Geometria Computacional

Professor:

Anselmo Montenegro
www.ic.uff.br/~anselmo

Conteúdo (aula 9):

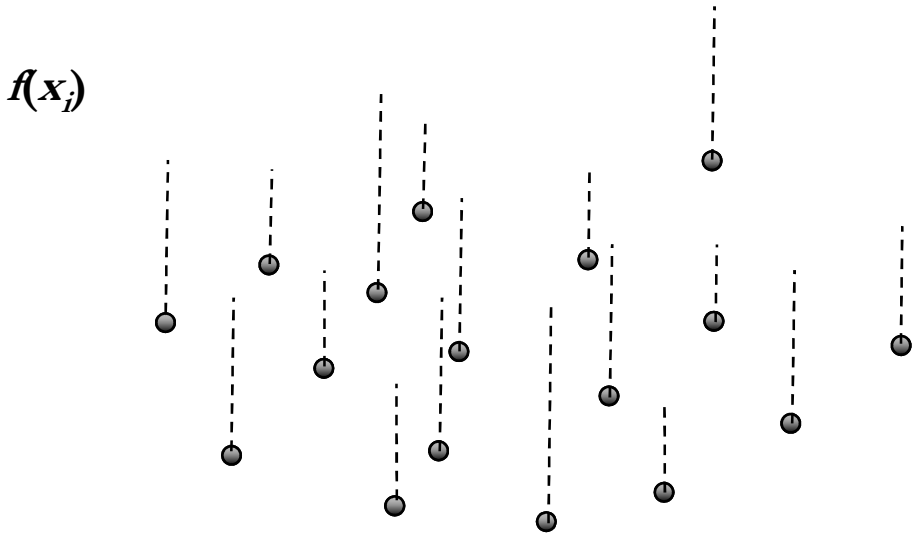
- Triangulações

Roteiro

- Introdução
- Construções fundamentais
- Algoritmo por subdivisão
- Algoritmo Incremental
- O grafo de inversão de arestas (*flip graph*)

Triangulações: introdução

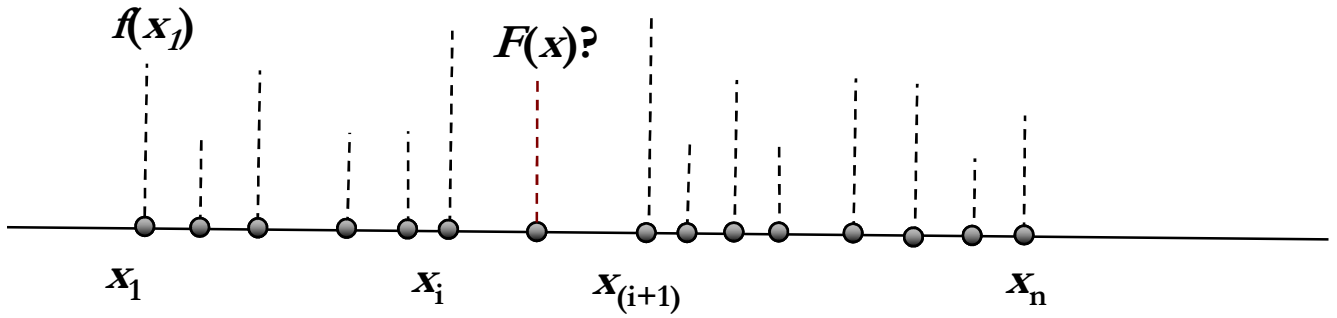
- Considere o seguinte problema: dado um conjunto de pontos $S=\{x_1,x_2,...,x_n\}$, $x_i \in R^2$ e uma função $f:S \rightarrow R$, determinar uma função F, linear por partes, definida sobre um domínio $D \supseteq S$ e tal que $F(x_i)=f(x_i)$



• Triangulações: introdução

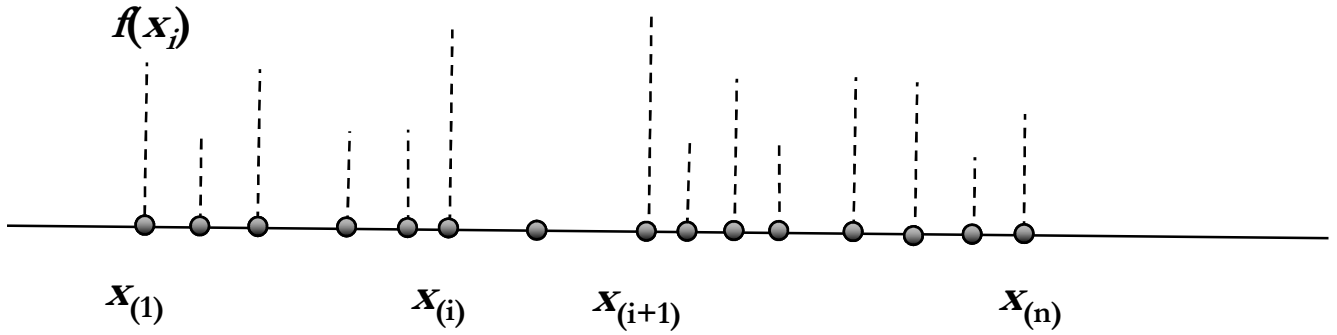
- Para ganhar intuição vamos resolver o problema no caso unidimensional

- Problema: dado um conjunto de pontos $S=\{x_1,x_2,...,x_n\}$, $x_i \in R$ e uma função $f:S \rightarrow R$, obter uma função F, linear por partes, definida sobre um domínio $D = [m,n]$ onde $m = \min(S)$ e $n = \max(S)$



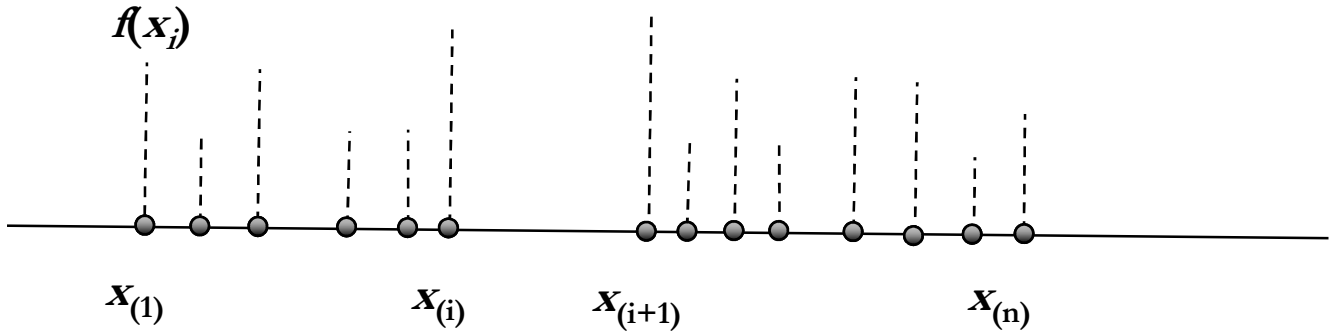
• Triangulações: introdução

- No caso unidimensional a solução é muito simples: interpolar linearmente por partes a função $f(x_i)$
- Para isto, primeiramente estruturamos o conjunto S onde a permutação dos valores originais x_i é representada por $x(i)$



• Triangulações: introdução

- A ordenação determina duas propriedades:
  - O pontos extremos do domínio $D = [m,n]$ onde $m = \min(S)$ e $n = \max(S)$que contém S
  - Uma partição de D, $\cup I_i$, $1 \leq i \leq n-1$, $I_i = [x(i), x(i+1)]$



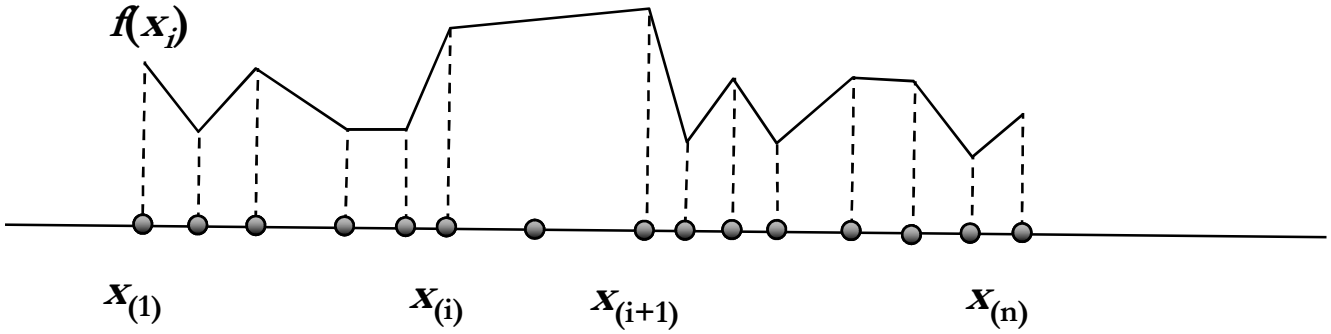
• Triangulações: introdução

- Para cada intervalo $I_i = [x_{(i)}, x_{(i+1)}]$ exprimimos $x \in I_i$ como

$$x = (1-\lambda_i)x_{(i)} + \lambda_i x_{(i+1)},\ 0 \leq \lambda_i \leq 1$$

- Deste modo, podemos expressar a função $F(x)$, $x \in$ Ii como

$$F(x) = (1-\lambda_i)\, f(x_{(i)}) + \lambda_i f(x_{(i+1)}),\ 0 \leq \lambda_i \leq 1$$



• Triangulações: introdução

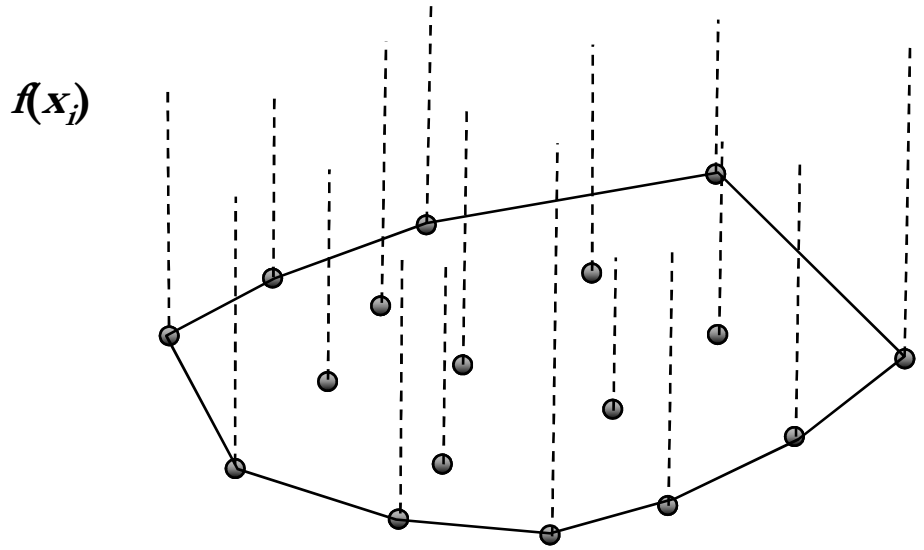
- Podemos observar que a ordenação é o passo fundamental para fornecer a estrutura ao conjunto de pontos, para que seja possível reconstruir a função em todo o domínio

- Que tipo de processamento produz estrutura semelhante para um conjunto de pontos $S \subset R^n$

- Estudaremos o caso em que $S \subset R^2$

• Triangulações: introdução

- Podemos definir D como o menor conjunto convexo que contém S, isto é, conv(S)



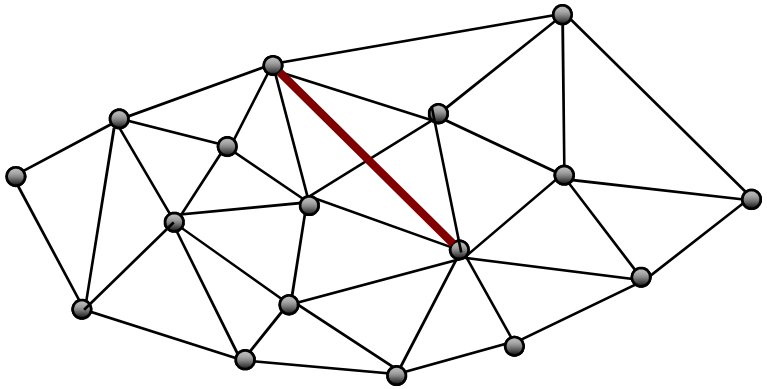
- O conv(S) faz o papel análogo aos extremos do intervalo de definição de S no caso unidimensional

- Porém não sabemos como definir uma partição de D análoga

• Triangulações: introdução

- Definição (Triangulação de um conjunto de pontos S) - Uma triangulação de um conjunto planar de pontos S é uma subdivisão do plano determinada por um conjunto maximal de arestas que não se intersectam e cujos vértices estão em S

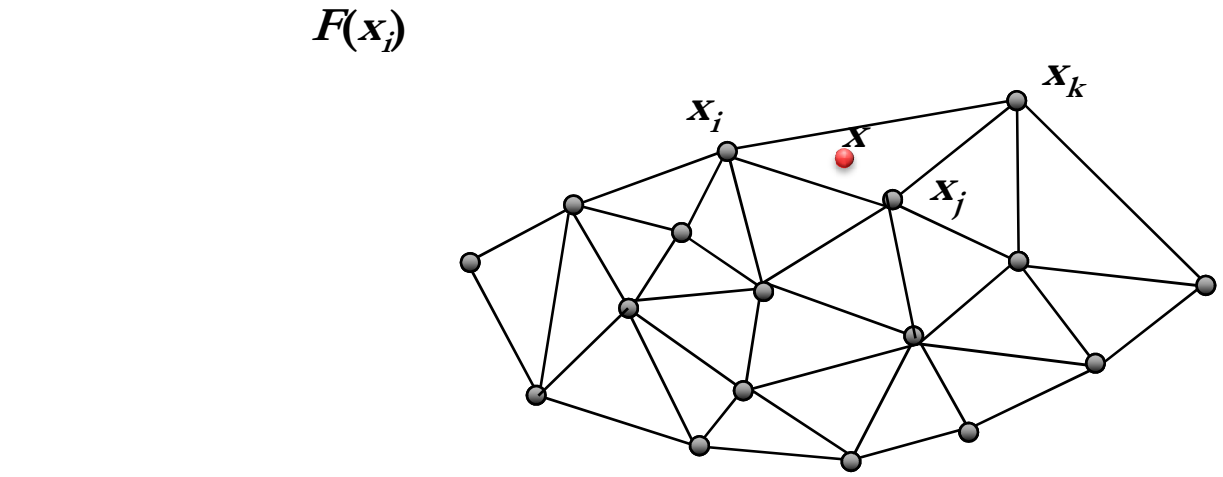
$F(x_i)$



- Obs.: maximal significa que qualquer aresta que não esteja na triangulação deve intersectar o interior de alguma aresta da triangulação

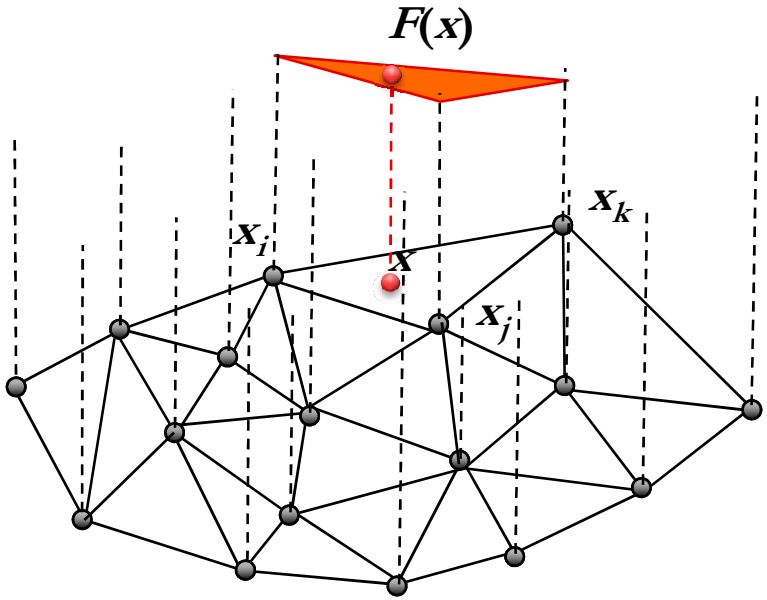
• Triangulações: introdução

- Uma vez obtida a triangulação podemos expressar um ponto $x \in \Delta x_i x_j x_k$ como $x = \lambda_i x_i + \lambda_j x_{j+} \lambda_j x_{k,}$ $\lambda_i + \lambda_j + \lambda_k = 1$usando coordenadas baricêntricas



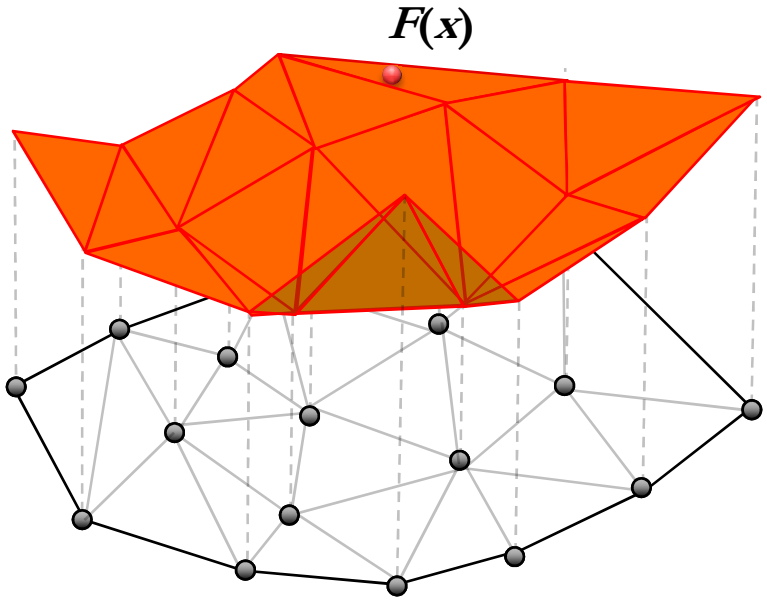
• Triangulações: introdução

- Consequentemente, podemos espressar F(x), $x \in \Delta x_i x_j x_k$ como $F(x) = \lambda_i f(x_i) + \lambda_j f(x_j) + \lambda_j f(x_k)$, $\lambda_i + \lambda_j + \lambda_k = 1$



• Triangulações: introdução

- Consequentemente, podemos espressar F(x), $x \in \Delta x_i x_j x_k$ como $F(x) = \lambda_i f(x_i) + \lambda_j f(x_j) + \lambda_j f(x_k)$, $\lambda_i + \lambda_j + \lambda_k = 1$



- **Triangulações: introdução**

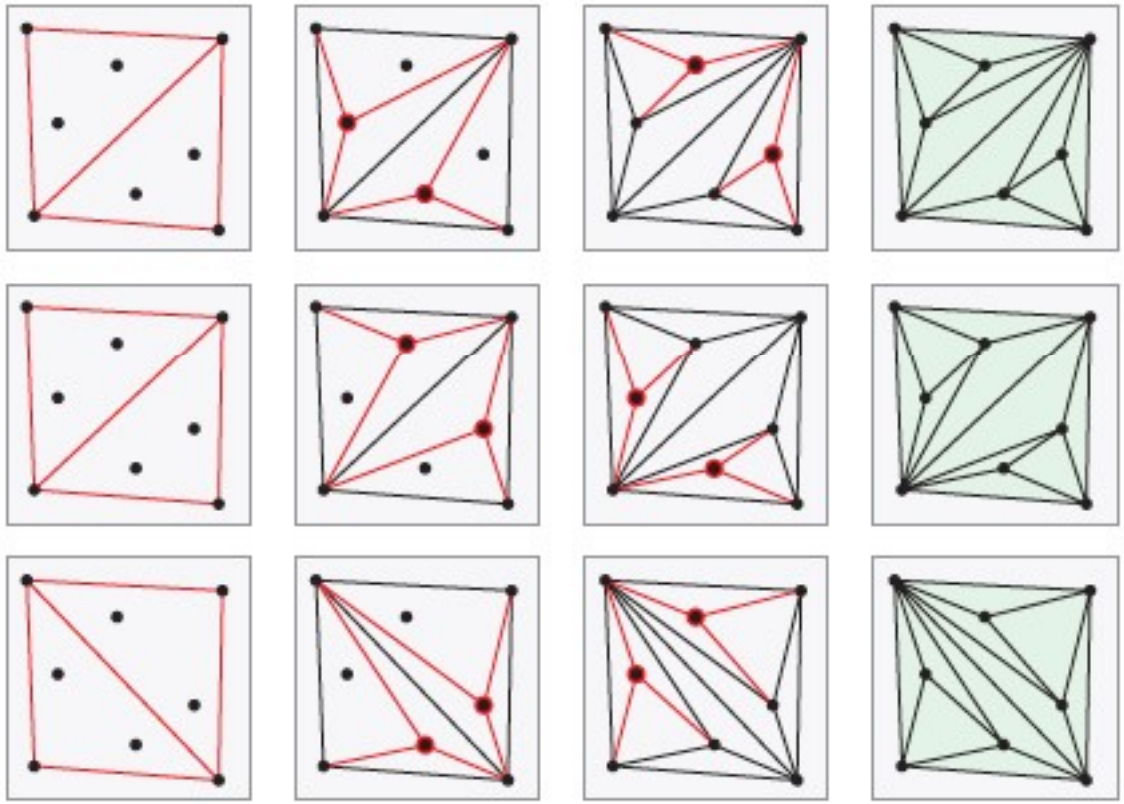
  - A triangulação de um conjunto de pontos *S* não é única

  - Antes de apresentar algumas propriedades sobre triangulações iremos apresentar alguns algoritmos

- **Triangulações: algoritmo por subdivisão**

Algorithm **TriangulaçãoPorSubdivisão(*S*)**
Input. Um conjunto de pontos S no plano.
Output. Triangulação de S
1.Computar conv(S)
2.Triangular a fronteira de conv(S) (ver aula sobre polígonos)
3.Para cada ponto p∈S que não está na fronteira de S
4. Determinar o triângulo *t* que contém
5. Construir as arestas de p para os três vértices de *t*

- **Triangulações: algoritmo por subdivisão**



- **Triangulações: algoritmo por subdivisão**

Lema 7.1 (Número de triângulos gerados pelo Algoritmo de Triangulação por Subdivisão) – Seja k o número de pontos no interior do conjunto de pontos S e h o número de pontos na fronteira de conv(S). Se nem todos os pontos forem colineares então, qualquer triangulação obtida com o Algoritmo por subdivisão tem exatamente $2k + h - 2$ triângulos.

- **Triangulações: algoritmo por subdivisão**

Prova. Pode ser demonstrado que a triangulação do fecho convexo de *S* tem *h*-2 triângulos.

Caso 1 - Se um ponto interior está dentro de um triângulo então o algoritmo conecta tal ponto aos 3 vértices do triângulo que o contém. Logo, tal ponto interior substitui um triângulo por 3 aumentando o número de triângulos em 2.

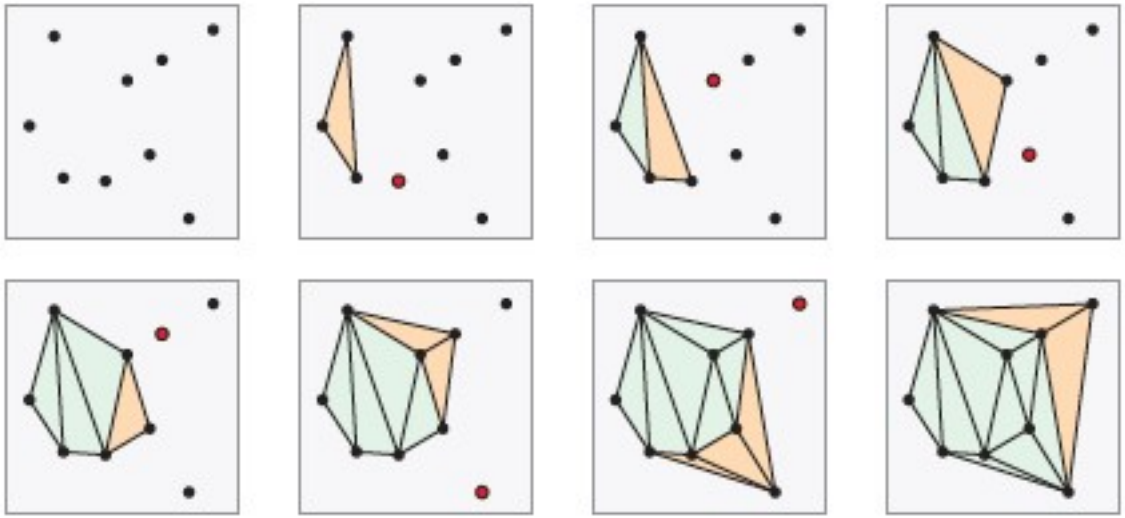
Caso 2- Se um ponto interior cai sobre uma aresta, o algoritmo conecta este ponto aos dois vértices dos triângulos incidentes a aresta em questão.

Logo, tal ponto subdivide cada triângulo dos dois lados em dois triângulos, novamente acrescentando 2 triângulos ao total. Como existem *k* pontos interiores em S, o número de triângulos resultantes é $2k + h - 2$.

- **Triangulações: algoritmo incremental**

Algorithm **TriangulaçãoIncremental(*S*)**
Input. Um conjunto de pontos S no plano.
Output. Triangulação de S
1.Ordenar os pontos de S de acordo com a coordenada x
2.Construir o primeiro triângulo com base nos três primeiros pontos
3.Para cada ponto p_i, 3<i<|S| faça
4. Conecte p_i com o conjunto de pontos já processados visíveis a partir de p_i

- **Triangulações: algoritmo incremental**



- **Triangulações: propriedades de uma triangulação**

  - Anteriormente, vimos que o número de triângulos na triangulação de um polígono *P* com *n* vértices possui *n*-2 triângulos
  - A pouco, vimos que o número de triângulos em uma triangulação de um conjunto de pontos S, obtida pelo Algoritmo de Triangulação por Subdivisão contêm sempre 2k+h-2 triângulos, onde h é o número de vértices na fronteira de conv(S) e k os vértices no interior
  - Com efeito, é possível mostrar que o resultado anterior é válido para qualquer triangulação de S
  - Para isto usaremos a fórmula de Euler

- **Triangulações: propriedades de uma triangulação**

Teorema 7.1 (Formula de Euler). Seja G um grafo planar conexo com *V* vértices, E arestas, and F faces no plano (a face externa é ilimitada). Então $V - E + F = 2$.

Prova (por indução no número de arestas).

Se $E = 0$ então G é um vértice isolado no plano e $V-E+F = 1-0+1 = 2$. Caso contrário, escolha qualquer aresta *e* de *G*. Se *e* conecta dois vértices de *G*, contraia esta aresta, reduzindo *V* e *E* de uma unidade.

Se e é incidente a apenas um vértice, (*e* é um ciclo (loop) separando duas faces), remova tal aresta, reduzindo *F* e *E* de uma unidade, Em ambos os casos o novo grafo permanece conectado e a fórmula segue por indução.

- **Triangulações: propriedades de uma triangulação**

Teorema 7.2 (Número de triângulos e arestas em uma triangulação no plano). Seja S um conjunto de pontos, com h pontos na fronteira de conv(S) e k no interior, e tal que o total de pontos seja $n = k + h$. Se nem todos os pontos forem colineares então, qualquer triangulação de S terá exatamente $2k+h-2$ triângulos e $3k+2h-3$ arestas.

- **Triangulações: propriedades de uma triangulação**

Prova. Seja *T* uma triangulação de um conjunto de pontos S e seja *t* o número de triângulos de *T*. Sabemos que *T* subdivide o plano em *t*+1 faces, t internas a conv(S) e uma(1) externa. Cada triângulo possui 3 arestas e a face externa possuir *h* arestas. Como cada aresta é adjacente a exatamente duas faces, (3*t*+*h*) contabiliza as arestas duas vezes. Logo, existem exatamente (3*t*+*h*)/2 arestas.

Aplicando a fórmula de Euler com *V*=*n* e *F*= *t*+1 temos

$$n-\frac{1}{2}(3t+h)+(t+1)=2$$

Resolvendo para *t* obtemos

$$t=2n-h-2=2k+h-2$$

- **Triangulações: o grafo de arestas trocadas (*flip graph*)**

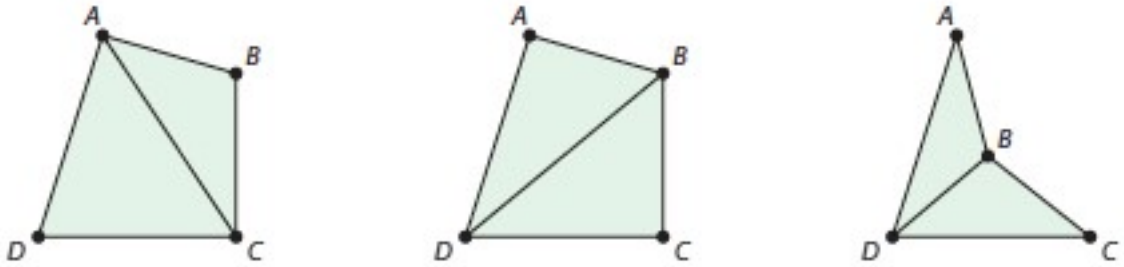
  - A estrutura de um conjunto de pontos pode ser avaliada pelo conjunto de suas triangulações

  - É possível identificar a proximidade de duas triangulações pela diferença entre suas diagonais

- **Triangulações: o grafo de arestas trocadas (*flip graph*)**

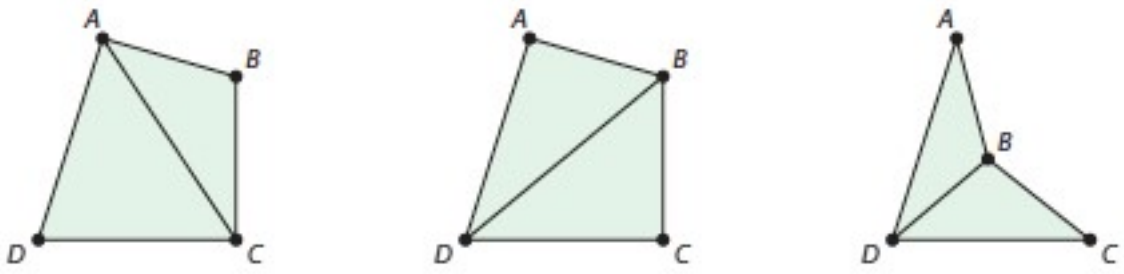
  - Considere um quadrilátero ABCD

  - Uma *edge flip* é uma operação que remove uma diagonal AC e a substitui por uma diagonal BD



- **Triangulações: o grafo de arestas trocadas (*flip graph*)**

  - Aplicando uma *edge flip* em BD voltamos a configuração original
  - Uma *edge flip* não pode ser definida sobre um quadrilátero não convexo

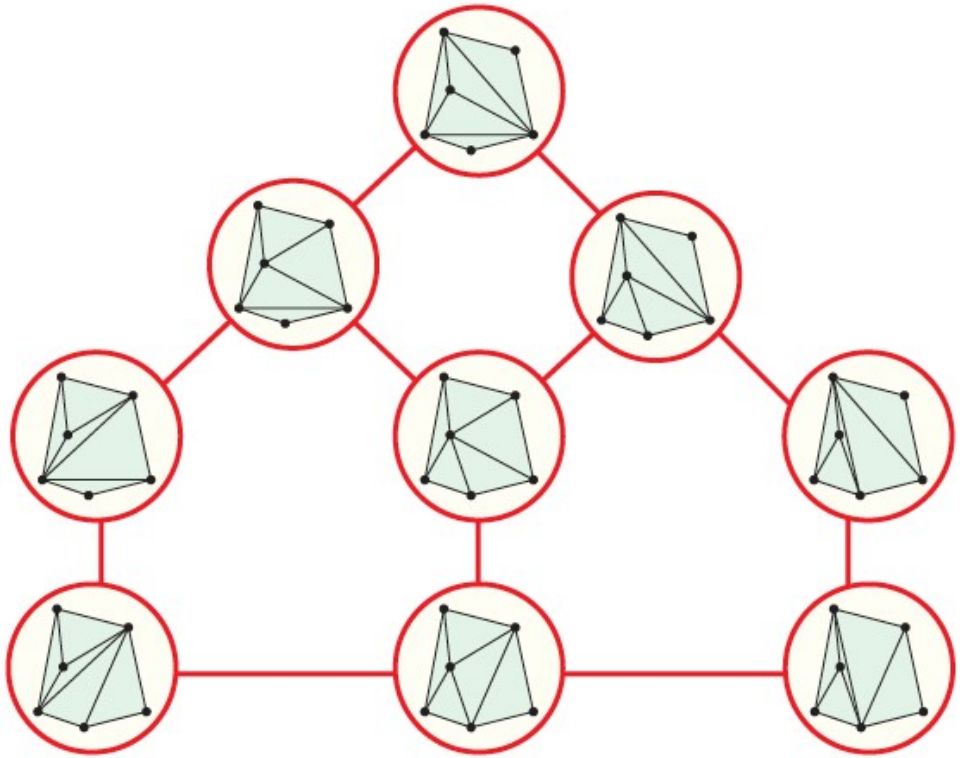


- **Triangulações: o grafo de arestas trocadas (*flip graph*)**

  - Operações de *edge flip* permitem estabelecer relações entre triangulações

  - Definição 7.1 – Para um dado conjunto de pontos S, o flip graph de S é o grafo cujos nós são o conjunto de todas as triangulações de S. Dois nós T1 e T2 no grafo são conectados por um arco se uma diagonal de T1 puder sofrer um operação *edge flip* de forma a obter T2

- **Triangulações: o grafo de arestas trocadas (*flip graph*)**



- **Triangulações: o grafo de arestas trocadas (*flip graph*)**

  - O *flip graph* revela uma espaço discreto de triangulações
  - Uma questão importante é se uma triangulação S pode ser transformada em outra efetuando-se uma sequência de *flips*

- **Triangulações: o grafo de arestas trocadas (*flip graph*)**

Teorema 7.2 (Conexidade do *flip graph*). O *flip graph* de qualquer conjunto de pontos é conectado

- **Triangulações: o grafo de arestas trocadas (*flip graph*)**

Prova. Seja S um conjunto com n pontos planares. Ordena-se os pontos de S de acordo com a coordenada x (se dois pontos tem a mesma coordenada x pode-se reorientar o conjunto)

Rotula-se o resultado da ordenação como $\{p_1,...,p_n\}$

Seja T* a triangulação gerada pelo algoritmo incremental. Provaremos que qualquer triangulação T de S pode ser obtida de T* usando *edge flips*

Isto é suficiente para provar que a conectividade do grafo uma vez que duas triangulações quaisquer estarão conectadas ao nó associado a T* no *flip graph* e consequentemente umas as outras invertendo-se os flips de um dos caminhos

- **Triangulações: o grafo de arestas trocadas (*flip graph*)**

Prova(preâmbulo). Seja S um conjunto com n pontos planares. Ordena-se os pontos de S de acordo com a coordenada x (se dois pontos tem a mesma coordenada x pode-se reorientar o conjunto)

Rotula-se o resultado da ordenação como $\{p_1,...,p_n\}$

Seja T* a triangulação gerada pelo algoritmo incremental. Provaremos que qualquer triangulação T de S pode ser obtida de T* usando *edge flips*

Isto é suficiente para provar que a conectividade do grafo uma vez que duas triangulações quaisquer estarão conectadas ao nó associado a T* no *flip graph* e consequentemente umas as outras invertendo-se os flips de um dos caminhos

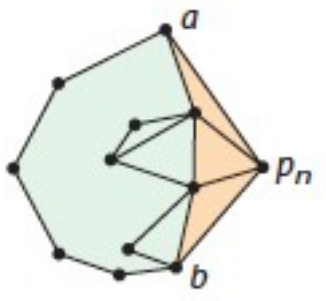
- **Triangulações: o grafo de arestas trocadas (*flip graph*)**

  - Prova(por indução).
  - Caso base(n=3) – nesse caso o flip graph tem apenas um nó.
  - Hipótese indutiva – assuma que para qualquer conjunto S com menos que n pontos, qualquer triangulação de S pode ser transformada na triangulação obtida pelo algoritmo incremental
  - Passo indutivo – Considere S com n pontos ordenados $\{p_1,...,p_n\}$ e seja T uma triangulação de S. Seja a estrela de um vértice v a união de todos os triângulos incidentes a *v*. É possível mostrar que a estrela de p_n em T pode ser convertida na estrela de pn em T*. Isto tendo sido mostrado, o que resta é a triangulação de $S\backslash\{p_n\}$ que pela hipótese indutiva pode ser transformada em T*

- **Triangulações: o grafo de arestas trocadas (*flip graph*)**

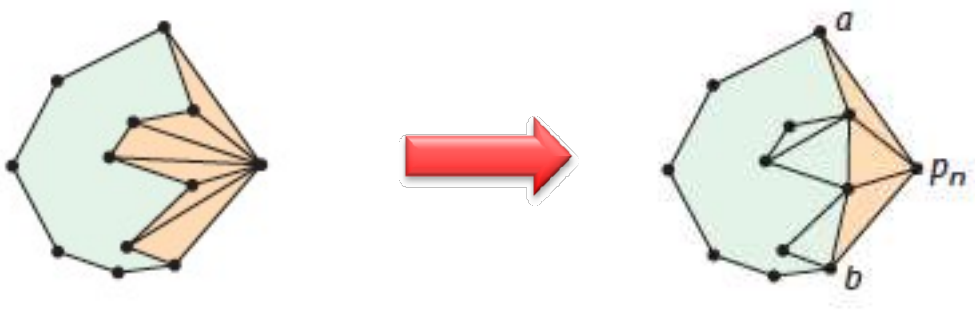
  - Prova(a estrela de p_n em T pode ser transformada na estrela de p_n em T*).

  - O algoritmo incremental produz um polígono convexo em cada passo. Logo, a estrela de p_n em T* tem exatamente 3 vértices convexos: p_n e os dois vértices adjacentes *a* e *b*, na fronteira de conv(S).



- **Triangulações: o grafo de arestas trocadas (*flip graph*)**

  - Observe que por construção a estrela de p_n em T* tem uma cadeia reflexa. Logo, temos que mostrar que a partir de T podemos construir a cadeia reflexa, obtendo deste modo a estrela de p_n em T*

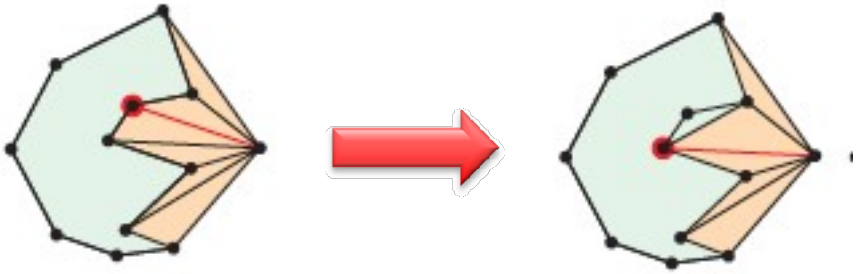


- **Triangulações: o grafo de arestas trocadas (*flip graph*)**

- Escolha um vértice v convexo na na estrela de p_n distinto de p_n , *a* e *b.*

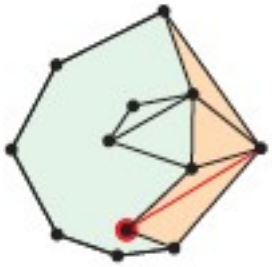
- Se não existir tal v já temos a estrela de p_n em T* e já temos o resultado desejado

- Sendo v convexo, a aresta entre v e p_n é uma diagonal de um quadrilátero convexo e pode sofrer *edge flip.*

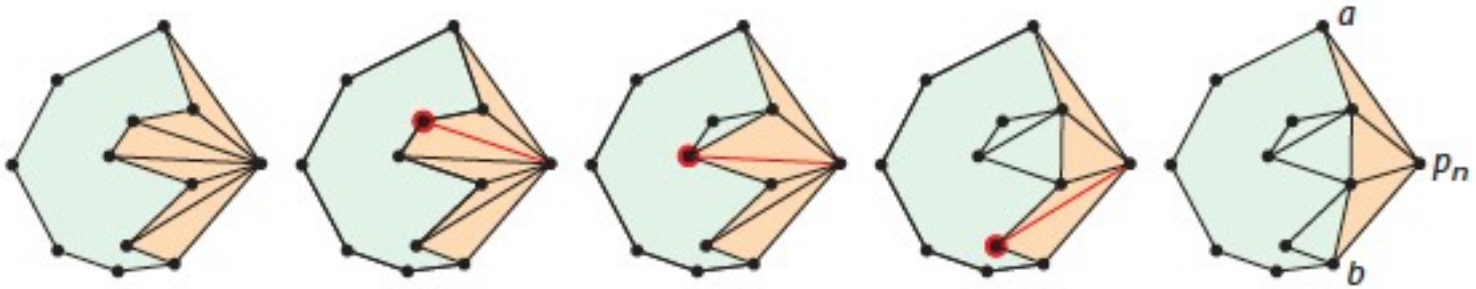


- **Triangulações: o grafo de arestas trocadas (*flip graph*)**

- Após o edge flip em v um vértice a menos é visível de p_n e o processo pode sere repetido um número finito de passos já que a cada *edge flip* o grau de p_n é reduzido.



- Como ficamos apenas com vértices não-convexo construímos a cadeia reflexa que forma junto com p_n a estrela de T*



- **Triangulações: o grafo de arestas trocadas (*flip graph*)**

  - A conectividade do *flip graph* permite transformar uma triangulação em outra a partir de movimentos locais

  - Esta é a base para muitos algoritmos que incrementalmente melhoram a qualidade da triangulação como será visto a seguir

Triangulações: Referências

- Introdução:

P. C. P. Carvalho e L. H. de Figueiredo,
Introdução à Geometria Computational, 18° Colóquio Brasileiro de Matemática,
IMPA, 1991.

- Demais slides incluindo as imagens (exceto as feitas pelo expositor):

Satyan L. Devadoss, Joseph O'Rourke,
Discrete and Computational Geometry, Princeton University Press, 2011.