Triangulação

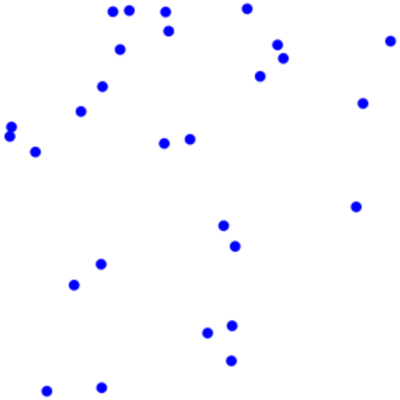
INF2604 – Geometria Computacional

Waldemar Celes
celes@inf.puc-rio.br

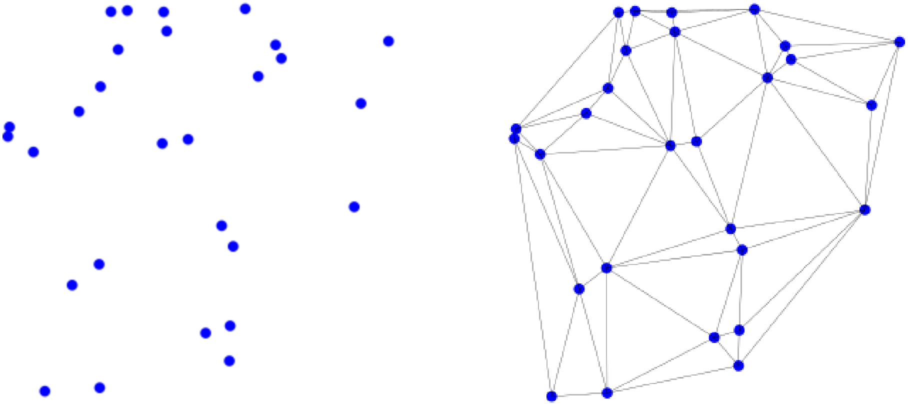
Departamento de Informática, PUC-Rio

Triangulação

Dado um conjunto fixo de pontos não estruturados S, determinar uma partição em triângulos onde os vértices são os pontos do conjunto



Dado um conjunto fixo de pontos não estruturados S, determinar uma partição em triângulos onde os vértices são os pontos do conjunto



Triangulação

Definição: **aresta**

- Qualquer segmento que conecta dois pontos do conjunto S

Definição: **triangulação planar**

- A triangulação de um conjunto planar de pontos S é uma subdivisão do plano determinada por um conjunto maximal de arestas que não se cruzam.
  - Não existe outra aresta que possa ser incluída

Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Triangulação

Definição: **aresta**

- Qualquer segmento que conecta dois pontos do conjunto S

Definição: **triangulação planar**

- A triangulação de um conjunto planar de pontos S é uma subdivisão do plano determinada por um conjunto maximal de arestas que não se cruzam.
  - Não existe outra aresta que possa ser incluída
    - **Arestas do fecho convexo pertencem à triangulação**
    - **Regiões internas ao fecho convexo são triângulos**

Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Triangulação

Definição: **aresta**

- Qualquer segmento que conecta dois pontos do conjunto S

Definição: **triangulação planar**

- A triangulação de um conjunto planar de pontos S é uma subdivisão do plano determinada por um conjunto maximal de arestas que não se cruzam.
  - Não existe outra aresta que possa ser incluída
    - **Arestas do fecho convexo pertencem à triangulação**
    - **Regiões internas ao fecho convexo são triângulos**
  - Existem diferentes triangulações para um conjunto de pontos

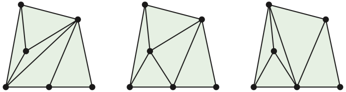


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Algoritmo subdivisão de triângulos

Entrada:

- Dado um conjunto de pontos
  - Assume que não existem pontos colineares

Algoritmo:

- Constrói o fecho convexo dos pontos
- Constrói uma triangulação do fecho convexo
  - Ignorando os pontos interiores
- Para cada ponto interior
  - Localiza o triângulo que o contém
  - Subdivide o triângulo em 3 subtriângulos

Algoritmo subdivisão de triângulos

Exemplo

- Triangulação do fecho e ordem de processamento dos pontos interiores afetam triangulação obtida

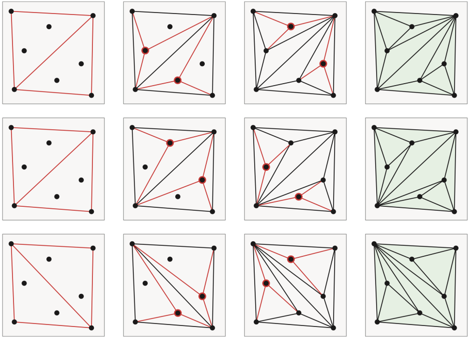


Figura extraída de *Discrete and Computational Geometry*, Devadoss and Rourke, 2011

Algoritmo subdivisão de triângulos

Tempo esperado

Algoritmo subdivisão de triângulos

Tempo esperado

- Determinação do fecho convexo
- Triangulação do fecho
- Construção da triangulação
  - Para cada ponto do interior
    - Busca do triângulo que contém determinado ponto
    - Subdivisão do triângulo

Algoritmo subdivisão de triângulos

Tempo esperado

- Determinação do fecho convexo: $O(n \log n)$
- Triangulação do fecho: $O(n)$
- Construção da triangulação: $O(n^2)$
  - Para cada ponto do interior: $O(n)$
    - Busca do triângulo que contém determinado ponto: $O(n)$
    - Subdivisão do triângulo: $O(1)$

Algoritmo subdivisão de triângulos

Tempo esperado

- Determinação do fecho convexo: $O(n \log n)$
- Triangulação do fecho: $O(n)$
- Construção da triangulação: $O(n^2)$
  - Para cada ponto do interior: $O(n)$
    - Busca do triângulo que contém determinado ponto: $O(n)$
    - Subdivisão do triângulo: $O(1)$

Tempo esperado total: $O(n^2)$

Triangulação

Propriedade

- Se temos h pontos no fecho convexo e k pontos interiores, o número total de triângulos obtidos é $2k + h - 2$

Triangulação

Propriedade

- Se temos h pontos no fecho convexo e k pontos interiores, o número total de triângulos obtidos é $2k + h - 2$

Prova pelo algoritmo:

Triangulação

Propriedade

- Se temos h pontos no fecho convexo e k pontos interiores, o número total de triângulos obtidos é $2k + h - 2$

Prova pelo algoritmo:

- Triangulação de P_n tem $n - 2$ triângulos (ver "Polígonos")
  - No caso, temos $h - 2$ triângulos iniciais
- Para cada ponto interior, adiciona-se 2 triângulos
  - Remove um e adiciona três

Triangulação

Propriedade

- Se temos h pontos no fecho convexo e k pontos interiores, o número total de triângulos obtidos é $2k + h - 2$

Prova pela fórmula de Euler: $V - E + F = 2$

Triangulação

Propriedade

- Se temos h pontos no fecho convexo e k pontos interiores, o número total de triângulos obtidos é $2k + h - 2$

Prova pela fórmula de Euler: $V - E + F = 2$

- Número de faces: $F = t + 1$ (triângulos + face externa)
- Número de arestas: $E = (3t + h)/2$
- Número de vértices: $V = h + k$
- Colocando na fórmula de Euler:

$$h + k - \frac{3t + h}{2} + t + 1 = 2 \quad \therefore \quad t = 2k + h - 2$$

Algoritmo de triangulação incremental

- Baseado no algoritmo incremental para fecho convexo
  - Ordena pontos em x
  - Cria primeiro triângulo e insere pontos no fecho em ordem

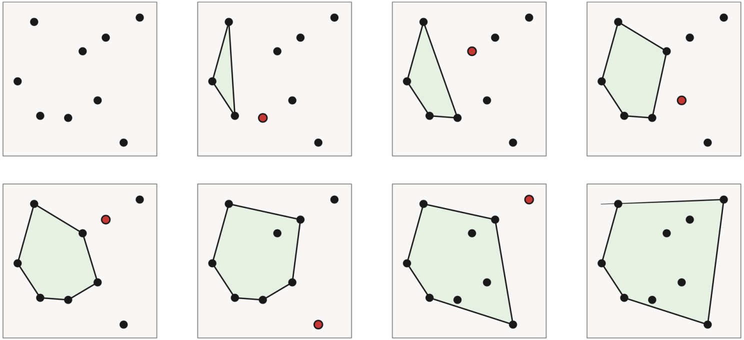


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Algoritmo de triangulação incremental

Algoritmo

- Ordena pontos em ordem crescente de coordenada x
- Forma triângulo com os três primeiros pontos
- Adiciona ponto $\mathbf{p}_{k+1}$
  - Conecta ponto a todos os anteriores visíveis

Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Algoritmo de triangulação incremental

Algoritmo

- Ordena pontos em ordem crescente de coordenada x
- Forma triângulo com os três primeiros pontos
- Adiciona ponto $\mathbf{p}_{k+1}$
  - Conecta ponto a todos os anteriores visíveis

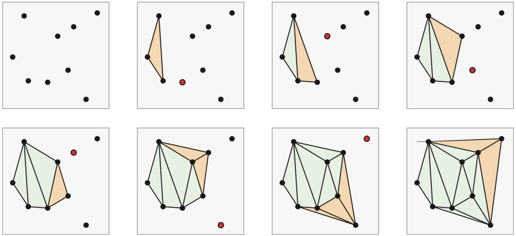


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Algoritmo de triangulação incremental

Tempo esperado

Algoritmo de triangulação incremental

Tempo esperado

- Ordenação dos pontos em x
- Triangulação do pontos visíveis
  - Para cada ponto processado
    - Determinação dos pontos visíveis
    - Construção dos triângulos

Algoritmo de triangulação incremental

Tempo esperado

- Ordenação dos pontos em x: $O(n \log n)$
- Triangulação do pontos visíveis: $O(n^2)$
  - Para cada ponto processado: $O(n)$
    - Determinação dos pontos visíveis: $O(n)$ ou O(1) amortizado
    - Construção dos triângulos: $O(n)$ ou O(1) amortizado

Algoritmo de triangulação incremental

Tempo esperado

- Ordenação dos pontos em x: $O(n \log n)$
- Triangulação do pontos visíveis: $O(n^2)$
  - Para cada ponto processado: $O(n)$
    - Determinação dos pontos visíveis: $O(n)$ ou O(1) amortizado
    - Construção dos triângulos: $O(n)$ ou O(1) amortizado

Tempo esperado total: $O(n^2)$ ou $O(n \log n)$ amortizado

Inversão de arestas

Operador inversão de arestas (*edge flip*)

- Em uma triangulação de um quadrilátero **abcd** convexo
  - Remove a diagonal **ac** e inclui a diagonal **bd**.

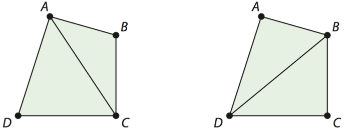
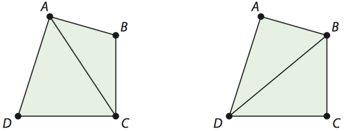


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Inversão de arestas

Operador inversão de arestas (*edge flip*)

- Em uma triangulação de um quadrilátero **abcd** convexo
  - Remove a diagonal **ac** e inclui a diagonal **bd**.



- Inversão de aresta não pode ocorrer em quadriláteros côncavos

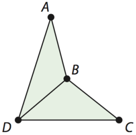


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Grafo de inversão (*flip graph*)

Grafo

- Nós são triangulações
- Arestas são operações de inversão de arestas

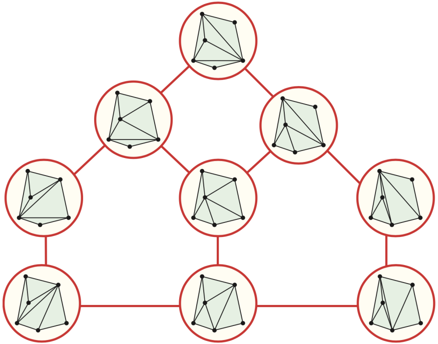


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Grafo de inversão (*flip graph*)

Teorema:

- O grafo de inversão de qualquer conjunto de pontos é **conexo**
  - Transformamos uma triangulação em qualquer outra por um conjunto finito de operações de inversão de arestas

Grafo de inversão (*flip graph*)

Teorema:

- O grafo de inversão de qualquer conjunto de pontos é **conexo**
  - Transformamos uma triangulação em qualquer outra por um conjunto finito de operações de inversão de arestas

Conectividade do grafo de inversão pode ser base de muitos algoritmos

- Dada uma triangulação inicial, de forma incremental, faça operações de inversão de arestas para "melhorar" a qualidade da triangulação

Grafo de inversão (*flip graph*)

Prova por indução:

Grafo de inversão (*flip graph*)

Prova por indução:

- Para $n = 3$, a triangulação é única (o grafo só tem um nó)

Grafo de inversão (*flip graph*)

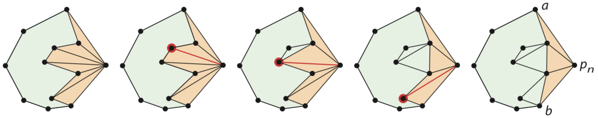
Prova por indução:

- Para $n = 3$, a triangulação é única (o grafo só tem um nó)
- Para $n > 3$:
  Qualquer T pode ser transformado em $T*$ (triangulação incremental)

Grafo de inversão (*flip graph*)

Prova por indução:

- Para $n = 3$, a triangulação é única (o grafo só tem um nó)
- Para $n > 3$:
  Qualquer T pode ser transformado em $T*$ (triangulação incremental)
  - Considere o ponto mais à direita de T: p_n
  - Considere a estrela de triângulos de p_n
  - Considere o polígono restante
  - Transforme esse polígono em convexo:
    - Enquanto existir vértice côncavo, inverta aresta
    - No final, estrela de triângulos igual a $T*$
  - Repita procedimento com $n-1$ pontos, até $n = 3$



Grafo de inversão (*flip graph*)

Custo máximo para melhorar uma triangulação

- **Diâmetro** do grafo de inversão
  - Diâmetro de um grafo é o comprimento do maior caminho entre dois nós quaisquer

Grafo de inversão (*flip graph*)

Custo máximo para melhorar uma triangulação

- **Diâmetro** do grafo de inversão
  - Diâmetro de um grafo é o comprimento do maior caminho entre dois nós quaisquer

Corolário

- O diâmetro do grafo de inversão é no máximo $(n-2)(n-3)$

Grafo de inversão (*flip graph*)

Prova:

- Com no máximo $\binom{n-2}{2}$ inversões, transformamos T em $T*$
  - Com no máximo $n-3$, convertemos a estrela de p_n de T em $T*$
  - Por indução: convertemos T resultante em $T*$ com $\binom{n-3}{2}$
    - De fato: $\binom{n-3}{2} + n - 3 = \binom{n-2}{2}$
- Para transformar T_1 em T_2 quaisquer
- Precisamos de $2\binom{n-2}{2} = (n-2)(n-3)$

Grafo de inversão (*flip graph*)

Teorema:

- Dadas duas triangulações T_1 e T_2 de S
  - A distância máxima entre T_1 e T_2 no grafo de inversão é igual ao número de interseções da superposição de T_1 e T_2

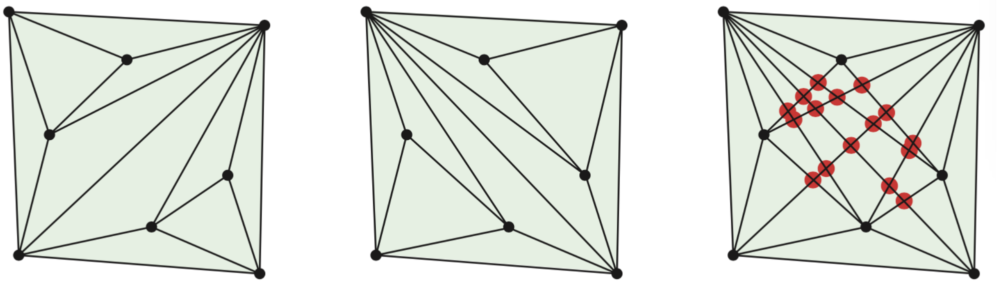


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Triangulação 3D

Triangulação 3D

Algoritmo incremental

- Ordene os pontos em x
- Os 4 primeiros formam um primeiro tetrahedro
- Considere cada ponto seguinte
  - Insira tetraedros ligando o novo ponto aos triângulos do fecho visíveis

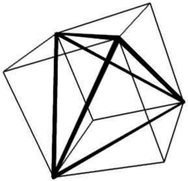
Triangulação 3D

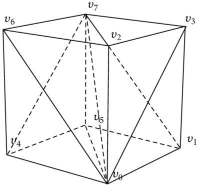
Algoritmo incremental

- Ordene os pontos em x
- Os 4 primeiros formam um primeiro tetrahedro
- Considere cada ponto seguinte
  - Insira tetraedros ligando o novo ponto aos triângulos do fecho visíveis

Observações:

- O número de tetrahedros pode variar entre triangulações





Triangulação 3D

- Operador de **inversão de face** (*face flip*)

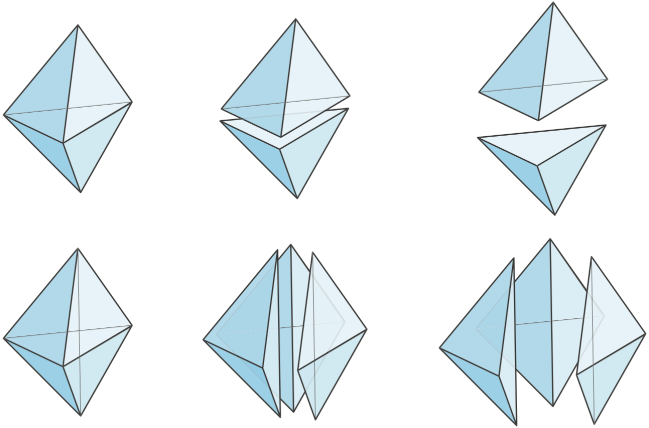


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Triangulação 3D

Grafo de inversão

- Operação de **inversão de face** (*face flip*)
- Não se sabe se o grafo é conexo

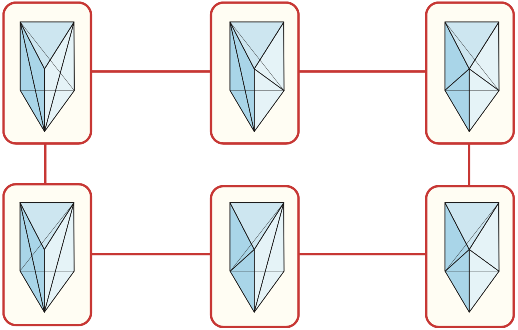


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Qualidade da triangulação

Das diversas triangulações possíveis, podemos avaliar a qualidade

- Exemplo: triangulação de terrenos (dados de elevação)
  - Inversão de aresta pode alterar formação da superfície

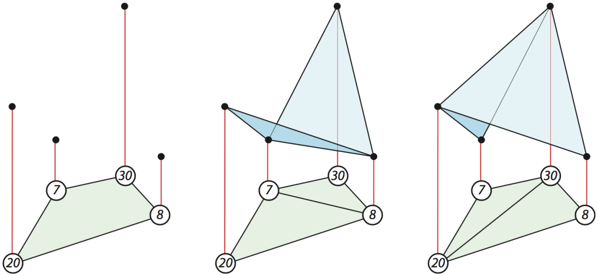
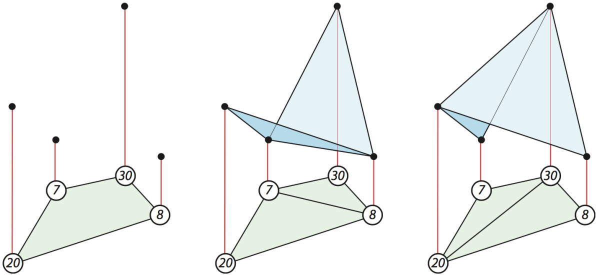


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Qualidade da triangulação

Das diversas triangulações possíveis, podemos avaliar a qualidade

- Exemplo: triangulação de terrenos (dados de elevação)
  - Inversão de aresta pode alterar formação da superfície



- Como avaliar qual é melhor se não temos mais dados?

Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Qualidade da triangulação

Triangulação de dados de elevação

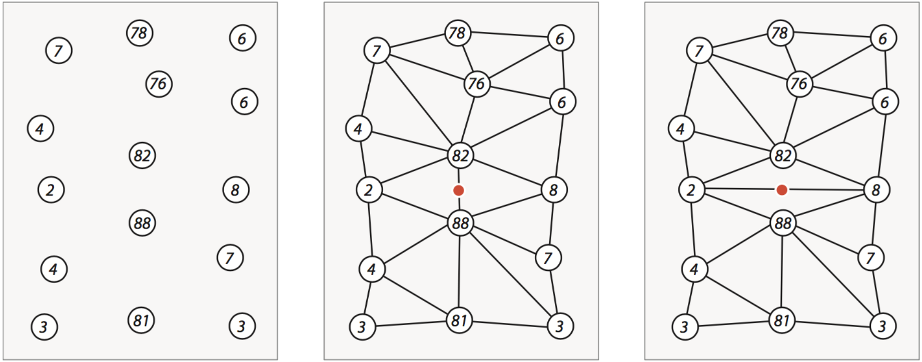
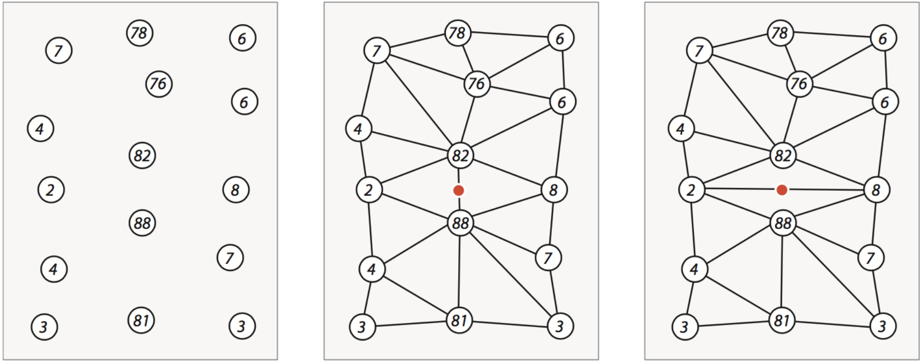


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Qualidade da triangulação

Triangulação de dados de elevação



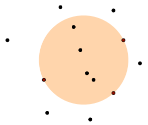
- Objetivo: evitar triângulos alongados

Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Triangulação de Delaunay

Suposição (ausência de casos degenerados)

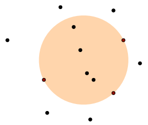
- Não existem 4 pontos sobre uma mesma circunferência



Triangulação de Delaunay

Suposição (ausência de casos degenerados)

- Não existem 4 pontos sobre uma mesma circunferência



Qualidade da triangulação

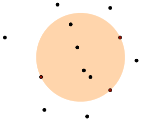
- Métrica de triangulação "gorda"
  - Sequência ordenada de ângulos dos triângulos de T

$$(\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_{3n})$$

Triangulação de Delaunay

Suposição (ausência de casos degenerados)

- Não existem 4 pontos sobre uma mesma circunferência



Qualidade da triangulação

- Métrica de triangulação "gorda"
  - Sequência ordenada de ângulos dos triângulos de T

$$(\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_{3n})$$

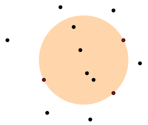
  - Exemplo: considere duas triangulações T_1 e T_2

$$T_1 : (20^\circ, 30^\circ, 45^\circ, 65^\circ, 70^\circ, 130^\circ)$$
$$T_2 : (20^\circ, 30^\circ, 45^\circ, 60^\circ, 75^\circ, 130^\circ)$$

Triangulação de Delaunay

Suposição (ausência de casos degenerados)

- Não existem 4 pontos sobre uma mesma circunferência



Qualidade da triangulação

- Métrica de triangulação "gorda"
  - Sequência ordenada de ângulos dos triângulos de T

$$(\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_{3n})$$

  - Exemplo: considere duas triangulações T_1 e T_2

$$T_1 : (20^\circ, 30^\circ, 45^\circ, 65^\circ, 70^\circ, 130^\circ)$$
$$T_2 : (20^\circ, 30^\circ, 45^\circ, 60^\circ, 75^\circ, 130^\circ)$$

    - Temos $T_1 > T_2$, pois $65^\circ > 60^\circ$

Triangulação de Delaunay

Definição: Aresta legal da triangulação

- Dada uma triangulação T_1
  - Dada uma aresta e de T_1
  - Dado o quadrilátero Q cujo e é diagonal
  - Se Q é convexo, T_2 é a triangulação com a inversão de e
  - Dizemos que e é uma **aresta legal** se $T_1 \geq T_2$

Triangulação de Delaunay

Definição: Aresta legal da triangulação

- Dada uma triangulação T_1
  - Dada uma aresta e de T_1
  - Dado o quadrilátero Q cujo e é diagonal
  - Se Q é convexo, T_2 é a triangulação com a inversão de e
  - Dizemos que e é uma **aresta legal** se $T_1 \geq T_2$

Definição: Triangulação de Delaunay

- A **triangulação de Delaunay** de um conjunto de pontos S, denotada por $Del(S)$, é uma triangulação de S que só tem *arestas legais*.

Triangulação de Delaunay

Definição: Aresta legal da triangulação

- Dada uma triangulação T_1
  - Dada uma aresta e de T_1
  - Dado o quadrilátero Q cujo e é diagonal
  - Se Q é convexo, T_2 é a triangulação com a inversão de e
  - Dizemos que e é uma **aresta legal** se $T_1 \geq T_2$

Definição: Triangulação de Delaunay

- A **triangulação de Delaunay** de um conjunto de pontos S, denotada por $Del(S)$, é uma triangulação de S que só tem *arestas legais*.
  - Pode-se provar que é a triangulação mais gorda possível

Triangulação de Delaunay

Algoritmo de inversão de arestas

- Constrói uma triangulação qualquer de S
- Verifica as arestas de T
  - Se e é ilegal, inverte a aresta
  - Até que não existam mais arestas ilegais

Triangulação de Delaunay

Algoritmo de inversão de arestas

- Constrói uma triangulação qualquer de S
- Verifica as arestas de T
  - Se e é ilegal, inverte a aresta
  - Até que não existam mais arestas ilegais

Esse algoritmo termina?

Triangulação de Delaunay

Algoritmo de inversão de arestas

- Constrói uma triangulação qualquer de S
- Verifica as arestas de T
  - Se e é ilegal, inverte a aresta
  - Até que não existam mais arestas ilegais

Esse algoritmo termina?

- A sequência de ângulos das triangulações visitadas só aumenta
  - Uma mesma triangulação nunca é revisitada
- O número de triangulações é finito
  - O algoritmo tem que terminar

Triangulação de Delaunay

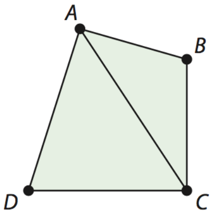
Como determinar se uma aresta é legal?

Figura extraída de *Discrete and Computational Geometry*, Devadoss and Rourke, 2011

Triangulação de Delaunay

Como determinar se uma aresta é legal?

- Procedimento exaustivo
  - Ordena os 6 ângulos de T_1
  - Ordena os 6 ângulos de T_2
  - Verifica se $T_1 \geq T_2$



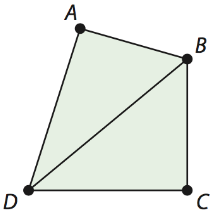


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Triangulação de Delaunay

Teorema de Thales

- Considere três pontos P, Q, e B cocirculares
- Considere A dentro e C fora do círculo

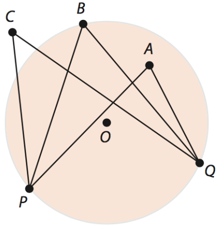


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Triangulação de Delaunay

Teorema de Thales

- Considere três pontos P, Q, e B cocirculares
- Considere A dentro e C fora do círculo
- Tem-se:

$$\widehat{PAQ} > \widehat{PBQ} > \widehat{PCQ}$$

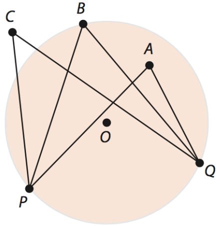


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Triangulação de Delaunay

Proposição

- Considere uma aresta $e = AC$ de T
- Considere os dois triângulos adjacentes ABC e ACD
- A aresta e é legal se D está fora do círculo definido por ABC

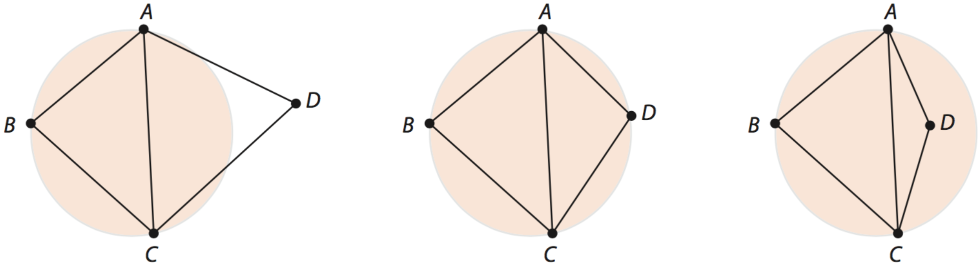
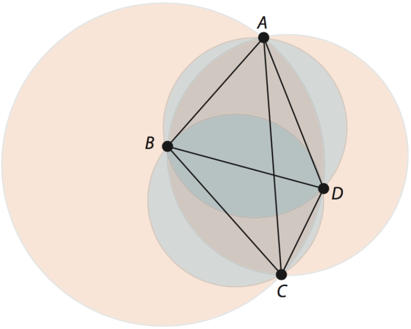


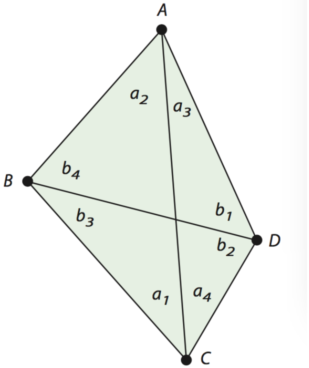
Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Triangulação de Delaunay

Propriedade do círculo vazio

- Em uma triangulação de Delaunay, nenhum ponto é interior ao círculo definido por qualquer triângulo





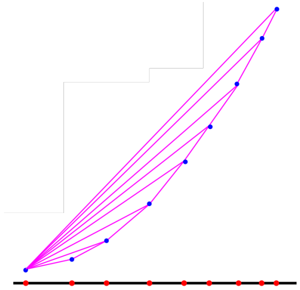
Triangulação de Delaunay

Limite inferior de complexidade

Triangulação de Delaunay

Limite inferior de complexidade

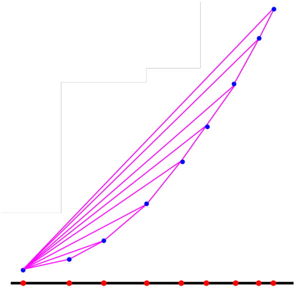
- Considere o conjunto de pontos (x_i, x_i^2)
- Compute a triangulação de Delaunay dos pontos: $f(n)$
- Da triangulação, ordena x_i: $O(n)$
  - Basta visitar os triângulos vizinhos



Triangulação de Delaunay

Limite inferior de complexidade

- Considere o conjunto de pontos (x_i, x_i^2)
- Compute a triangulação de Delaunay dos pontos: $f(n)$
- Da triangulação, ordena x_i: $O(n)$
  - Basta visitar os triângulos vizinhos



Logo:

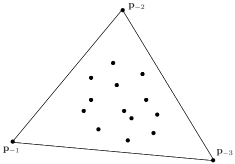
$$f(n) + O(n) \in \Omega(n \log n)$$

Triangulação de Delaunay

Algoritmo incremental

M. de Berg et al., "Computational Geometry", 2nd edition, Springer, 2000

Triangulação de Delaunay



Algoritmo incremental

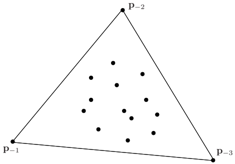
- Cria um triângulo envolvendo S: $E = \{\mathbf{p}_{-3}, \mathbf{p}_{-2}, \mathbf{p}_{-1}\}$
  - Vamos construir Delaunay de $S \cup E$
  - O triângulo envolvente é a triangulação inicial

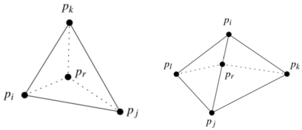
M. de Berg et al., "Computational Geometry", 2nd edition, Springer, 2000

Triangulação de Delaunay

Algoritmo incremental

- Cria um triângulo envolvendo S: $E = \{\mathbf{p}_{-3}, \mathbf{p}_{-2}, \mathbf{p}_{-1}\}$
  - Vamos construir Delaunay de $S \cup E$
  - O triângulo envolvente é a triangulação inicial
- Em **ordem aleatória**, considere cada ponto $\mathbf{p}_i$
  - Localize o ponto na triangulação
    - Ponto no inteiror de um triângulo:
      subdivida o triângulo em 3 triângulos
    - Ponto sobre aresta:
      subdivida triângulos adjacentes em 2 triângulos cada



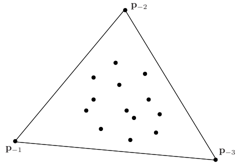


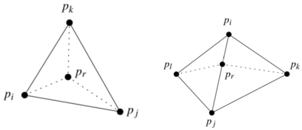
M. de Berg et al., "Computational Geometry", 2nd edition, Springer, 2000

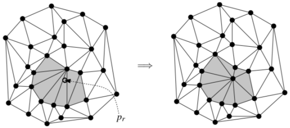
Triangulação de Delaunay

Algoritmo incremental

- Cria um triângulo envolvendo S: $E = \{\mathbf{p}_{-3}, \mathbf{p}_{-2}, \mathbf{p}_{-1}\}$
  - Vamos construir Delaunay de $S \cup E$
  - O triângulo envolvente é a triangulação inicial
- Em **ordem aleatória**, considere cada ponto $\mathbf{p}_i$
  - Localize o ponto na triangulação
    - Ponto no inteiror de um triângulo:
      subdivida o triângulo em 3 triângulos
    - Ponto sobre aresta:
      subdivida triângulos adjacentes em 2 triângulos cada
  - Elimine eventuais arestas ilegais introduzidas
    - Apenas arestas de triângulos alterados são testadas





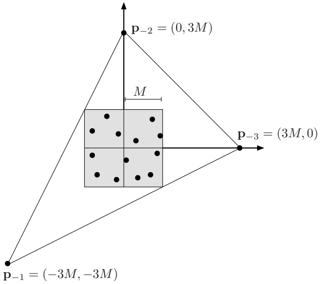


M. de Berg et al., "Computational Geometry", 2nd edition, Springer, 2000

Triangulação de Delaunay

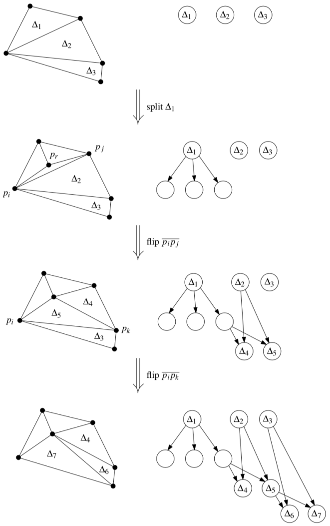
Algoritmo incremental

- Criação do triângulo envolvente
  - Triângulo envolvendo quadrado centrado
  - Pontos $\{\mathbf{p}_{-3}, \mathbf{p}_{-2}, \mathbf{p}_{-1}\}$
  - Teste de arestas ilegais assume pontos fora de qualquer círculo



Triangulação de Delaunay: Algoritmo incremental

- Estrutura de dados para localização de ponto
  - Árvore de triângulos $\mathbb{D}$
    - Subdivisão de triângulos registrados em $\mathbb{D}$



M. de Berg et al., "Computational Geometry", 2nd edition, Springer, 2000

Triangulação de Delaunay: Algoritmo incremental

Tempo esperado

Triangulação de Delaunay: Algoritmo incremental

Tempo esperado

- Criação de triângulo envolvente
- Inserção dos pontos em ordem aleatória
  - Localização do ponto na triangulação
  - Subdivisão de triângulos e atualização de $\mathbb{D}$
  - Eliminação de arestas ilegais

Triangulação de Delaunay: Algoritmo incremental

Tempo esperado

- Criação de triângulo envolvente: $O(n)$
- Inserção dos pontos em ordem aleatória: $O(n \log n)$
  - Localização do ponto na triangulação: $O(\log n)$
  - Subdivisão de triângulos e atualização de $\mathbb{D}$: $O(1)$
  - Eliminação de arestas ilegais: $O(1)$

Triangulação de Delaunay: Algoritmo incremental

Tempo esperado

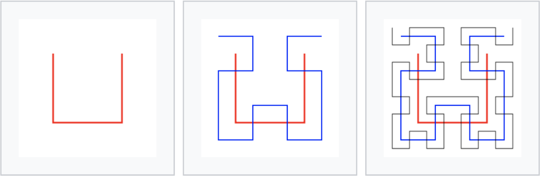
- Criação de triângulo envolvente: $O(n)$
- Inserção dos pontos em ordem aleatória: $O(n \log n)$
  - Localização do ponto na triangulação: $O(\log n)$
  - Subdivisão de triângulos e atualização de $\mathbb{D}$: $O(1)$
  - Eliminação de arestas ilegais: $O(1)$

Tempo esperado total: $O(n \log n)$

Triangulação de Delaunay: Algoritmo incremental

Alternativa de implementação

- Cria triângulo envolvente
- Processa pontos em **ordem espacial**
  - Curva de preenchimento espacial
    - Exemplo: curva de Hilbert

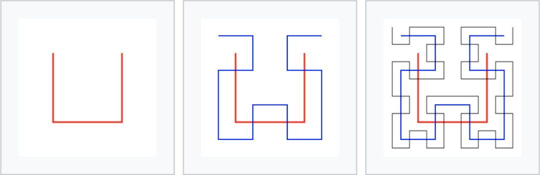


- Busca topológica do triângulo que contém ponto

Triangulação de Delaunay: Algoritmo incremental

Alternativa de implementação

- Cria triângulo envolvente
- Processa pontos em **ordem espacial**
  - Curva de preenchimento espacial
    - Exemplo: curva de Hilbert



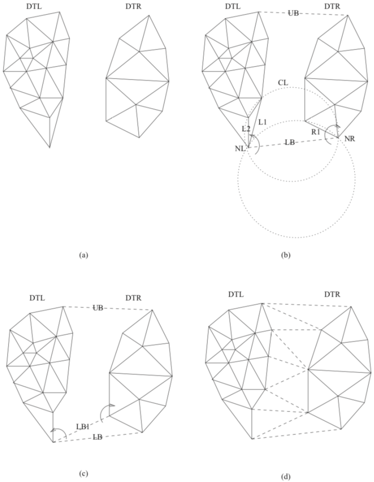
- Busca topológica do triângulo que contém ponto

Tempo esperado: $O(n \log n)$

Triangulação de Delaunay: Algoritmo dividir & conquistar

Algoritmo

- Ordena pontos em x
- Recursivamente
  - Divide pontos em duas partições
  - Triangula cada partição
  - Combina as duas triangulações



Triangulação de Delaunay: Algoritmo dividir & conquistar

Algoritmo

- Ordena pontos em x
- Recursivamente
  - Divide pontos em duas partições
  - Triangula cada partição
  - Combina as duas triangulações

- Tempo esperado: $O(n \log n)$
  - Na prática, mais eficiente que incremental
  - Não se estende para 3D