Geometria Computacional
Fecho Convexo II

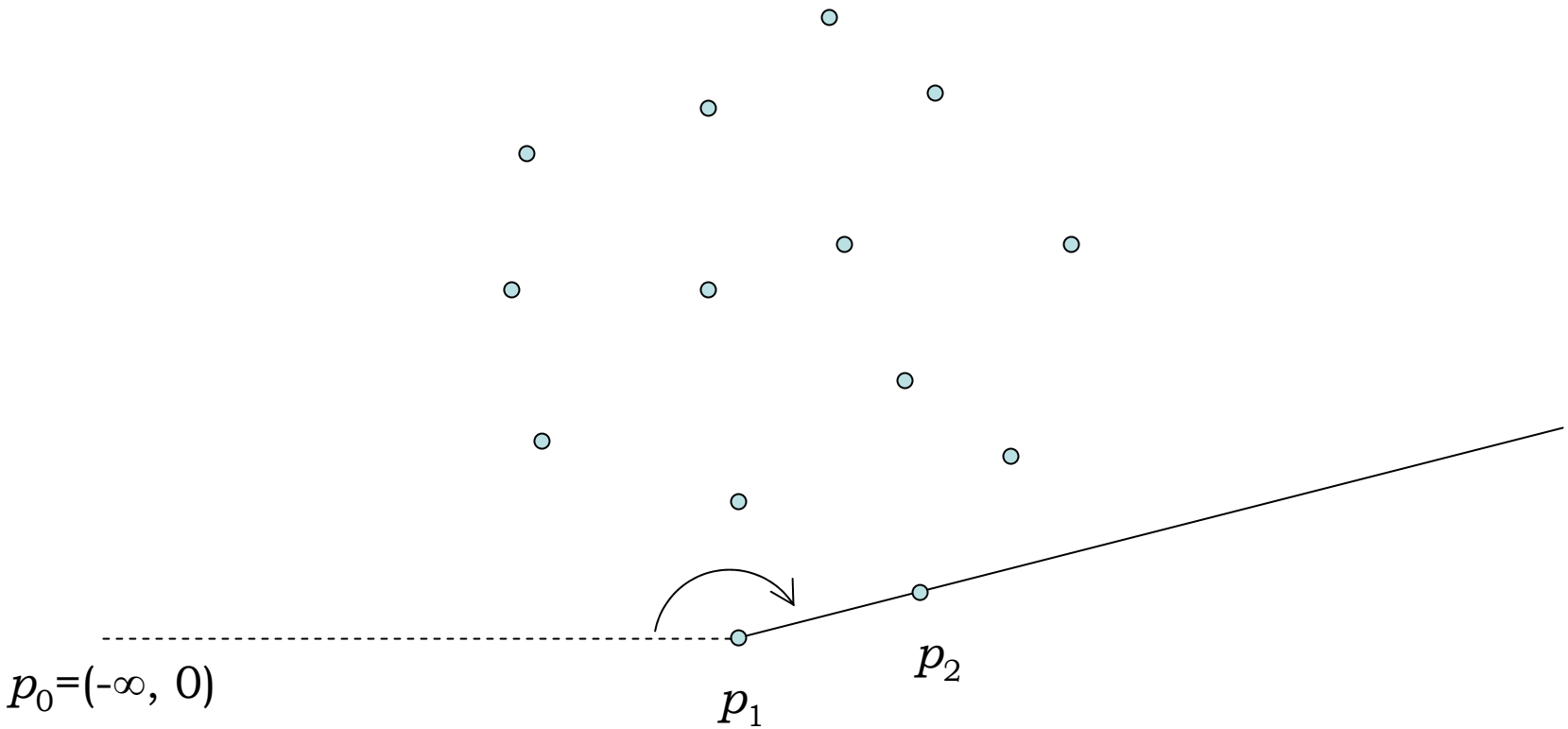
Claudio Esperança

Paulo Roma Cavalcanti

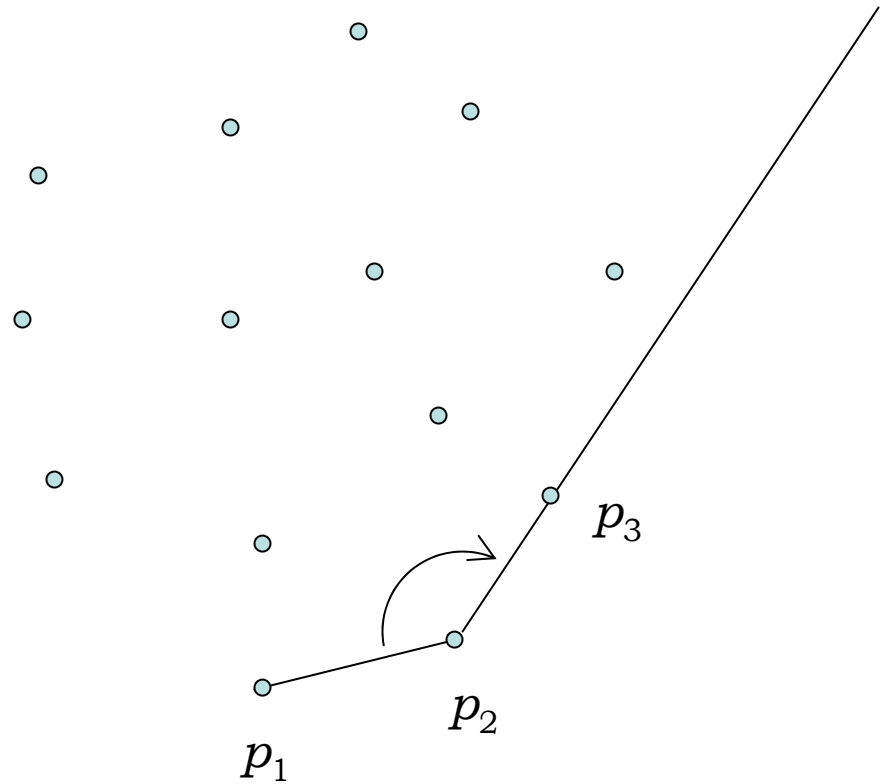
Marcha de Jarvis (*Gift Wrapping*)

- Análogo ao algoritmo de ordenação "*Selection Sort*"
  - A cada passo, escolhe o menor dos valores e acrescenta à coleção ordenada
  - Analogamente, deseja-se obter a cada passo o próximo ponto do fecho convexo (em ordem anti-horária)
    - Se $p_{i-1}p_i$ é a última aresta acrescentada, o próximo vértice p_{i+1} é aquele que maximiza o ângulo $p_{i-1}p_i\,p_{i+1}$
    - Durante a iniciação assume-se p_1 = ponto com menor coordenada y e p_0 um ponto com coordenada $x = -\infty$

Marcha de Jarvis (*Gift Wrapping*)



Marcha de Jarvis (*Gift Wrapping*)



Marcha de Jarvis - Complexidade

- Claramente, escolher o próximo vértice é $O(n)$
- Se o fecho convexo tem h vértices, então, o algoritmo tem complexidade $O(nh)$
  - No pior caso, $n \approx h$ e o algoritmo tem complexidade $O(n^2)$, pior, portanto que o algoritmo de Graham
  - No melhor caso, h é $o(\log n)$, isto é, assintoticamente menor que $\log n$, o que o torna melhor que o algoritmo de Graham

Algoritmo Sensível à Saída

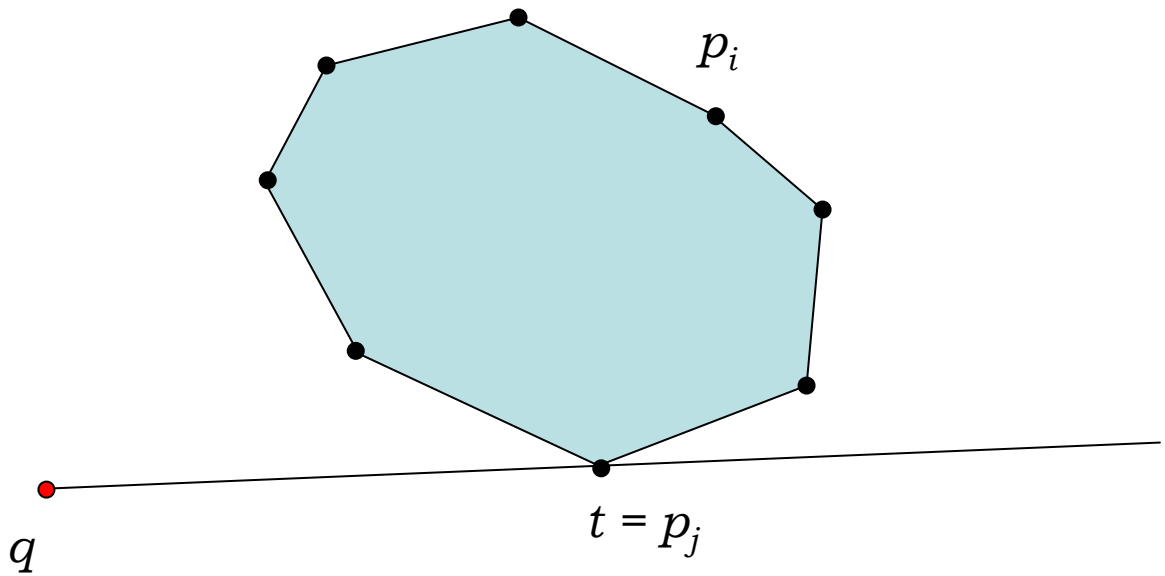
- Um algoritmo ótimo deveria rodar em $O(n \log h)$
  - Comporta-se como marcha de Jarvis para saídas pequenas
  - Comporta-se como a varredura de Graham para saídas grandes (≈ número total de pontos)
- Kirkpatrick e Seidel desenvolveram um algoritmo $O(n \log h)$ em 1986
  - Muito complicado
- Em 1996, Chan apresenta um algoritmo ótimo relativamente simples
  - Combina marcha de Jarvis e varredura de Graham!

Algoritmo de Chan

- Idéia geral
  - Divide-se os n pontos em grupos contendo não mais que m pontos
    - $r = \lceil n/m \rceil$ grupos no total
  - Computa-se o fecho convexo de cada grupo usando varredura de Graham
  - Computa-se o fecho convexo geral aplicando-se a marcha de Jarvis sobre os fechos dos grupos
    - É necessário empregar um algoritmo para obter a tangente entre um ponto e um polígono com m lados em tempo $O(\log m)$

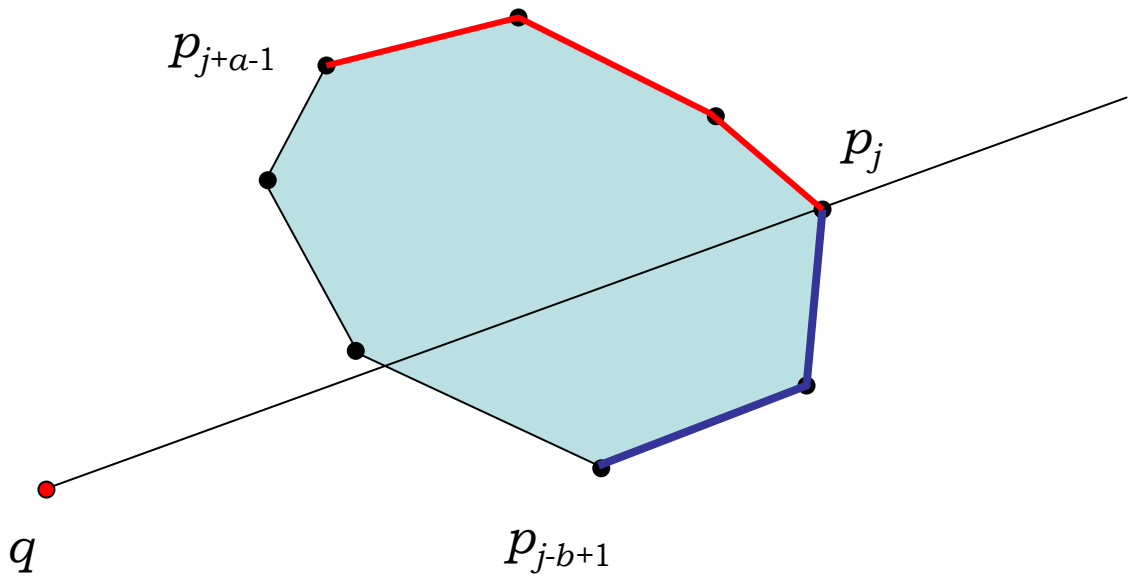
Computando Tangentes

- São dados um polígono convexo com m vértices p_0, p_1 ...p_{m-1} (circulação anti-horária) e um ponto q fora do polígono
- Deseja-se obter uma semi-reta que passa por q e é tangente ao polígono num ponto t de tal forma que qualquer vértice p_i do polígono é tal que *orientação* $(q,t,p_i) \neq$ horária



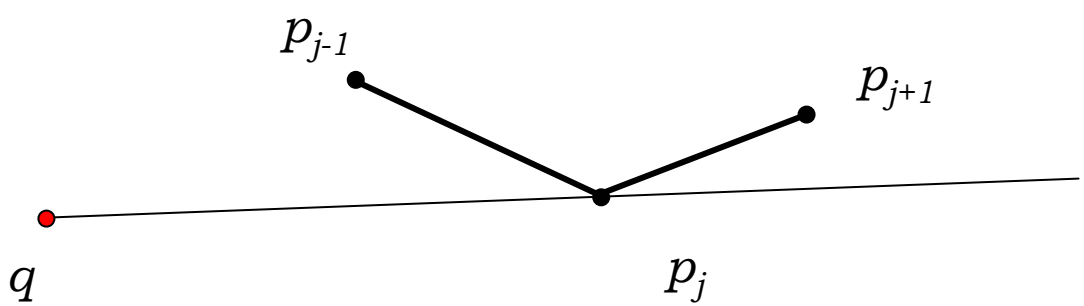
Computando Tangentes

- Assumimos que os vértices p_0, p_1 ...p_{m-1} estão armazenados num array e que os índices são tomados módulo m
- O algoritmo funciona à semelhança do algoritmo de busca binária
- A cada passo examina-se um vértice p_j sabendo que o vértice t procurado está entre p_j e p_{j+a-1} ou entre p_{j-b+1} e p_j ou
  - Inicialmente, $a = b = m$



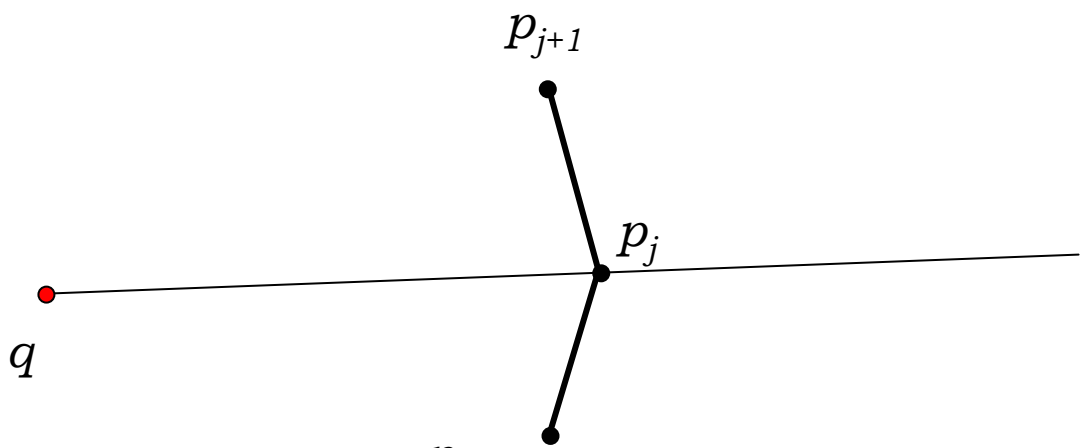
Computando Tangentes

- p_j é classificado examinando seus vizinhos p_{j+1} e p_{j-1}
- 1º caso: *orientação* (q, p_j, p_{j+1}) ≠ horária e *orientação* (q, p_j, p_{j-1}) ≠ horária
  - Algoritmo termina com $t = p_j$



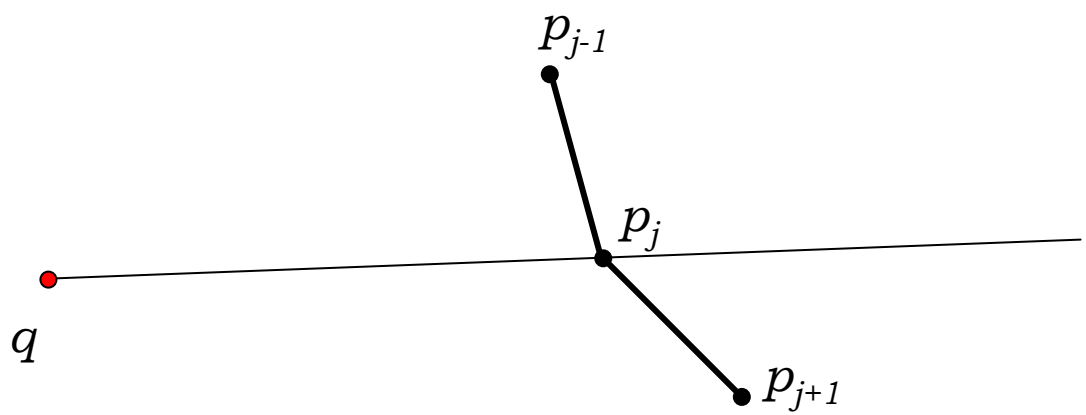
Computando Tangentes

- 2º caso: *orientação* (q, p_j, p_{j+1}) ≠ horária e *orientação* (q, p_j, p_{j-1}) = horária
  - t está entre p_{j-b+1} e p_j



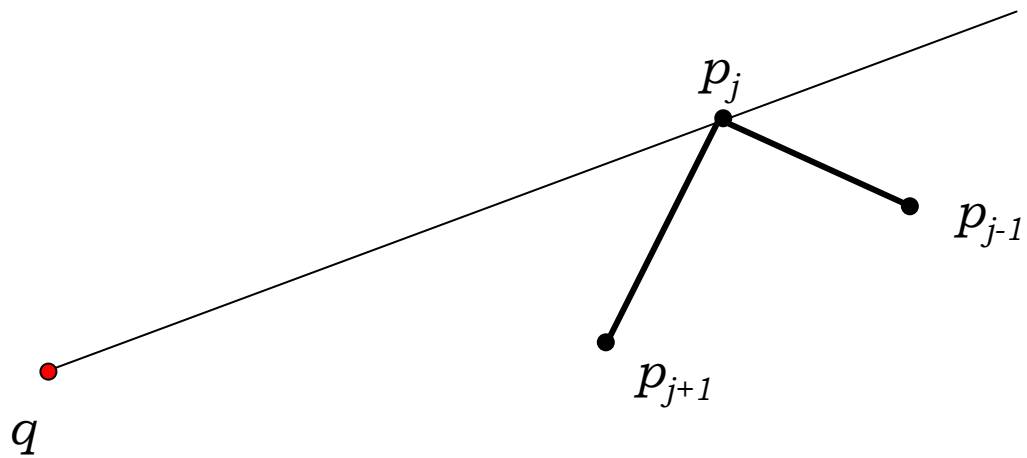
Computando Tangentes

- 3º caso: *orientação* (q, p_j, p_{j+1}) = horária e *orientação* (q, p_j, p_{j-1}) ≠ horária
  - t está entre p_j e p_{j+a-1}



Computando Tangentes

- 4º caso: *orientação* (q, p_j, p_{j+1}) = horária e *orientação* (q, p_j, p_{j-1}) = horária
  - Se *orientação* (q, p_j, p_{j+a-1}) ≠ horária
    - Então t está entre p_j e p_{j+a-1}
    - Senão t está entre p_{j-b+1} e p_j



Computando Tangentes

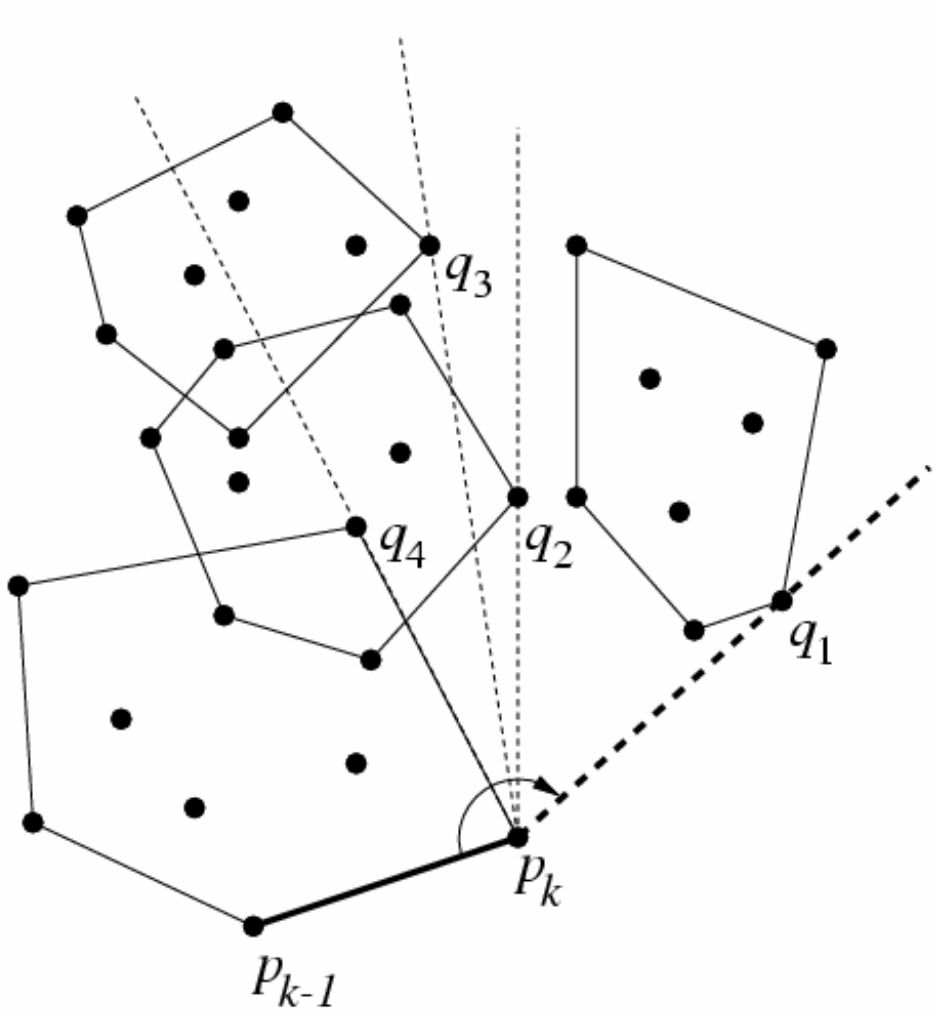
- Sempre que se opta por um intervalo, este é partido em dois semi-intervalos e o vértice do meio é testado a seguir
- Pode-se ver que este processo tem no máximo $\log_2 m$ iterações
- Como cada teste é $O(1)$, o algoritmo tem complexidade de pior caso $O(\log m)$

Algoritmo de Chan (Fecho Parcial)

Dados um conjunto P com n pontos e um valor $m < n$:

1. Divide-se P em $r = \lceil n/m \rceil$ grupos $P_1 \ldots P_r$, cada um contendo não mais que m pontos
2. Para $i = 1$ até r fazer
   a) Computar *Fecho* (P_i) usando Graham
3. Seja $p_0 = (-\infty, 0)$ e p_1 = ponto de P com menor coordenada y
4. Para $k = 1$ até m fazer
   a) Para $i = 1$ até r fazer
      - Computar o ponto q_i de P_i que maximiza o ângulo $\angle\, p_{k-1}p_kq_i$
   b) Fazer p_{k+1} = ponto $q \in \{ q_1 \ldots q_r \}$ que maximiza o ângulo $\angle p_{k-1}p_kq$
   c) Se $p_{k+1} = p_1$ então retornar $\langle p_1 \ldots p_k \rangle$
5. Retornar "m muito pequeno!"

Algoritmo de Chan



Complexidade do Algoritmo de Chan

- No passo 2 computamos r fechos de conjuntos com m pontos: $O(r\, m \log m)$
- No passo 4 temos:
  - Em 4a computamos r tangentes de conjuntos com m pontos: $O(r \log m)$
  - Em 4b computamos uma etapa da marcha de Jarvis, a um custo de $O(r)$
  - Como o passo 4 tem m iterações, o custo total é $O(r\, m \log m)$
- Portanto, o algoritmo tem complexidade $O(r\, m \log m) = O(n \log m)$
- Se pudermos adivinhar um valor de m tal que $m \approx h$, poderemos assegurar complexidade $O(n \log h)$

Adivinhando o valor de *h*

- 1ª idéia: tentar $m = 1, 2, 3$, etc
  - Converge muito lentamente
- 2ª idéia: usar busca binária
  - Converge rápido, mas se usarmos um valor muito alto de m ($n/2$, por exemplo) teremos complexidade $O(n \log n)$
- 3ª idéia: iniciar com m pequeno e incrementar muito rapidamente
  - Dependência de m está no termo log
    - Se chamarmos a rotina com $m = h^c$ para alguma constante c, teremos complexidade $O(n \log h)$
  - Solução: $m = 2^k$, para $k = 2^1, 2^2, 2^3$, etc

Algoritmo de Chan

Para $t = 1, 2, \ldots$ fazer

1. $k \leftarrow 2^t$
2. $m \leftarrow \min(2^k, n)$
3. Chamar Fecho Parcial (P, m)
4. Se resultado $\neq$ "m pequeno demais" retornar

- Quantas iterações?
  - O algoritmo termina quando $t = \lceil \lg \lg h \rceil$
  - Cada iteração leva tempo $O(n \log 2^{2^t}) = O(n\, 2^t)$

$$\sum_{t=1}^{\lg\lg h} n\, 2^t = n \sum_{t=1}^{\lg\lg h} 2^t \leq n\, 2^{1+\lg\lg h} = 2n \lg h \in O(n \log h)$$

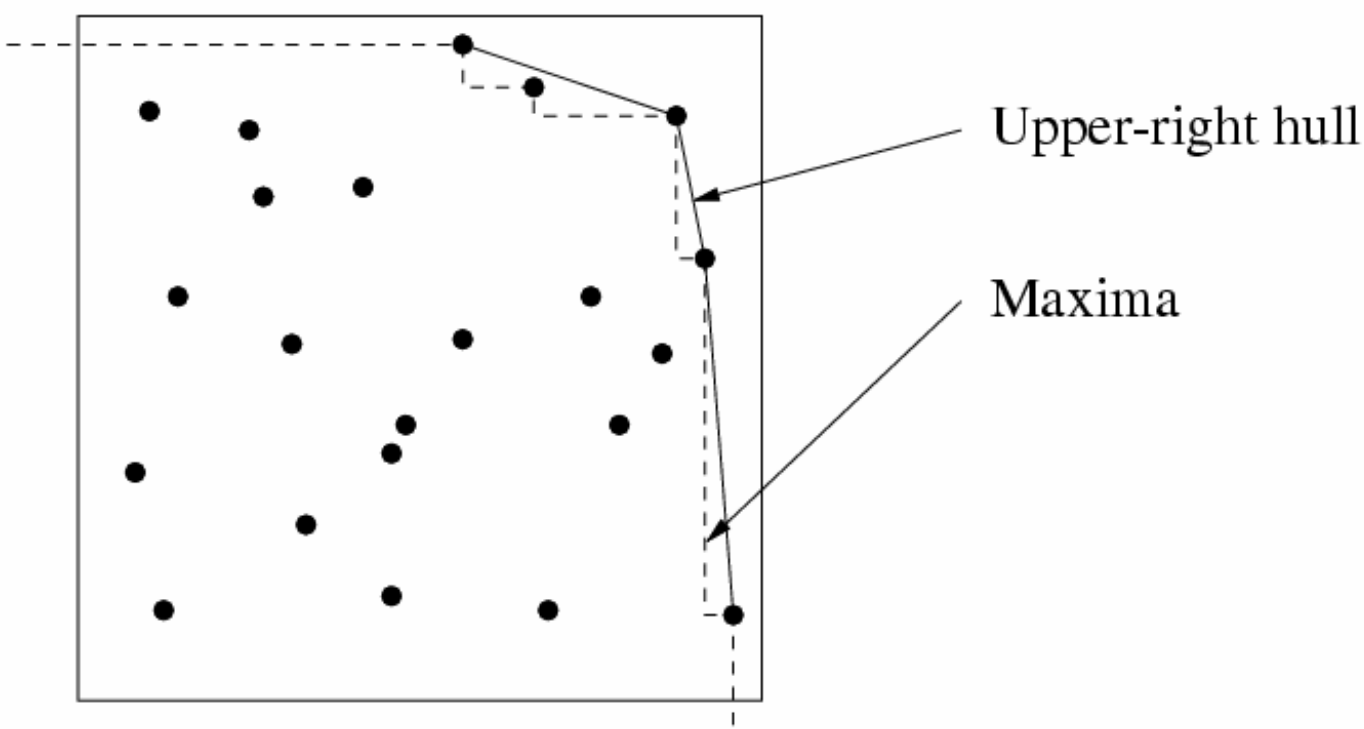
Tempo Ótimo

- Um problema relacionado consiste em verificar se o fecho de um dado conjunto com n pontos tem h vértices
  - Prova-se que este problema é resolvido em $\Omega(n \log h)$
  - O problema do fecho convexo, que garantidamente é mais complexo que este, não pode portanto ser resolvido em menos que $O(n \log h)$

Número esperado de vértices no fecho

- Para pontos distribuídos uniformemente em um quadrado de lado unitário, prova-se que o número esperado de pontos no fecho é $O(\log n)$
- Prova:
  - Todo ponto do fecho é máximo para alguma das 4 orientações do quadrado
    - (Nem todo ponto máximo é do fecho)
  - Prova-se que o número esperado de pontos máximos é $O(\log n)$

Número esperado de vértices no fecho



Número esperado de vértices no fecho

- Ordena-se os pontos em ordem decrescente de x: $p_1, p_2, \ldots p_n$
- Se o ponto p_i é máximo, então sua coordenada y é ≥ que a coordenada y de $p_1, p_2, \ldots p_i$
  - Como a distribuição é uniforme, a chance de isso acontecer é $1/i$
  - Temos então como a soma das expectativas:

$$E_n = \sum_{i=1}^{n} \frac{1}{i} \cong \ln n = O(\log n)$$