1	e1,1

Sobreposição de Duas Subdivisões



Sobreposição S1 e S2

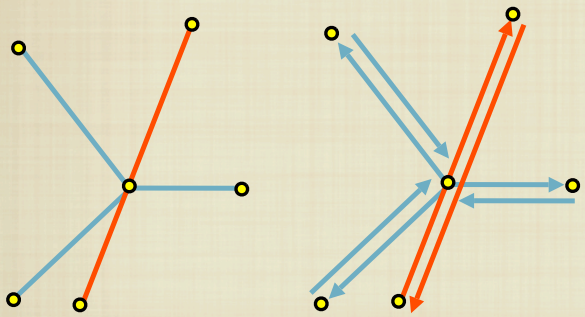
- Copiar S1 e S2 em S
  - S não é válida
- Tornar S válida
  - Algoritmo de Sobreposição
- Como fazer ?
  - interseções
  - semi-arestas
  - faces



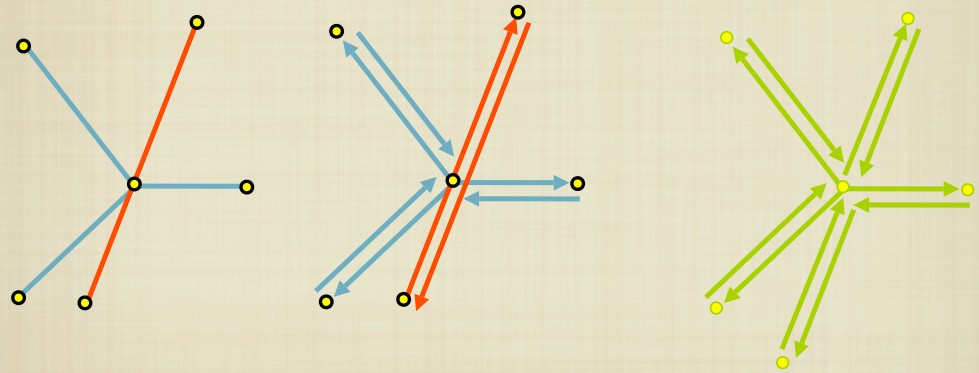
Abordagem Sweep

1. Aplicar o algoritmo de interseções de segmentos de linha ao conjunto de segmentos de S1 e S2
2. Estruturas de dados:
   - D lista de arestas duplamente encadeadas
   - Q fila de eventos
   - T segmentos que cruzam a linha de sweep
3. Corrigir D

Processando um evento



Processando um evento



MapOverlay(S1,S2)

Algoritmo MapOverlay(S1,S2):
Entrada: Duas subdivisões planares S1, S2
Saída: Sobreposição de S1 e S2 armazenados em D

1. Copiar as listas duplamente encadeadas de S1, S2 em D
2. Calcular interseções entre arestas de S1 e S2
   - atualizar D quando arestas do evento vem de S1 e S2
   - armazenar a semi-aresta imediatamente a esquerda de um ponto evento no vértice de D
3. Determinar os ciclos de fronteira percorrendo D
4. Construir um grafo G cujos nodos representam os componentes conexos em D
5. FOR EACH componente conexo em G
6. DO Atualizar registros de faces
7. Nomear faces gerada com nomes das faces de S1 e S2

Operações booleanas