Fecho Convexo

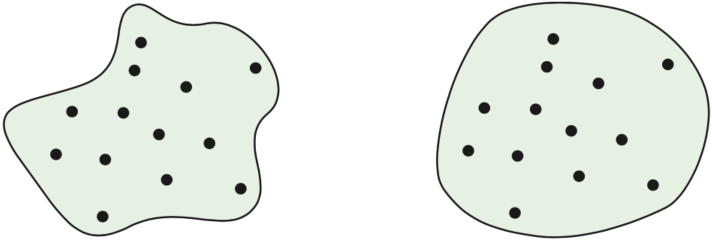
INF2604 – Geometria Computacional

Waldemar Celes
celes@inf.puc-rio.br

Departamento de Informática, PUC-Rio

Região convexa

Uma região é dita convexa se todos os pares de pontos da região são visíveis entre si.



- Dada uma região convexa R, qualquer linha $\overline{\mathbf{pq}}$ entre dois pontos de R está contida na região: $\overline{\mathbf{pq}} \subset R$.

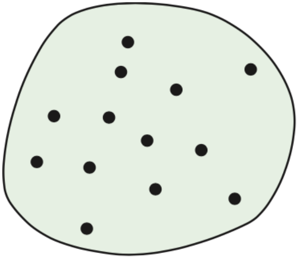
$$\alpha \mathbf{p} + \beta \mathbf{q} \in R, \quad \alpha + \beta = 1$$

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

Fecho convexo

Visão intutitiva

- Considere um conjunto de pregos numa superfície plana; o fecho convexo 2D é definido por um elástico que envolve o conjunto de pregos (pontos).



Fecho convexo

Visão intutitiva

- Considere um conjunto de pregos numa superfície plana; o fecho convexo 2D é definido por um elástico que envolve o conjunto de pregos (pontos).

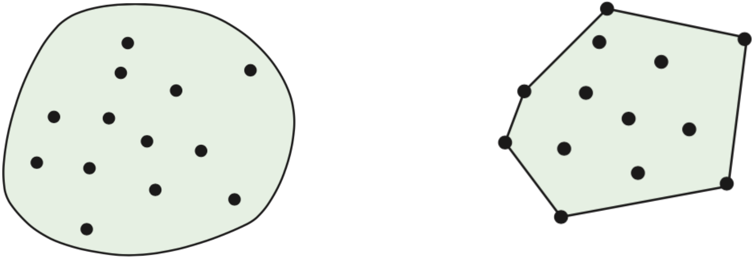


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Fecho convexo

Dado um conjunto de pontos $S = \mathbf{p}_1, ...\mathbf{p}_n$, o fecho convexo de S é o conjunto de todas as combinações convexas de S:

- Uma combinação convexa de S é expressa por:

$$\lambda_1 \mathbf{p}_1 + \cdots + \lambda_n \mathbf{p}_n$$

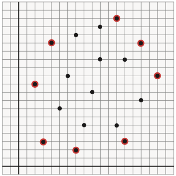
- Fecho convexo:

$$conv(S) = \left\{ \lambda_1 \mathbf{p}_1 + \cdots + \lambda_n \mathbf{p}_n | \lambda_i \geq 0, \quad \sum \lambda_i = 1 \right\}$$

Fecho convexo

Determinação do fecho convexo de um conjunto de pontos

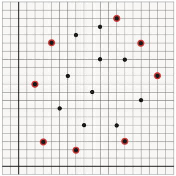
- Identificação dos vértices pertencenes ao fecho



Fecho convexo

Determinação do fecho convexo de um conjunto de pontos

- Identificação dos vértices pertencenes ao fecho



Algoritmos para construção de fecho convexo

- Incremental
- *Gift wrapping*
- Varredura de Graham
- Dividir e conquistar

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

Algoritmo incremental

Ideia: assume que temos o fecho de k pontos

- Incrementalmente, calculamos o fecho de $k+1$ pontos

Algoritmo incremental

Ideia: assume que temos o fecho de k pontos

- Incrementalmente, calculamos o fecho de $k + 1$ pontos

Suposições:

- Não existem pontos com coordenadas x coincidentes
- Não existem pontos colineares

Algoritmo incremental

Ideia: assume que temos o fecho de k pontos

- Incrementalmente, calculamos o fecho de $k+1$ pontos

Suposições:

- Não existem pontos com coordenadas x coincidentes
- Não existem pontos colineares

Algoritmo

- Ordena pontos em ordem crescente de coordenada x
- Forma $conv(H_3)$ com os três primeiros pontos
  - Orientação antihorária
- Adiciona ponto $\mathbf{p}_{k+1}$
  - Como $\mathbf{p}_{k+1}$ é o ponto mais à direita do conjunto corrente, ele pertence a $conv(H_{k+1})$
  - Acrescenta $\mathbf{p}_{k+1}$ no fecho na ordem correta
  - Elimina pontos que não mais pertencem ao fecho

Algoritmo incremental

Definição

- Seja P um polígono convexo e $\mathbf{x}$ um ponto em ∂P
  - Uma linha L contento $\mathbf{x}$ *suporta* P em $\mathbf{x}$
    se P está todo em um dos lados de L
  - L é dita *tangente* de P em $\mathbf{x}$

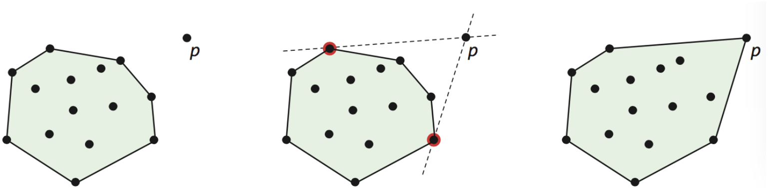
Algoritmo incremental

Definição

- Seja P um polígono convexo e $\mathbf{x}$ um ponto em ∂P
  - Uma linha L contento $\mathbf{x}$ *suporta* P em $\mathbf{x}$ se P está todo em um dos lados de L
  - L é dita *tangente* de P em $\mathbf{x}$

Determinação dos pontos eliminados do fecho devido a $\mathbf{p}_{k+1}$

- Determinar as duas tangentes a P que passa por $\mathbf{p}_{k+1}$



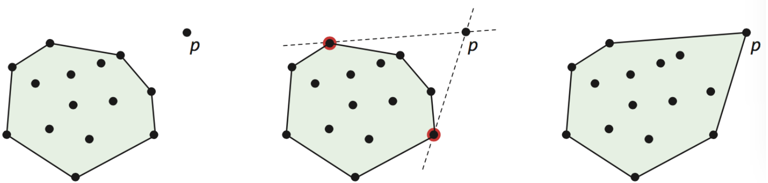
Algoritmo incremental

Definição

- Seja P um polígono convexo e $\mathbf{x}$ um ponto em ∂P
  - Uma linha L contento $\mathbf{x}$ *suporta* P em $\mathbf{x}$ se P está todo em um dos lados de L
  - L é dita *tangente* de P em $\mathbf{x}$

Determinação dos pontos eliminados do fecho devido a $\mathbf{p}_{k+1}$

- Determinar as duas tangentes a P que passa por $\mathbf{p}_{k+1}$
- Computar a visibilidade das arestas de P em relação a $\mathbf{p}_{k+1}$
  - Vértices de silhueta pertencem às tangentes
  - Vértices “visíveis” são retirados do fecho



Algoritmo incremental

Algoritmo em funcionamento

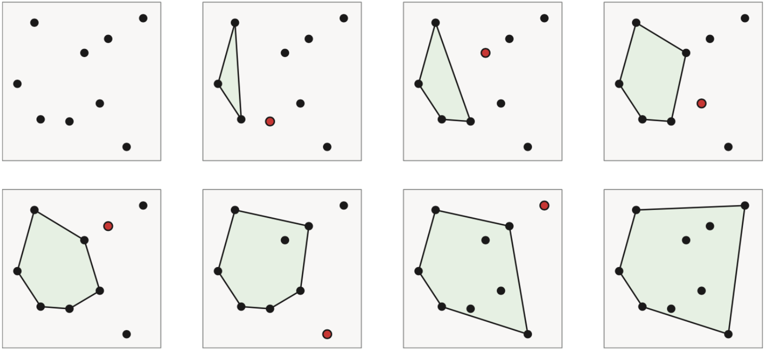


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Algoritmo incremental

Análise de complexidade

- Ordenação dos pontos:
- Teste de visibilidade:

Algoritmo incremental

Análise de complexidade

- Ordenação dos pontos:
  - $O(n \quad \log n)$
- Teste de visibilidade:

Algoritmo incremental

Análise de complexidade

- Ordenação dos pontos:
  - $O(n \quad \log n)$
- Teste de visibilidade:
  - $O(n^2)$
    - $O(k)$ para cada inserção $\mathbf{p}_{k+1}$

Algoritmo incremental

Análise de complexidade

- Ordenação dos pontos:
  - $O(n \log n)$
- Teste de visibilidade:
  - $O(n^2)$
    - $O(k)$ para cada inserção $\mathbf{p}_{k+1}$

- **Complexidade do algoritmo:** $O(n^2)$

Algoritmo incremental

Casos degenerados

- Pontos com mesma coordenada x
  - Podemos aplicar uma rotação arbitrária

Algoritmo incremental

Casos degenerados

- Pontos com mesma coordenada x
  - Podemos aplicar uma rotação arbitrária
- Pontos colineares
  - Ambiguidade de que pontos pertencem ao fecho

Algoritmo incremental

Casos degenerados

- Pontos com mesma coordenada x
  - Podemos aplicar uma rotação arbitrária
- Pontos colineares
  - Ambiguidade de que pontos pertencem ao fecho

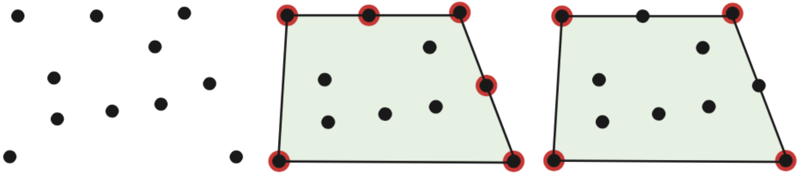


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Algoritmo *gift wrapping*

- Selecione o ponto mais abaixo (e mais à direita)
- Trace segmentos de reta aos demais pontos
  - Escolha o ponto com menor ângulo (antihorário) com eixo x
  - Tem-se a primeira aresta do fecho
- Para cada novo ponto adicionado no fecho
  - Trace segmentos para demais pontos
  - Escolha o ponto com menor ângulo da última aresta do fecho

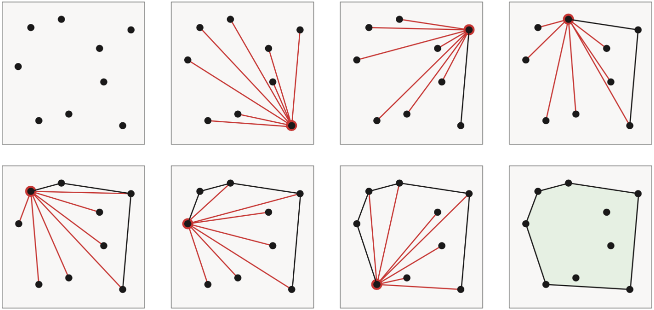


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Algoritmo *gift wrapping*

Análise de complexidade

- Verificação de todos os segmentos aos demais pontos

- Número de vezes que a verificação é feita

Algoritmo *gift wrapping*

Análise de complexidade

- Verificação de todos os segmentos aos demais pontos
  - $O(n)$
- Número de vezes que a verificação é feita
  - Número de pontos no fecho: h

Algoritmo *gift wrapping*

Análise de complexidade

- Verificação de todos os segmentos aos demais pontos
  - $O(n)$
- Número de vezes que a verificação é feita
  - Número de pontos no fecho: h

- **Complexidade do algoritmo:** $O(nh)$

Algoritmo por varredura de Graham

Ron Graham, Bell Laboratories

- Necessidade de achar fecho de 10 000 pontos
- Publicação em artigo de 1972
  - Provavelmente, primeiro artigo em Geometria Computacional

Algoritmo por varredura de Graham

- Selecione o ponto mais abaixo (e mais à direita)
- Trace segmentos de reta aos demais pontos
  - Ordene pontos em ordem crescente de ângulo com eixo x
- Processa cada ponto c em ordem, com última aresta $\overline{ab}$
  - Se dobra à esquerda: acrescenta no fecho
  - Se dobra à direita: b deve ser descartado do fecho
  - Retrocede até achar uma dobra à esquerda

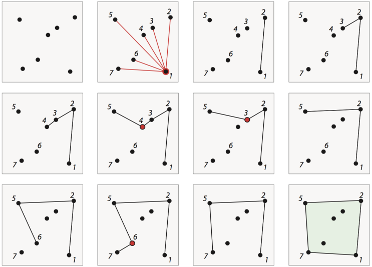


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Algoritmo por varredura de Graham

Análise de complexidade

Algoritmo por varredura de Graham

Análise de complexidade

- Ordenação dos pontos: $O(n \log n)$
- Processamento de cada ponto: $O(n)$
  - Cada ponto é processado no máximo 2 vezes
    - Adição ao fecho
    - Remoção do fecho

Algoritmo por varredura de Graham

Análise de complexidade

- Ordenação dos pontos: $O(n \quad \log n)$
- Processamento de cada ponto: $O(n)$
  - Cada ponto é processado no máximo 2 vezes
    - Adição ao fecho
    - Remoção do fecho

- **Complexidade do algoritmo:** $O(n \quad \log n)$

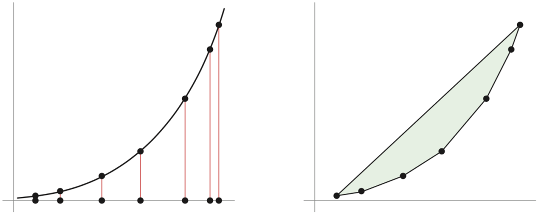
Algoritmo por varredura de Graham

É possível melhorar a complexidade do algoritmo?

Algoritmo por varredura de Graham

É possível melhorar a complexidade do algoritmo?

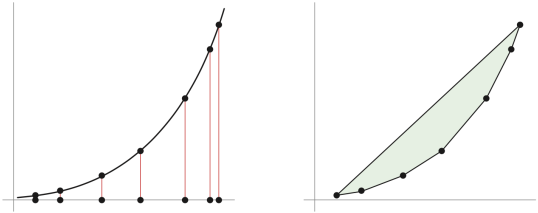
- Determinação de fecho engloba ordenação
- $O(n \quad \log n)$ é limite inferior



Algoritmo por varredura de Graham

É possível melhorar a complexidade do algoritmo?

- Determinação de fecho engloba ordenação
- $O(n \quad \log n)$ é limite inferior



Algoritmo pode ser estendido para 3D?

- Como ordenar os pontos?

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

Algoritmo dividir e conquistar

- Ordena os pontos segundo coordenada x
- Recursivamente
  - Divide em dois conjuntos A e B
    - $dim[A] = \lceil n/2 \rceil$
    - $dim[B] = \lfloor n/2 \rfloor$
  - Constroi fechos de A e B
  - Combina $conv(A)$ e $conv(B)$

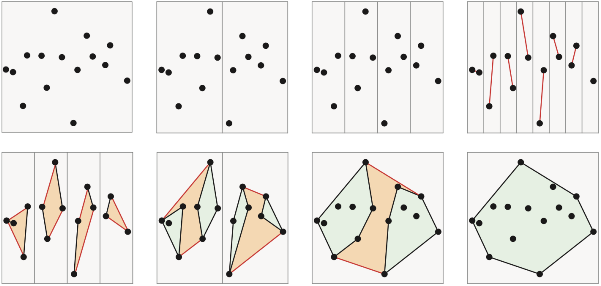


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Algoritmo dividir e conquistar

Subdivisão até $dim[S] \leq 3$

- $conv(S)$ trivial

Desafio reside na combinação

- Achar duas tangentes entre os polígonos $conv(A)$ e $conv(B)$
  - Suportam os fechos por cima e por baixo

Algoritmo dividir e conquistar

Subdivisão até $dim[S] \leq 3$

- $conv(S)$ trivial

Desafio reside na combinação

- Achar duas tangentes entre os polígonos $conv(A)$ e $conv(B)$
  - Suportam os fechos por cima e por baixo

Complexidade do algoritmo: $O(n \quad \log n)$

- Desde que a combinação seja em tempo $O(n)$

Algoritmo dividir e conquistar

Determinação das tangentes em ordem linear

- A representa o fecho à esquerda e B à direita
  - **a** e **b** são pontos mais à direita e à esquerda de A e B
- A partir de **a**, ache a tangente que suporta B
  - A partir de **b**, percorre no sentido antihorário, atualizando **b**
- A partir de **b**, ache a tangente que suporta A
  - A partir de **a**, percorre no sentido horário, atualizando **a**
- Repita até que a tangente dos dois fechos seja encontrada
- Execute procedimento similar para achar tangente superior

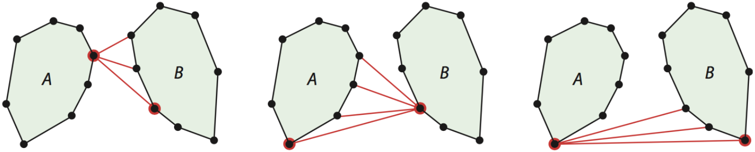


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Fecho convexo em 3D

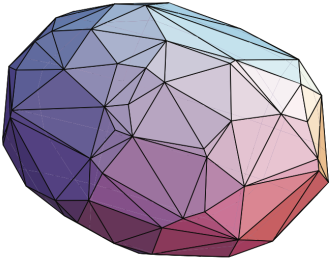
Algorithm	*2D Complexity*	*3D Complexity*
Incremental	$O(n^2)$	$O(n^2)$
Gift wrapping	$O(nh)$	$O(nf)$
Divide-and-conquer	$O(n \log n)$	$O(n \log n)$
Graham scan	$O(n \log n)$	?

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

Fecho convexo em 3D

Algoritmo incremental

- Dado um fecho Q, determinar o fecho de $Q \cup \mathbf{p}$
  - Determinar os planos tangentes
  - Arestas de silhueta suportam esses planos
  - Faces visíveis são removidas
  - Faces formadas pelas arestas e o ponto **p** são adicionadas



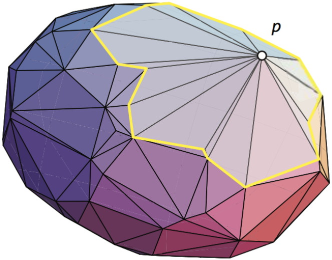


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Fecho convexo em 3D

Algoritmo dividir e conquistar

- Desafio: dados dois fechos A e B, determinar $conv(A \cup B)$
  - Determinar o plano π que suporta A e B por baixo
    - Considerar que este plano toca A e B nos pontos **a** e **b**
  - Rotacionar o semi-plano ao redor de $\overline{\mathbf{ab}}$ até tocar **r**
    - **r** é vizinho de **a** ou de **b**, o que torna a busca local
  - Repetir a rotação até fechar o cilindro de faces a incluir
  - Descartar as faces dentro desse cilindro de A e B

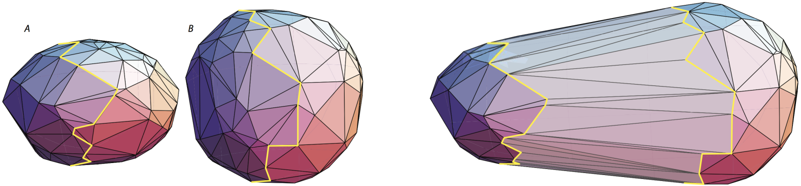


Figura extraída de Discrete and Computational Geometry, Devadoss and Rourke, 2011

Aplicação: visibilidade de pontos

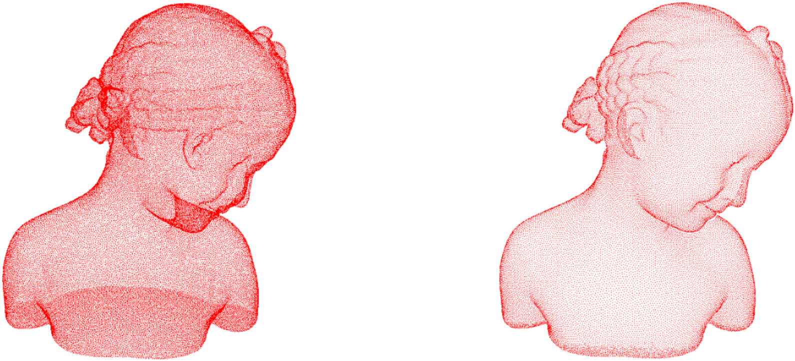
Problema:

- Dada um nuvem densa de pontos P, determinar o conjunto de pontos visíveis V do ponto de vista do ponto $\mathbf{c}$

Aplicação: visibilidade de pontos

Problema:

- Dada um nuvem densa de pontos P, determinar o conjunto de pontos visíveis V do ponto de vista do ponto $\mathbf{c}$



Ref: "Direct Visibility of Point Sets", S. Katz, A. Tal, R. Basri, 2007

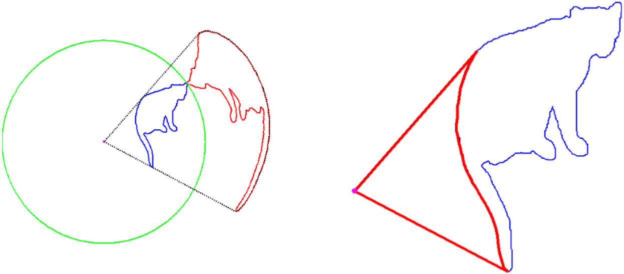
Aplicação: visibilidade de pontos

Solução:

- Inversão dos pontos: espelhamento esférico

$$\mathbf{p}_i' = f(\mathbf{p}_i) = \mathbf{p}_i + 2(r - \|\mathbf{p}_i\|)\frac{\mathbf{p}_i}{\|\mathbf{p}_i\|}$$

- Constrói o fecho convexo $conv(\{\mathbf{p}_i'\}, \mathbf{c})$
  - Pontos do fecho são os pontos visíveis



Trabalho

- Implemente um algoritmo para calcular o fecho convexo 2D de uma nuvem de pontos. Seu algoritmo deve receber como entrada a nuvem de pontos e ter como saída a lista de índices dos pontos que pertencem ao fecho convexo, em ordem antihorária.
  - Considere como entrada uma lista de pontos representando uma nuvem de pontos, armazenada em um arquivo com o seguinte formato:

    ```
    0 x_0 y_0
    1 x_1 y_1
    ...
    n-1 x_n-1  y_n-1
    ```

  - Como saída, espera-se uma lista de IDs dos vértices que pertencem ao fecho:

    ```
    id_i
    id_j
    ...
    ```

  - Você pode assumir que não existem pontos colineares nem pontos com mesmo valores de coordenadas x ou y.