Geometria Computacional

Professor:

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

Conteúdo (aula 6):

- Fecho Convexo 2D

Roteiro

- Introdução
- Noções de convexidade
- Algoritmo Incremental
- Algoritmo “Embrulho para presente” (Gift Wrapping)
- Algoritmo Graham Scan
- Algoritmo Dividir para Conquistar

Fecho Convexo: introdução

- O problema de computar o fecho convexo de um conjunto de pontos é central para muitos problemas em G. C.

- Ele tem sido estudado extensivamente e tem aplicações em inúmeras áreas tais como:

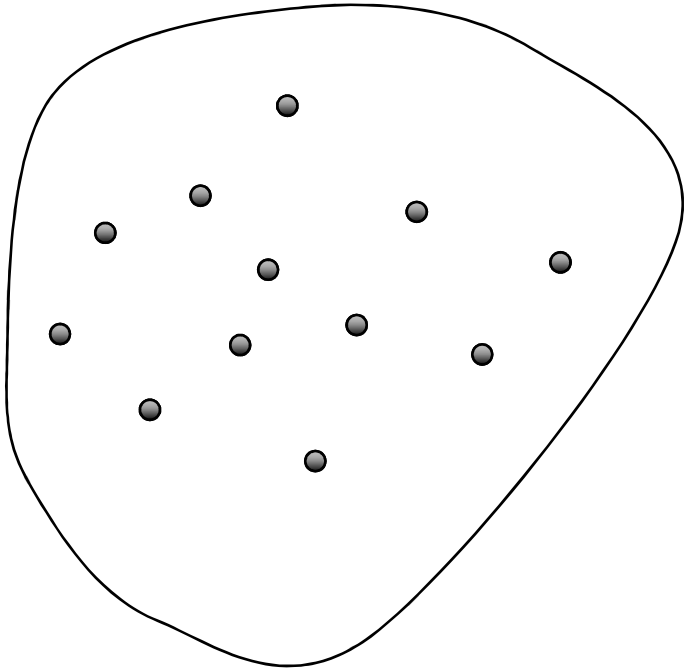
  - Processamento de imagens
  - Reconhecimento de padrões
  - Pesquisa operacional

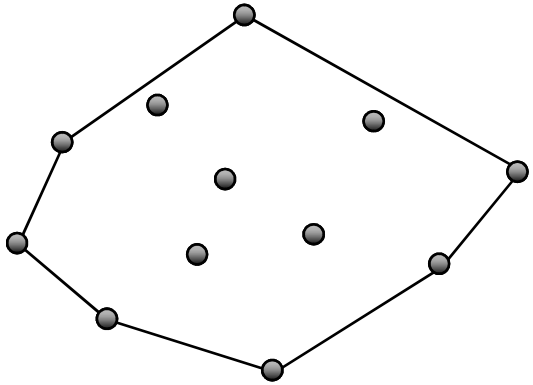
• Fecho Convexo: introdução

- *Definição 6.1* (Fecho Convexo): o fecho convexo de um conjunto de pontos S é o menor conjunto convexo que contêm todos os pontos em S.
- Definição simples de entender
- Mas para isso precisamos compreender a noção de convexidade

- **Fecho Convexo: introdução**

  - Intuitivamente, podemos entender o fecho convexo imaginando uma borracha elástica que envolve o conjunto de pontos e que depois é solta até alcançar a condição de equilíbrio





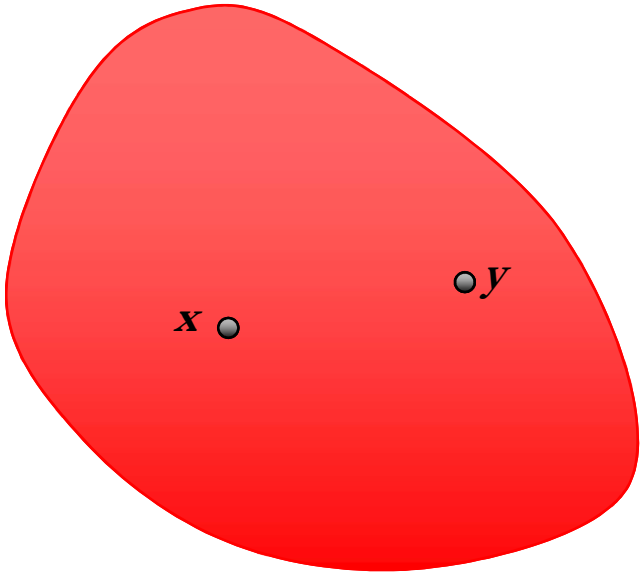
Fecho Convexo: noções de convexidade

- Utilizaremos uma noção baseada em visibilidade para chegar a formas computacionais de tratar o problema de obter o fecho convexo

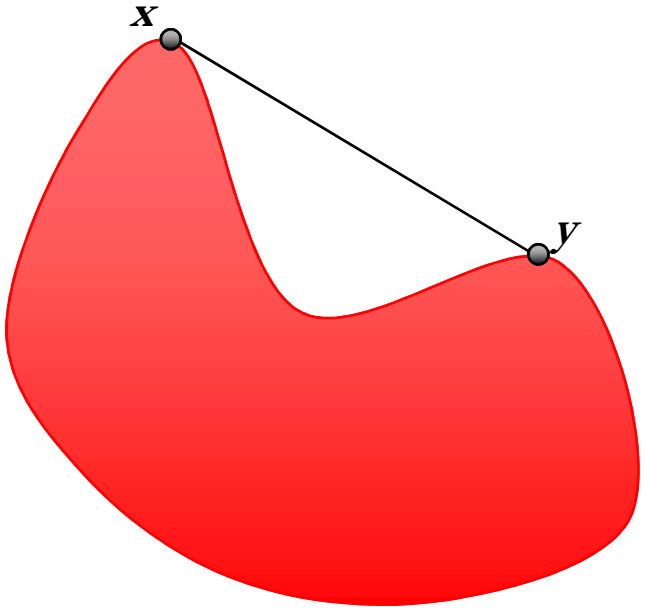
- Definição 6.3 (Região Convexa) - Uma região é *R* é dita convexa se dados dois de seus pontos *x* e *y*, então *x* é visível a partir de y e vice-versa. Consideramos *x* e y mutuamente visíveis se o segmento de reta xy está completamente contido em *R*

Fecho Convexo: noções de convexidade

- Região Convexa



- Região não-convexa



Fecho Convexo: noções de convexidade

- Todo ponto no segmento de reta passando por x e y pode ser escrito como αx+βy, α≥0 e β≥0,α+β=1.
- Tal noção pode ser generalizada para um número arbitrário de pontos
- Definição 6.4 (Combinação Convexa) A combinação convexa de um conjunto de pontos $S = \{p_0,...,p_1\}$ é dada por:

$$\lambda_1 p_1 + \ldots + \lambda_n p_n,$$
$$\lambda_i \geq 0$$
$$\sum \lambda_i = 1$$

Fecho Convexo: noções de convexidade

- Teorema 6.1- Para um conjunto de pontos $S = \{p_0,...,p_1\}$, o fecho convexo é o conjunto de todas as combinações convexas de S

Fecho Convexo: noções de convexidade

- Prova – Seja M o conjunto de todas as combinações convexas de S

$$M = \{\lambda_1 p_1 + \ldots + \lambda_n p_n, \lambda_i \geq 0, \sum \lambda_i = 1\}$$

- Precisamos mostrar que

(I) conv(S) $\subseteq$M
(II) M $\subseteq$ conv(S)

Fecho Convexo: noções de convexidade

- Parte (I): conv(S) $\subseteq M$. Para mostrar que M contém S basta verificar que cada ponto p_i pode ser obtido fazendo λ_i=0 e λ_j=1, j≠i.

- Resta mostrar que M é convexo. Para isso considere dois pontos quaisquer x e y de M

$$x = \{\lambda_1 p_1 + \ldots + \lambda_n p_n\}, \lambda_i \geq 0, \sum \lambda_i = 1$$

$$y = \{\lambda_1' p_1 + \ldots + \lambda_n' p_n\}, \lambda_i' \geq 0, \sum \lambda_i' = 1$$

Fecho Convexo: noções de convexidade

- Parte (I): conv(S) $\subseteq M$ (continuação). O segmento xy pode ser escrito como

$$\alpha x + \beta y = \alpha \sum \lambda_i p_i + \beta \sum \lambda'_i p_i = \sum (\alpha_i \lambda_i + \beta \lambda'_i) p_i$$

- onde

$$\lambda_i \geq 0, \sum \lambda_i = 1$$

$$\lambda'_i \geq 0, \sum \lambda'_i = 1$$

Fecho Convexo: noções de convexidade

- Parte (I): conv(S) $\subseteq M$ (continuação). Logo,

$$\sum(\alpha_i\lambda_i + \beta\lambda_i') = \alpha\sum\lambda_i + \beta\sum\lambda_i' = \alpha\cdot 1 + \beta\cdot 1 = 1$$

- E, portanto, xy está contido em M o que permite afirma que M é convexo

Fecho Convexo: noções de convexidade

- Parte (II) (continuação) Precisamos mostrar que qualquer ponto em M está em conv(S).
- Caso base (n=1). M = conv(S) = p_1
- Hipótese indutiva(k<n). Todo conjunto S' com |S'|<n pontos está em conv(S)

Fecho Convexo: noções de convexidade

- Parte (II) (continuação): Considere um conjunto de pontos $S = \{p_0,\ldots,p_n\}$.
- Pela hipótese indutiva um ponto $x = \lambda_1' p_1 + \ldots + \lambda'_{n-1} p_{n-1}$ está em conv(S') desde que $\lambda_i' \geq 0, \sum \lambda_i' = 1$
- Como $\lambda_1 p_1 + \ldots + \lambda_{n-1} p_{n-1} = 1 - \lambda_n p_n$ escolhamos $\lambda_i' = \frac{\lambda_i}{1-\lambda_i}$
- Sabemos que x $\in$conv(s')$\subset$conv(S) e $p_n \in$ conv(S)

Fecho Convexo: noções de convexidade

- Parte (II) (continuação):
- Como x $\in$conv(S) e $p_n \in$ conv(S), o segmento entre x e p_n está em conv(S) porque conv(S) é convexo
- Finalmente, o segmento formado por *x* e p_n é dado por

$$(1-\lambda_n)\left(\frac{\lambda_1}{1-\lambda_n}p_1+\ldots+\frac{\lambda_{n-1}}{1-\lambda_n}p_{n-1}\right)+\lambda_n p_n=\lambda_1 p_1+\ldots+\lambda_n p_n$$

- O que mostra que $\lambda_1 p_1+\ldots+\lambda_n p_n$ pertence a conv(S) concluíndo a prova por indução

Fecho Convexo: algoritmo incremental

- O teorema 6.1 ainda não nos ajuda a encontrar um processo algoritmo para resolver o problema
- Vamos olhar o problema sobre o seu aspecto geométrico, que seja subjacente tanto as provas dos teoremas quanto aos algoritmos

Fecho Convexo: algoritmo incremental

- Primeiramente, o que entendemos como computar o fecho convexo?

- Computar o fecho convexo significa obter uma representação do bordo da região na forma de uma lista de pontos

Fecho Convexo: algoritmo incremental

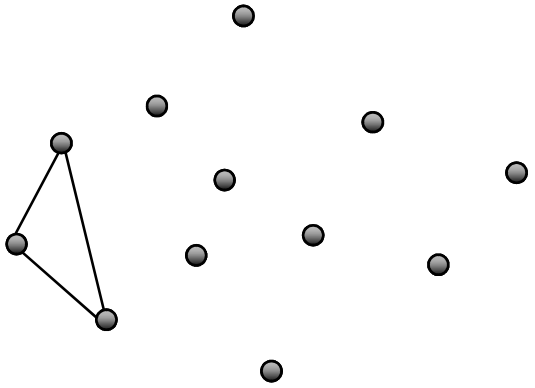
- Vamos propor um algoritmo incremental próximo ao raciocínio utilizado na prova por indução
- Suporemos construído o fecho convexo conv(H_k) para um conjunto H_k com k pontos e posteriormente iremos construir o fecho para k+1 pontos
- Como devemos prosseguir de modo que isto seja possível

Fecho Convexo: algoritmo incremental

- O segredo é ordenar os pontos de acordo com a abscissa *x*
- Assim podemos partir do fecho construído conv(H_k) para os *k* pontos a esquerda do ponto p_{k+1} atual para construir o fecho conv(H_{k+1}) com *k*+1 pontos

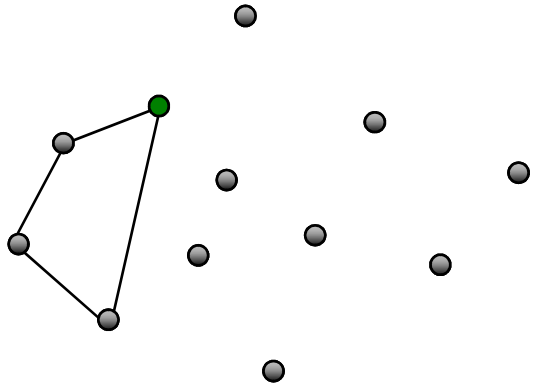
Fecho Convexo: algoritmo incremental

- Começamos construindo o fecho convexo conv(H_3) formado pelos três primeiros pontos do conjunto H_3



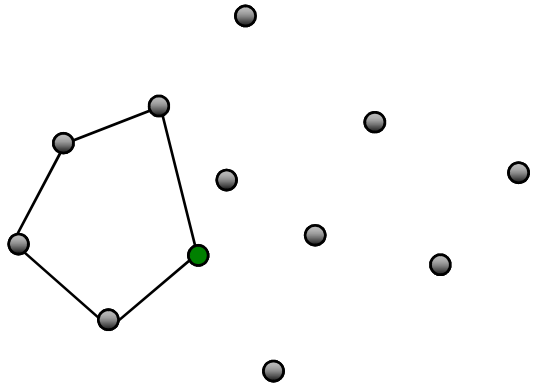
Fecho Convexo: algoritmo incremental

- Prosseguimos adicionando um novo ponto e construindo o novo fecho convexo conv(H_4). Observe que o novo ponto inserido faz parte de conv(H_4), já que é o extremo mais a direita do conjunto.



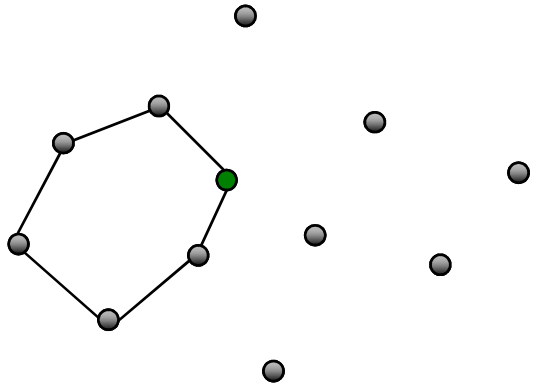
Fecho Convexo: algoritmo incremental

- Prosseguimos adicionando um novo ponto e construindo o novo fecho convexo conv(H_5)



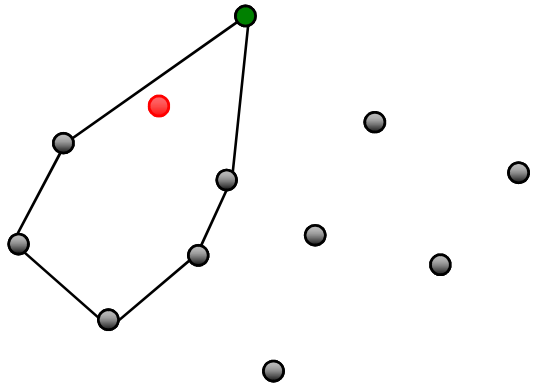
Fecho Convexo: algoritmo incremental

- Prosseguimos adicionando um novo ponto e construindo o novo fecho convexo conv(H_6)



Fecho Convexo: algoritmo incremental

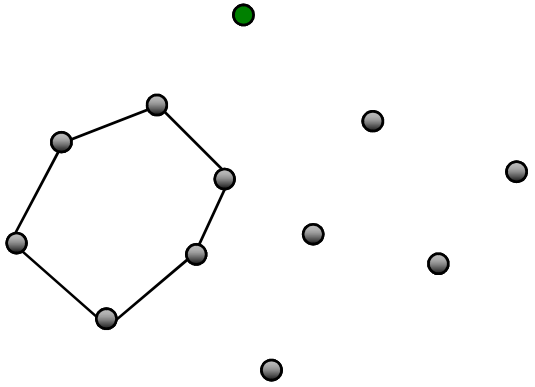
- Ao tentar construir o fecho conv(H_7) inserindo o sétimo ponto verificamos que alguns pontos do fecho anterior deixam de fazer parte do novo fecho (no caso um ponto, em vermelho)



Fecho Convexo: algoritmo “embrulho para presente”

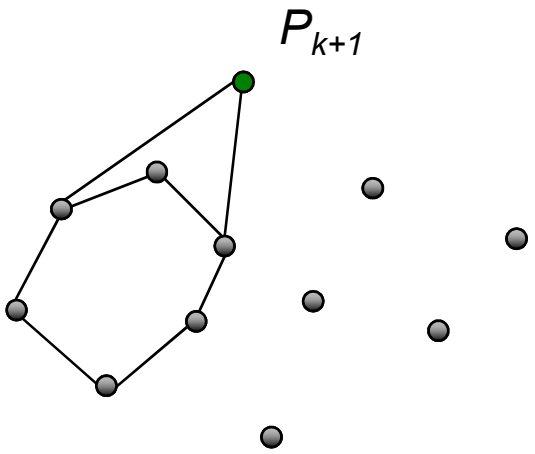
- Como resolver o problema de remover pontos que não fazem mais parte do fecho no novo estado do algoritmo?

- Considere a seguinte situação em que o ponto verde será adicionado ao conjunto para determinação do novo fecho corrente



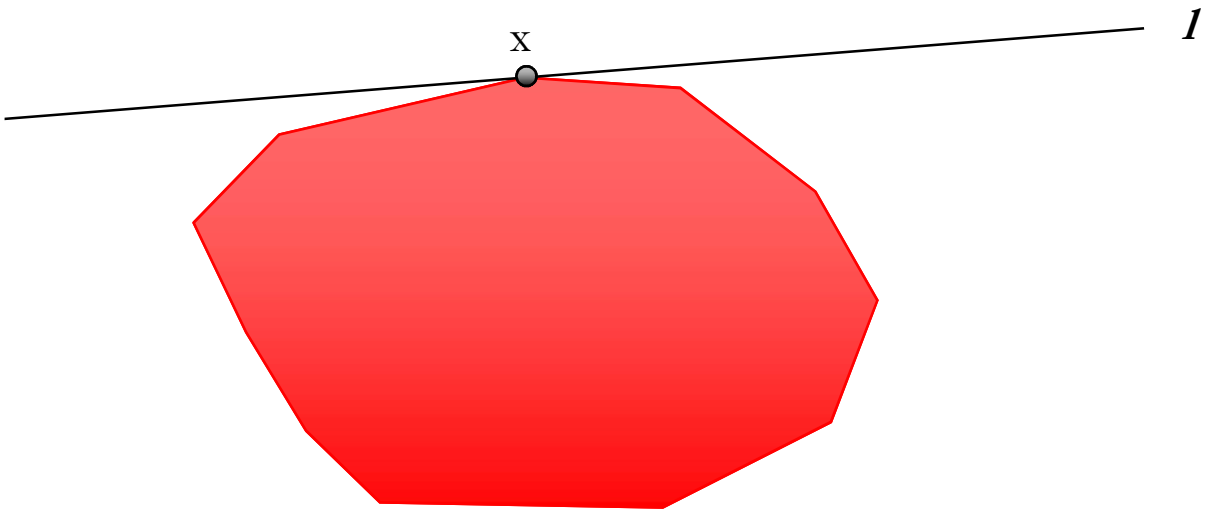
Fecho Convexo: algoritmo “embrulho para presente”

- Para resolver o problema verificamos que os pontos a serem removidos são os pontos visíveis a partir de p_{k+1} exceto os pontos que definem o limite da região visível



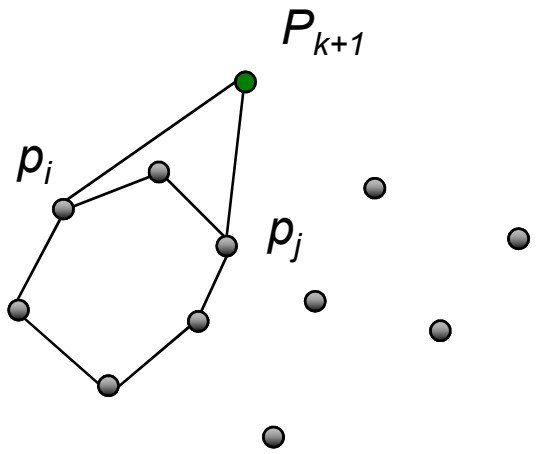
Fecho Convexo: algoritmo “embrulho para presente”

- Definição 6.5 – Seja *P* um polígono convexo e *x* um ponto na fronteira de *P*. Uma reta *l* passando em *x* suporta *P* se todo *P* está de um lado de *l*. Diz-se que *l* é tangente a *P* em *x*.



Fecho Convexo: algoritmo “embrulho para presente”

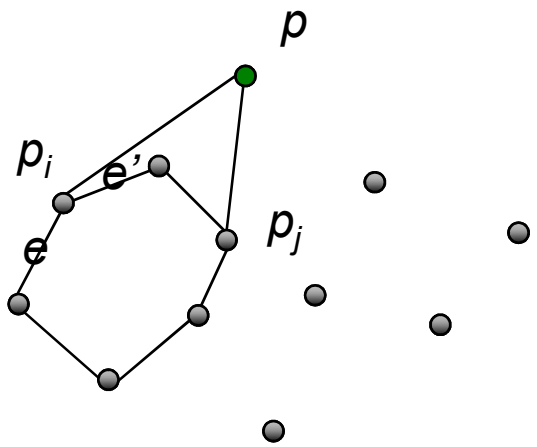
- Logo precisamos encontrar duas retas tangentes a conv(H_k)em pontos p_i e p_j, ambas passando pelo ponto p_{k+1}



- Como encontrar os vértices p_i e p_j?

Fecho Convexo: algoritmo “embrulho para presente”

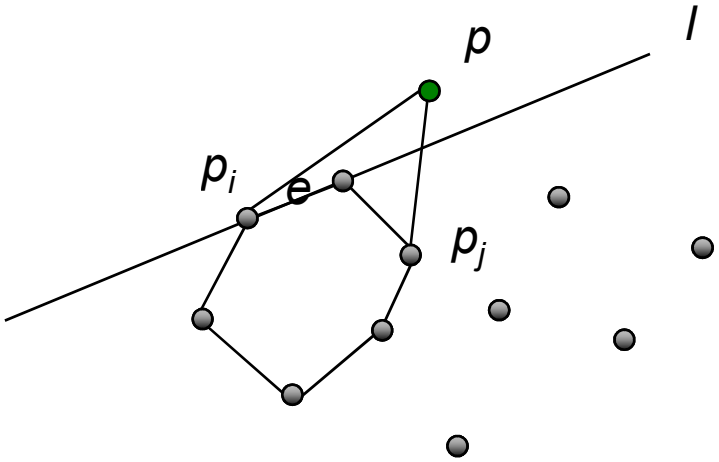
- O solução está em pensar em arestas e não em vértices
- Uma aresta, em relação a um ponto *p*, pode ser: visível, não visível ou estar sobre a mesma reta
- Ignoremos o último caso considerando que *S* não possui 3 pontos colineares



- Deste modo procuramos os vértices incidentes a duas arestas *e* e *e’* tal que uma seja visível e outra não em relação a p

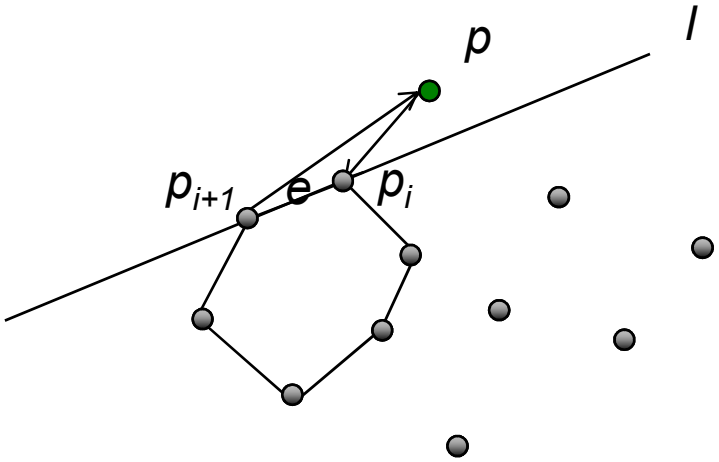
Fecho Convexo: algoritmo “embrulho para presente”

- Uma aresta *e* é visível a *p* se, em relação a reta suporte *l* que contém *e*, p e o conjunto P estão em lados opostos



Fecho Convexo: algoritmo “embrulho para presente”

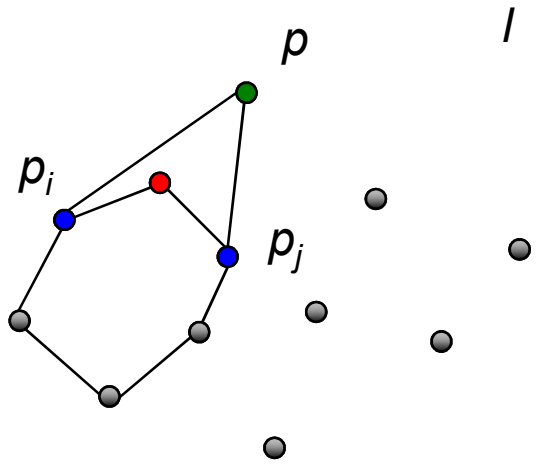
- Dada uma aresta e que passa por vértices p_i e p_{i+1}, e está visível em relação a p se os vetores p_i, p_{i+1} e p, em sequência, formam uma tripla orientada no sentido dos ponteiros do relógio (*clockwise*)



Fecho Convexo: algoritmo “embrulho para presente”

- Utilizamos então os seguintes passos:

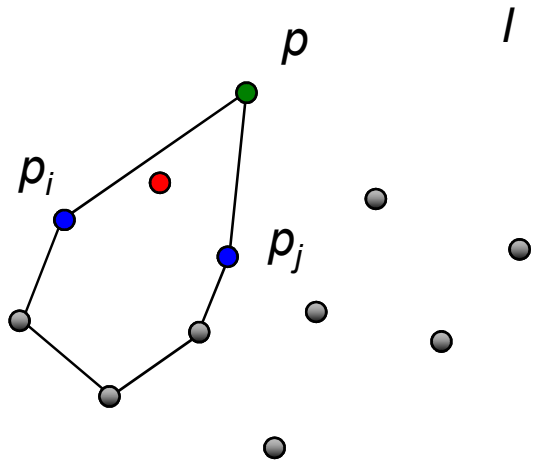
- percorremos as arestas de P
- identificamos os dois vértices p_i e p_j tal que $p\ p_i$ e $p\ p_j$ sejam retas suporte
- Removemos os vértices entre p_i e p_j em conv(H_k)
- Inserimos p_k+1 entre p_i e p_j



Fecho Convexo: algoritmo “embrulho para presente”

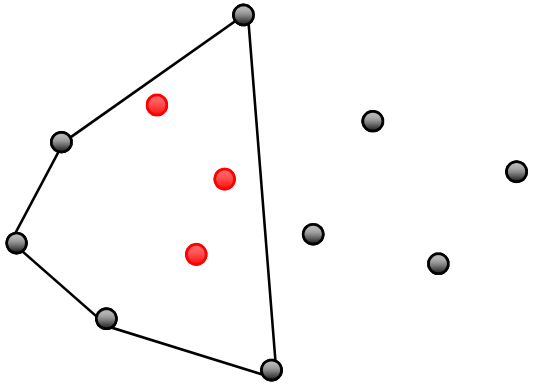
- Utilizamos então os seguintes passos:

- percorremos as arestas de P
- identificamos os dois vértices p_i e p_j
- Removemos os vértices entre p_i e p_j em conv(H_k)
- Inserimos p_k+1 entre p_i e p_j



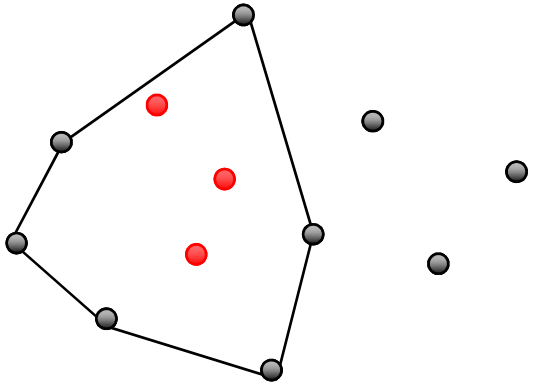
Fecho Convexo: algoritmo incremental

- Prosseguindo...



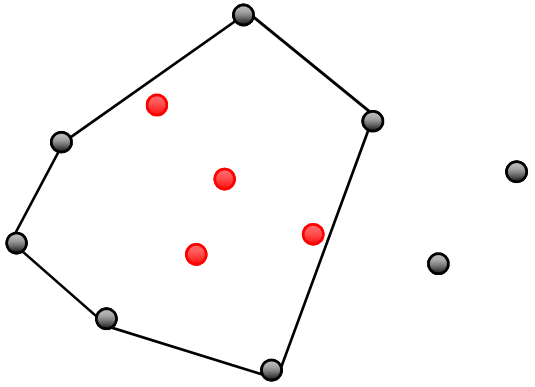
Fecho Convexo: algoritmo incremental

- Prosseguindo...



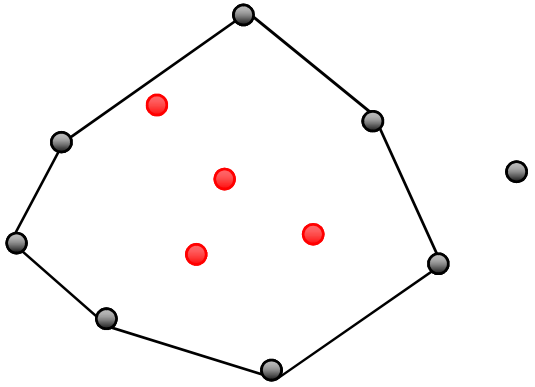
Fecho Convexo: algoritmo incremental

- Prosseguindo...



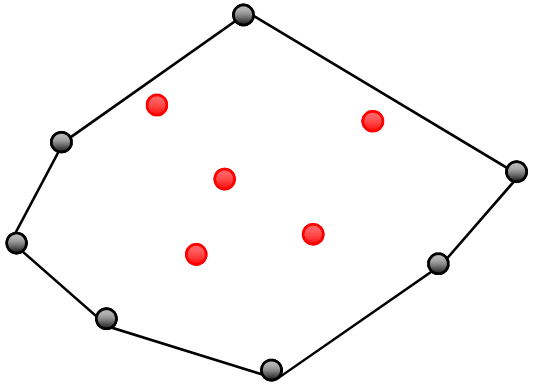
Fecho Convexo: algoritmo incremental

- Prosseguindo...



Fecho Convexo: algoritmo incremental

- Prosseguindo...



Fecho convexo: algoritmo incremental

- Análise de Complexidade:
- O primeiro passo do algoritmo consiste em ordenar os vértices em relação ao eixo das abcissas com custo O($n$log$n$).
- Para cada ponto, a partir dos três primeiro é necessário testar cada uma das k-1 arestas com relação a visibilidade. Admitindo custo O(1) para o teste de visibilidade temos

$$3+4+\ldots+(n-1)=\frac{n(n-1)}{2}-(1+2)=\frac{n^2}{2}-\frac{n}{2}-3$$

- Logo, a complexidade assintótica do algoritmo é O(n^2)

Fecho convexo: algoritmo incremental

- A complexidade $O(n^2)$ do algoritmo incremental parece aceitável
- Entretanto, para um número muito grande de pontos a solução pode ser custosa demais para o tempo disponível para resolver o problema
- A seguir veremos outros algoritmos que conseguem ser mais eficientes que o algoritmo incremental

- **Fecho convexo: Algoritmo "Embrulho para presente" (Gift Wrapping)**

  - Um dos problemas com o algoritmo incremental é o de que ele processa todos os pontos do conjunto S mesmo que os pontos do fecho sejam em número bem menor

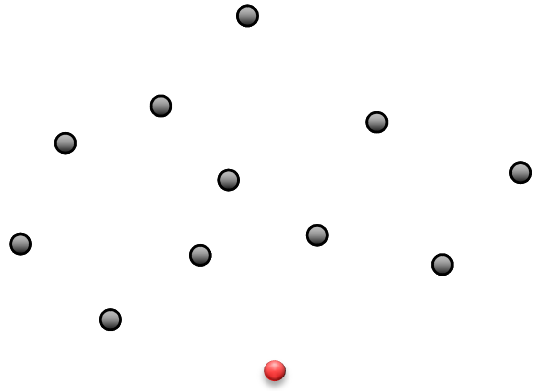
  - Uma possível ideia é "embrulhar" o conjunto de pontos com uma fita a partir de um dos extremos do conjunto

  - Lembre que os extremos de um conjunto estão garantidamente no fecho

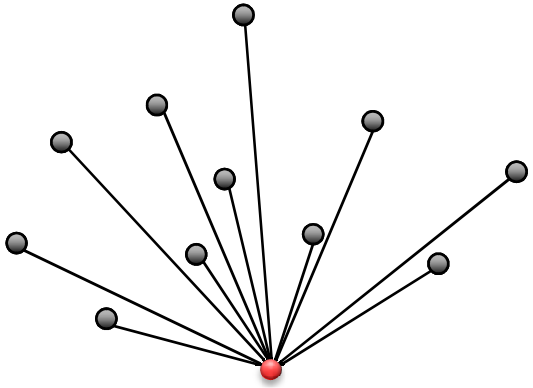
- **Fecho convexo: Algoritmo "Embrulho para presente" (Gift Wrapping)**

  - Seja *S* um conjunto de pontos em posição geral tal que não exista 3 pontos colineares

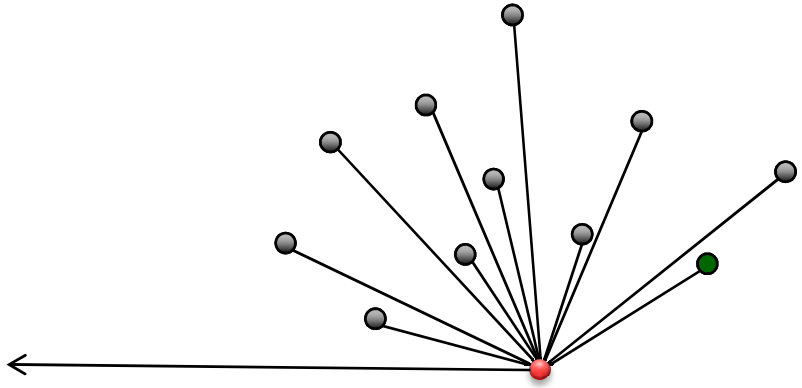
  - Considere o ponto mais inferior, em caso de empate escolha o mais a direita. Tal ponto será a âncora da fita ao redor de S



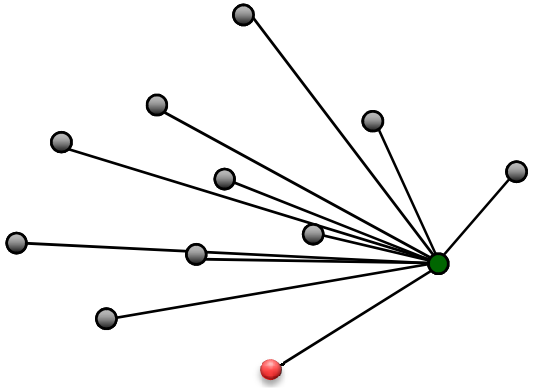
- **Fecho convexo: Algoritmo "Embrulho para presente" (Gift Wrapping)**
  - Trace segmentos de reta para cada um dos pontos do conjunto a partir da âncora



- **Fecho convexo: Algoritmo "Embrulho para presente" (Gift Wrapping)**
  - Escolha como nova âncora o vértice extremo do segmento que forma o maior ângulo com a reta horizontal

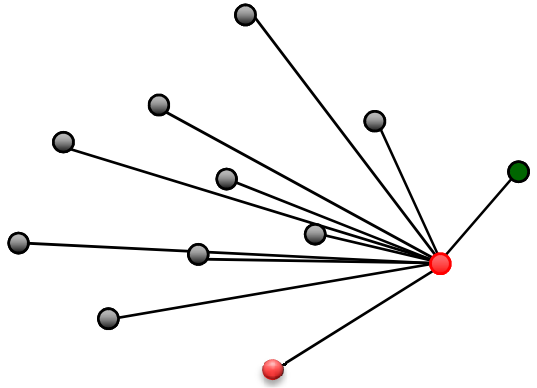


- **Fecho convexo: Algoritmo "Embrulho para presente" (Gift Wrapping)**
  - A partir da nova âncora, trace novamente segmentos de reta em direção aos demais vértices



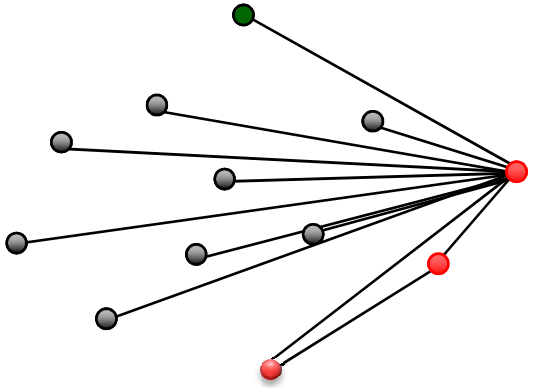
- **Fecho convexo: Algoritmo "Embrulho para presente" (Gift Wrapping)**

  - Escolha agora como nova âncora o vértice extremo do segmento que forma o maior ângulo com a última aresta construída do fecho



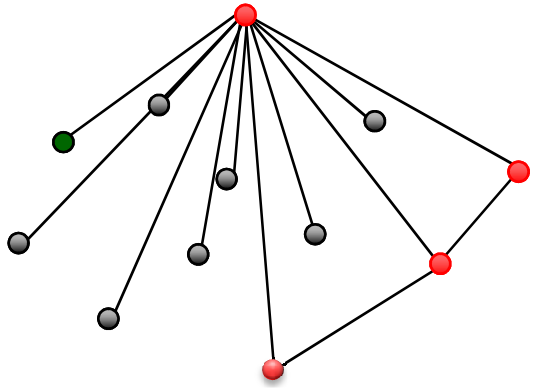
- **Fecho convexo: Algoritmo "Embrulho para presente" (Gift Wrapping)**

  - Escolha agora como nova âncora o vértice extremo do segmento que forma o maior ângulo com a última aresta construída do fecho



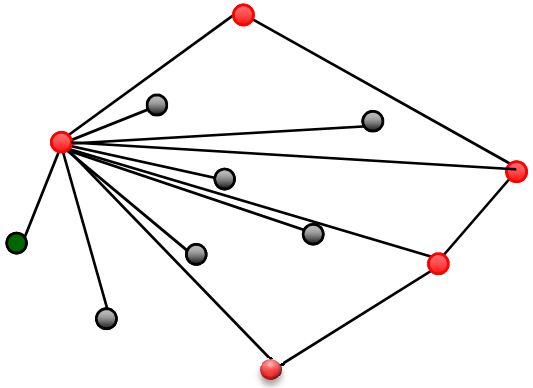
- **Fecho convexo: Algoritmo "Embrulho para presente" (Gift Wrapping)**

  - Escolha agora como nova âncora o vértice extremo do segmento que forma o maior ângulo com a última aresta construída do fecho



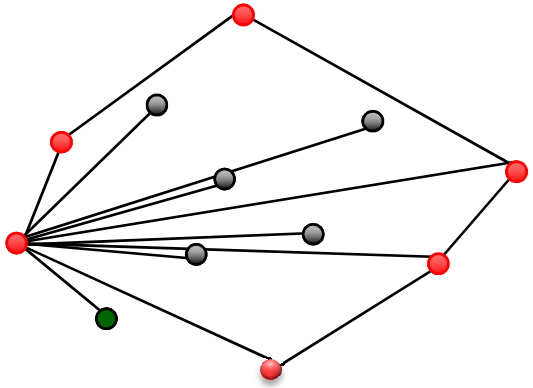
- **Fecho convexo: Algoritmo "Embrulho para presente" (Gift Wrapping)**

  - Escolha agora como nova âncora o vértice extremo do segmento que forma o maior ângulo com a última aresta construída do fecho



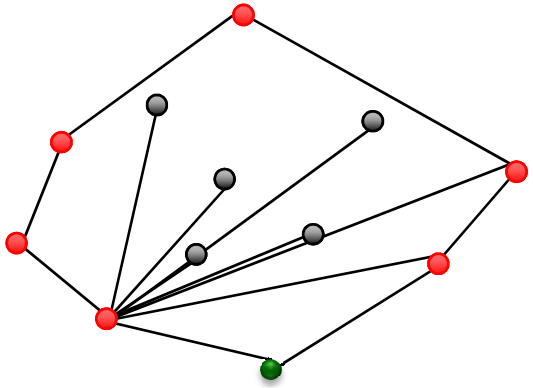
- **Fecho convexo: Algoritmo "Embrulho para presente" (Gift Wrapping)**

  - Escolha agora como nova âncora o vértice extremo do segmento que forma o maior ângulo com a última aresta construída do fecho



- **Fecho convexo: Algoritmo "Embrulho para presente" (Gift Wrapping)**

  - Escolha agora como nova âncora o vértice extremo do segmento que forma o maior ângulo com a última aresta construída do fecho



- **Fecho convexo: Algoritmo "Embrulho para presente" (Gift Wrapping)**
  - Complexidade:
  - Em cada ponto, o ângulo para todos os demais pontos deve ser calculado, que são em um total de n
  - Como isto deve ser feito para cada ponto do fecho, supondo S com n pontos e h o número de pontos no fecho, temos a complexidade total de O(nh)
  - No pior caso h = n e o algoritmo será $O(n^2)$
  - A vantagem do algoritmo Gift Wrapping é que ele é sensível ao tamanho da saída

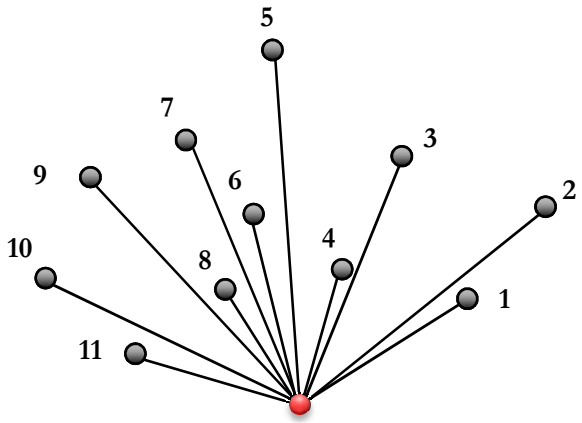
• Fecho convexo: Algoritmo Graham Scan

- Será possível obter um algoritmo ainda mais eficiente?
- Em 1972, Graham consegui o feito, publicando possivelmente o primeiro artigo em G.C.
- Ao invés de calcular o ângulo em cada ponto, Graham propôs iniciar o algoritmo ordenado os pontos em função de tal ângulo
- Após a ordenação, com um pouco mais de esforço, o algoritmo é capaz de eliminar os pontos que não pertenciam ao fecho

• Fecho convexo: Algoritmo Graham Scan

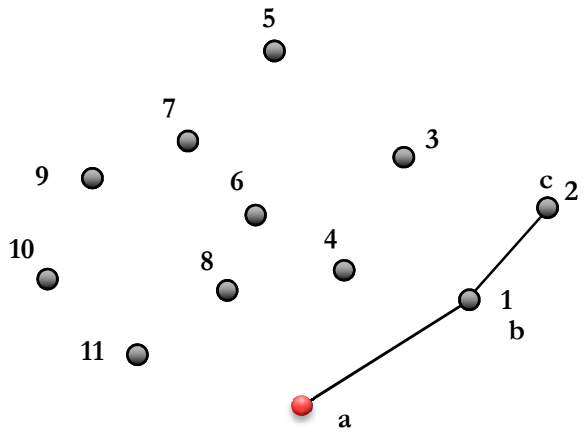
- Para um conjunto de pontos *S*, em posição geral, começamos escolhendo o ponto mais inferior a direita

- Em seguida, ordenamos os demais pontos de *S* pelo ângulo que os segmentos formam com a reta horizontal, do maior, para o menor



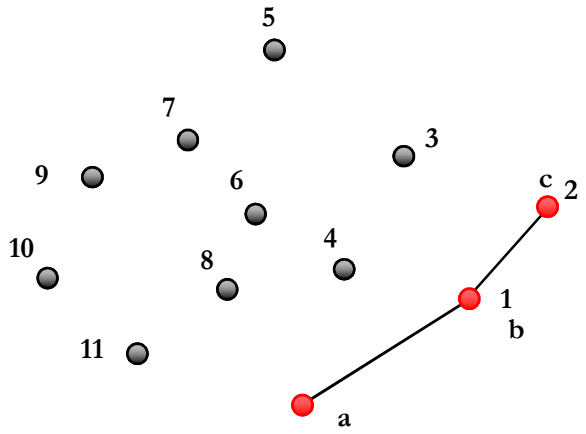
• Fecho convexo: Algoritmo Graham Scan

- Para cada ponto c verifica-se se os extremos a e b do último segmento construído e c formam um giro para a esquerda ou para a direita



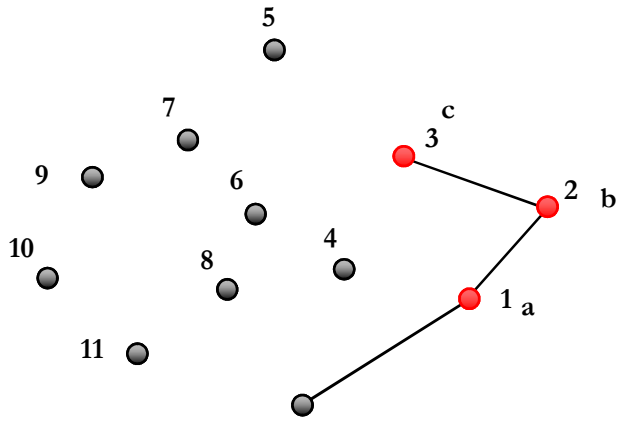
• Fecho convexo: Algoritmo Graham Scan

- Se abc faz um giro a direita então em b é formado um ângulo reflexo. Logo b deve ser removido do fecho em construção. O processo de descarte continua enquanto os últimos três pontos formem um ângulo reflexo



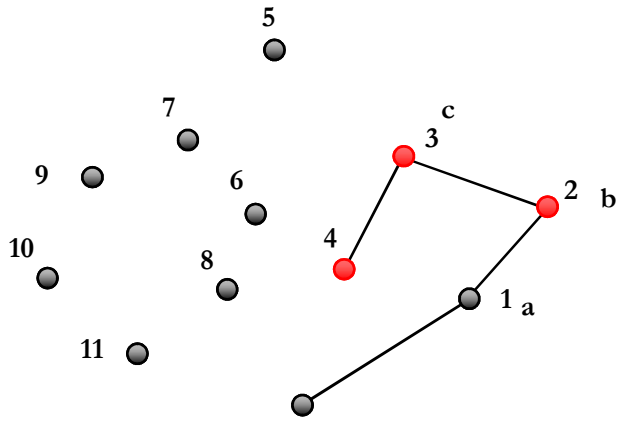
• Fecho convexo: Algoritmo Graham Scan

- Se abc faz um giro a direita então em b é formado um ângulo reflexo. Logo b deve ser removido do fecho em construção. O processo de descarte continua enquanto os últimos três pontos formem um ângulo reflexo



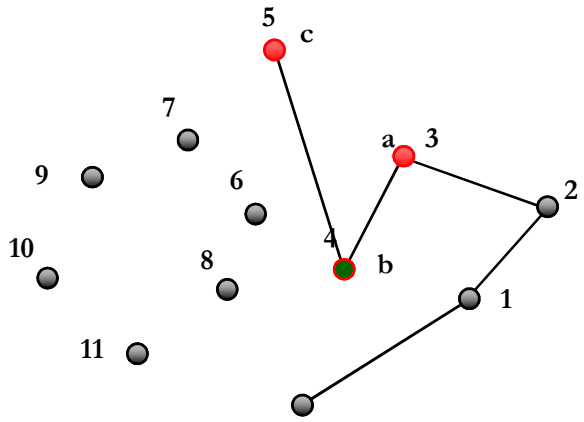
• Fecho convexo: Algoritmo Graham Scan

- Se abc faz um giro a direita então em b é formado um ângulo reflexo. Logo b deve ser removido do fecho em construção. O processo de descarte continua enquanto os últimos três pontos formem um ângulo reflexo



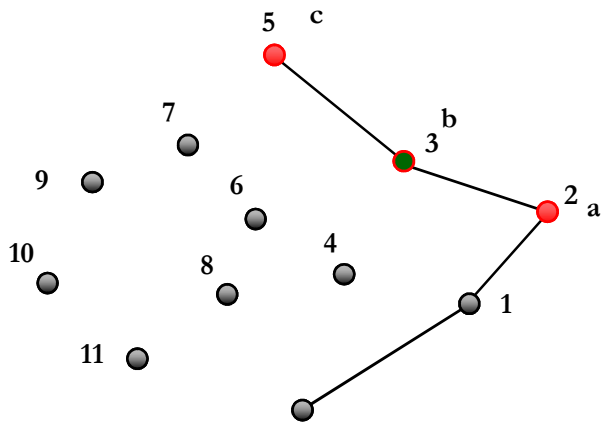
• Fecho convexo: Algoritmo Graham Scan

- Se abc faz um giro a direita então em b é formado um ângulo reflexo. Logo b deve ser removido do fecho em construção. O processo de descarte continua enquanto os últimos três pontos formem um ângulo reflexo



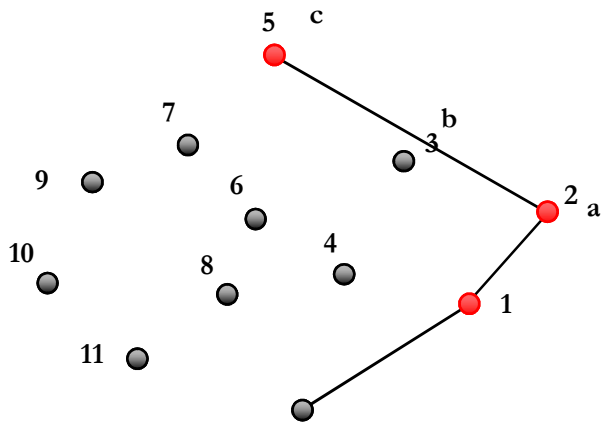
• Fecho convexo: Algoritmo Graham Scan

- Se abc faz um giro a direita então em b é formado um ângulo reflexo. Logo b deve ser removido do fecho em construção. O processo de descarte continua enquanto os últimos três pontos formem um ângulo reflexo



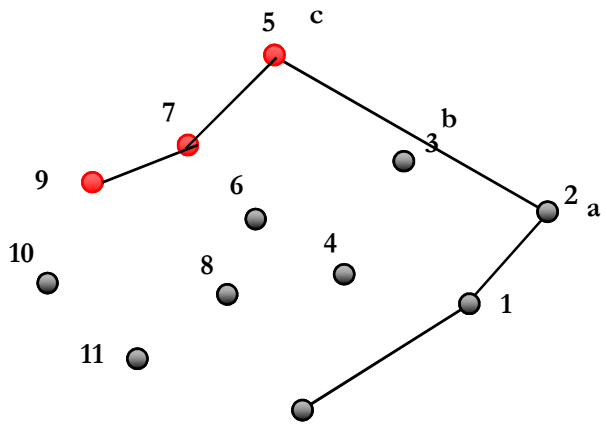
• Fecho convexo: Algoritmo Graham Scan

- Se abc faz um giro a direita então em b é formado um ângulo reflexo. Logo b deve ser removido do fecho em construção. O processo de descarte continua enquanto os últimos três pontos formem um ângulo reflexo



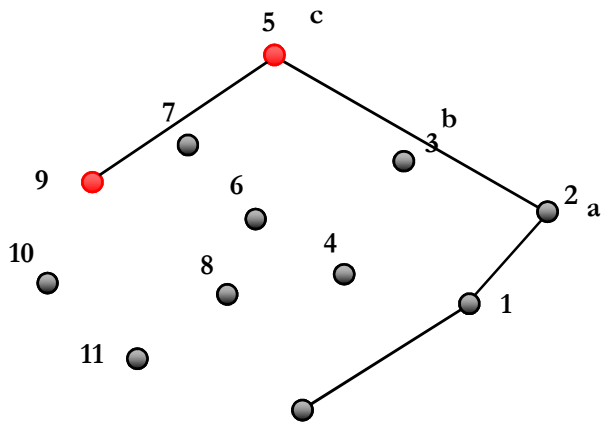
• Fecho convexo: Algoritmo Graham Scan

- Se abc faz um giro a direita então em b é formado um ângulo reflexo. Logo b deve ser removido do fecho em construção. O processo de descarte continua enquanto os últimos três pontos formem um ângulo reflexo



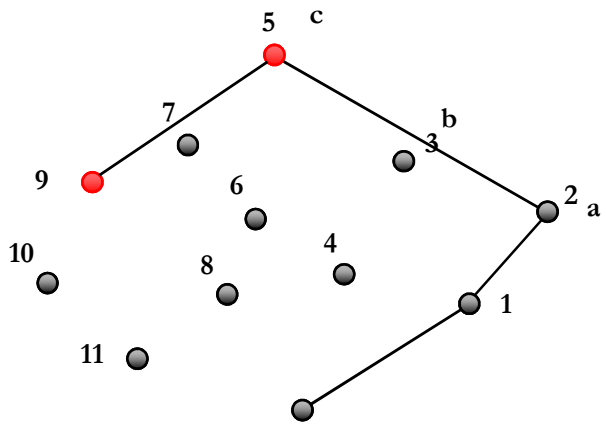
• Fecho convexo: Algoritmo Graham Scan

- Se abc faz um giro a direita então em b é formado um ângulo reflexo. Logo b deve ser removido do fecho em construção. O processo de descarte continua enquanto os últimos três pontos formem um ângulo reflexo



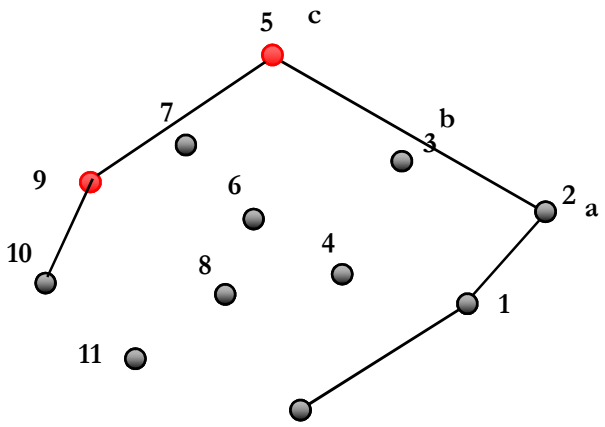
• Fecho convexo: Algoritmo Graham Scan

- Se abc faz um giro a direita então em b é formado um ângulo reflexo. Logo b deve ser removido do fecho em construção. O processo de descarte continua enquanto os últimos três pontos formem um ângulo reflexo



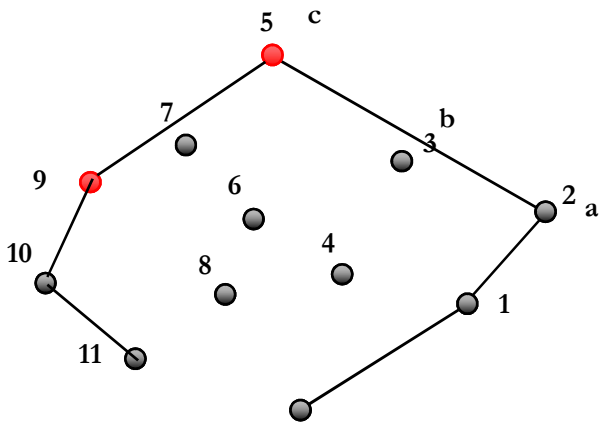
• Fecho convexo: Algoritmo Graham Scan

- Se abc faz um giro a direita então em b é formado um ângulo reflexo. Logo b deve ser removido do fecho em construção. O processo de descarte continua enquanto os últimos três pontos formem um ângulo reflexo



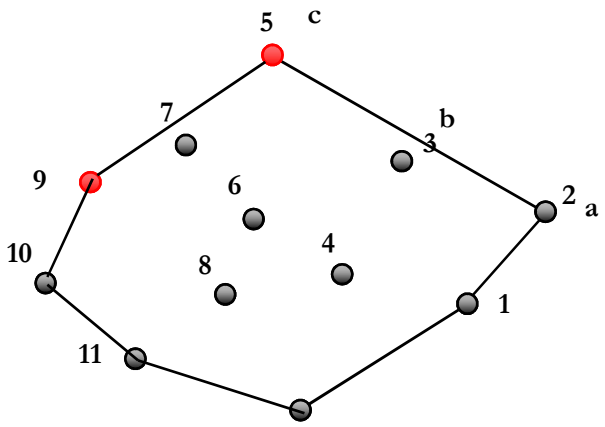
• Fecho convexo: Algoritmo Graham Scan

- Se abc faz um giro a direita então em b é formado um ângulo reflexo. Logo b deve ser removido do fecho em construção. O processo de descarte continua enquanto os últimos três pontos formem um ângulo reflexo



• Fecho convexo: Algoritmo Graham Scan

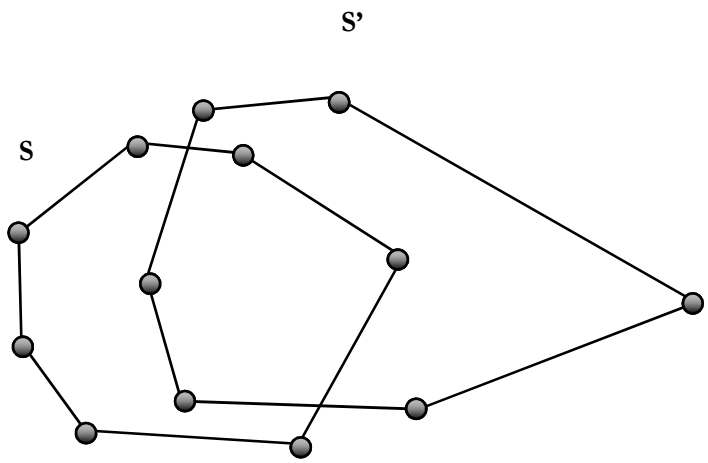
- Se abc faz um giro a direita então em b é formado um ângulo reflexo. Logo b deve ser removido do fecho em construção. O processo de descarte continua enquanto os últimos três pontos formem um ângulo reflexo



- **Fecho convexo: Algoritmo Divisão-por-Conquista**

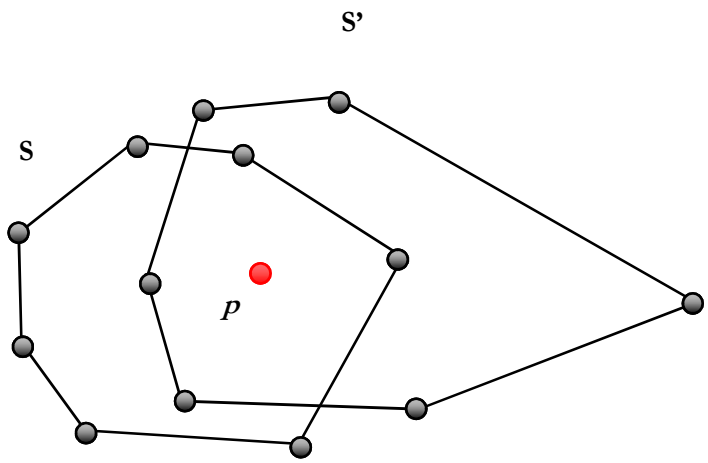
  - Uma outra possível estratégia é usar o paradigma de construção de algoritmos por divisão e conquista

  - Considere abaixo o seguinte problema: dados dois conjuntos de pontos S e S', para os quais foram construídos os fechos convexos conv(S) e conv(S'), determinar o fecho conv(S+S')



- **Fecho convexo: Algoritmo Divisão-por-Conquista**

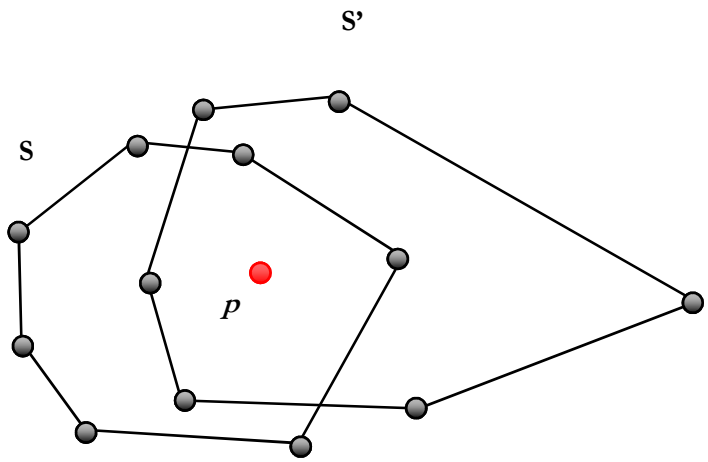
  - Primeiramente especificamos um ponto p no interior de S, por exemplo, o centróide de três pontos de S; p é interior a conv(S+S’)
  - Determine se p é interno a S’ – pode ser resolvido em tempo O(n)



- **Fecho convexo: Algoritmo Divisão-por-Conquista**

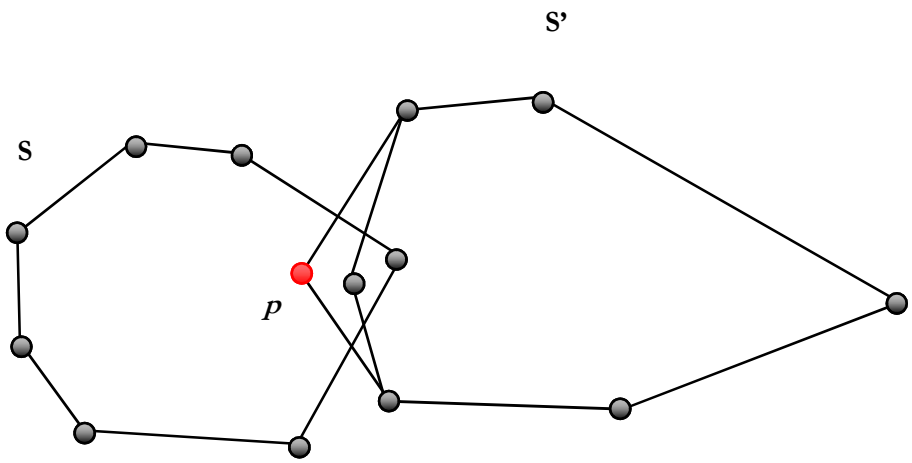
  - Caso(I) p interno a S'
  - Teorema – vértices consecutivos de um polígono convexo ocorrem em ordem angular em torno de um ponto interior

  - 1 - Podemos efetuar o merge entre os pontos de *S* e *S'* em tempo O(*n*) usando o ângulo formado pelo segmento *pip* com a linha horizontal
  - 2 - Usar o passo do Graham Scan O(*n*)



- **Fecho convexo: Algoritmo Divisão-por-Conquista**

  - Caso(II) p externo a S'
  - 1 - Determine os pontos u e v que determinam duas retas suporte com p. u e v separam os vértices em S' em duas partes, uma cadeia visível a p e a outra não visível. Desconsidere a cadeia visível
  - 2 – Aplique o passo Graham Scan



- **Fecho convexo: Algoritmo Divisão-por-Conquista**

Algoritmo MergeHull(S)

Entrada: um conjunto de pontos S

Saída: conv(S)

1. Se |S|<k0 (k0 é um inteiro pequeno), construa o fecho convexo diretamente usando algum método e pare, caso contrário vá para o passo 2
2. Particione o conjunto original S arbitrariamente em dois subconjuntos S1 e S2 com cardinalidades aproximadamente iguais
3. conv(S) <- MergeHull(S1); conv(S2) <- MergeHull(S2)
4. conv(S) <- Merge(conv(S1),conv(S2)).

- **Fecho convexo: cota inferior para o problema de fecho convexo**

  - Iremos agora determinar a cota inferior para o problema do fecho convexo

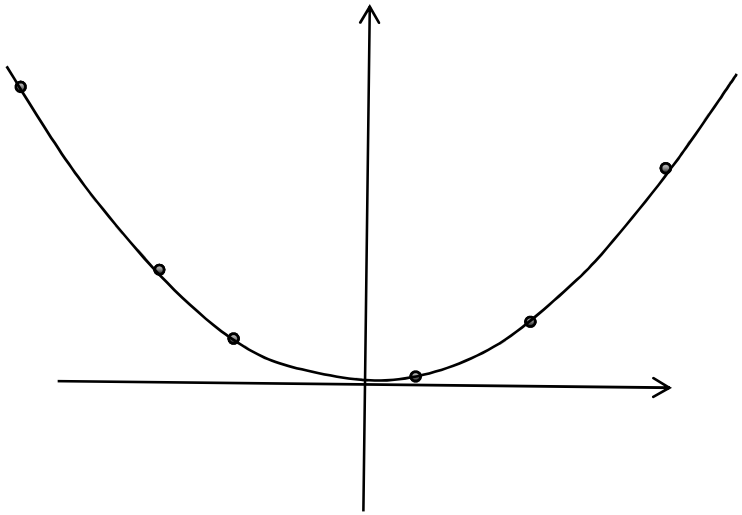
  - Para isto iremos mostrar que Ordenação < ConvexHull em tempo linear

- **Fecho convexo: cota inferior para o problema de fecho convexo**

  - Considere U={x0,x1,x2,...,xn} um conjunto de número reais não negativos.
  - Seja Iord uma instância do problema de ordenar U.
  - Iremos construir uma instância Iconv do problema de fecho convexo de um conjunto de pontos $S = (p0, p1, ..., pn)$, em tempo linear, a partir de uma instância Iord, tal que a solução para Iord pode ser obtida em tempo linear a partir de conv(U)

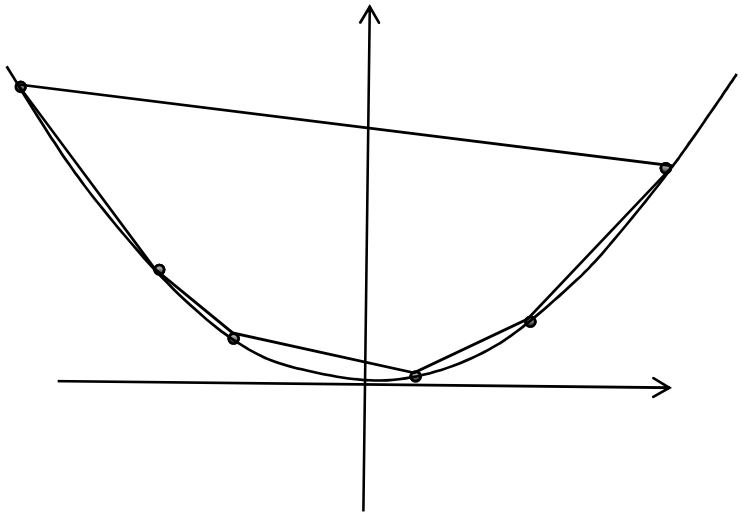
- **Fecho convexo: cota inferior para o problema de fecho convexo**

  - Para cada real $x_i \in U$ construímos um par (x_i, x_i^2)



- **Fecho convexo: cota inferior para o problema de fecho convexo**

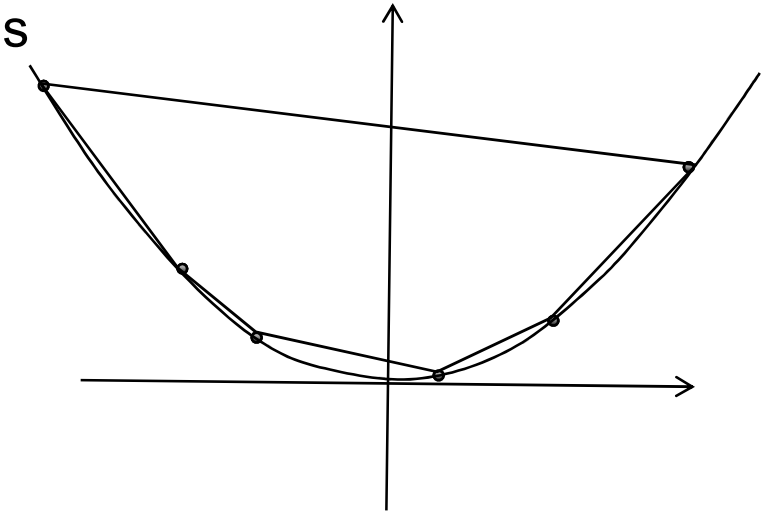
  - Solucionamos o problema de fecho convexo



- **Fecho convexo: cota inferior para o problema de fecho convexo**

  - Observe que o fecho convexo gera um lista de pontos ordenados em relação ao eixo das abscissas

  - Logo, uma solução para o problema de ordenação pode ser obtido através



Fecho convexo: Referências

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

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