Universidade Federal de Uberlândia - UFU
Programa de Pós Graduação em Ciência da Computação
Inteligência Artificial

Fecho Convexo Bidimensional

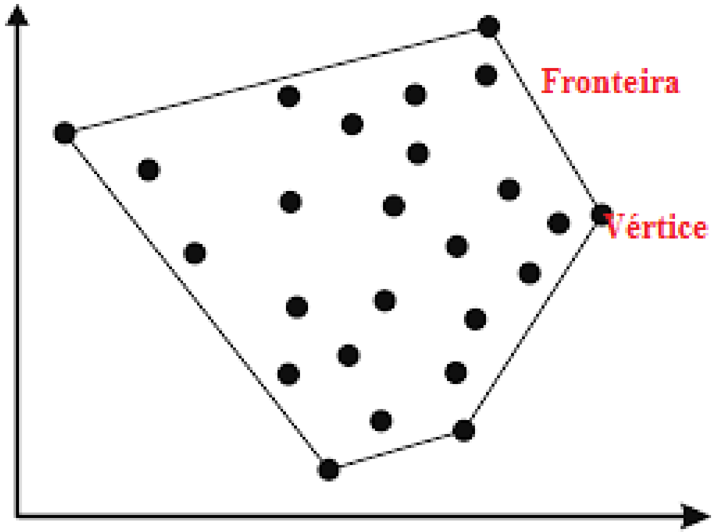
CIBELE MARA FONSECA
MALENA ALVES RUFINO
DISCIPLINA: ANÁLISE DE ALGORITMOS
PROFESSORA DRA: MÁRCIA APARECIDA FERNANDES

Fecho Convexo Bidimensional

Dados pontos no plano, encontrar o fecho convexo desses pontos.

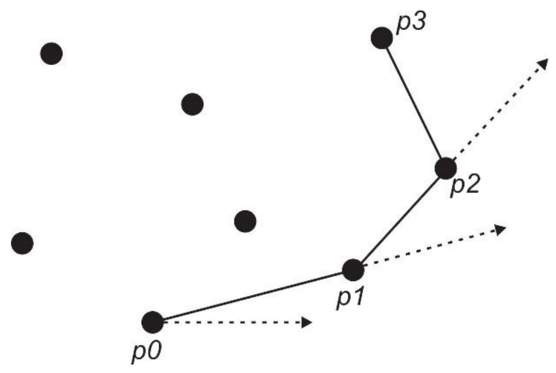
Seja fecho convexo do conjunto ***S***, temos o problema:

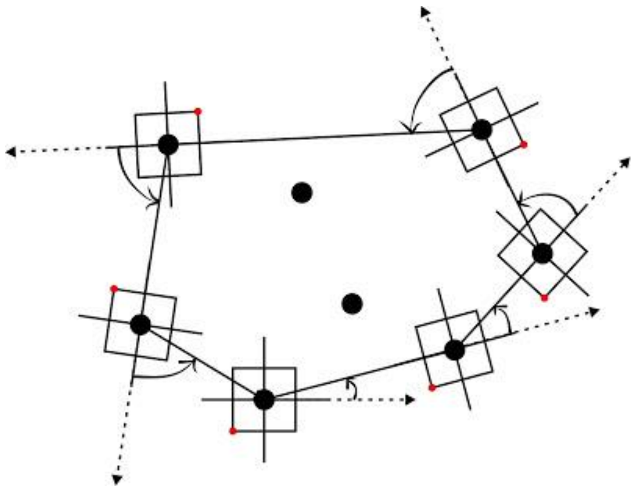
1. Encontra as fronteiras do polígono ***S***.
2. Encontrar os vertices do polígonos ***S***.



Jarvis March (Embrulho de presente)

A partir de um ponto extremo do fecho convexo, encontrar o próximo ponto extremo no sentido anti-horário. A cada passo, escolhe o menor dos valores e acrescenta à coleção ordenada.





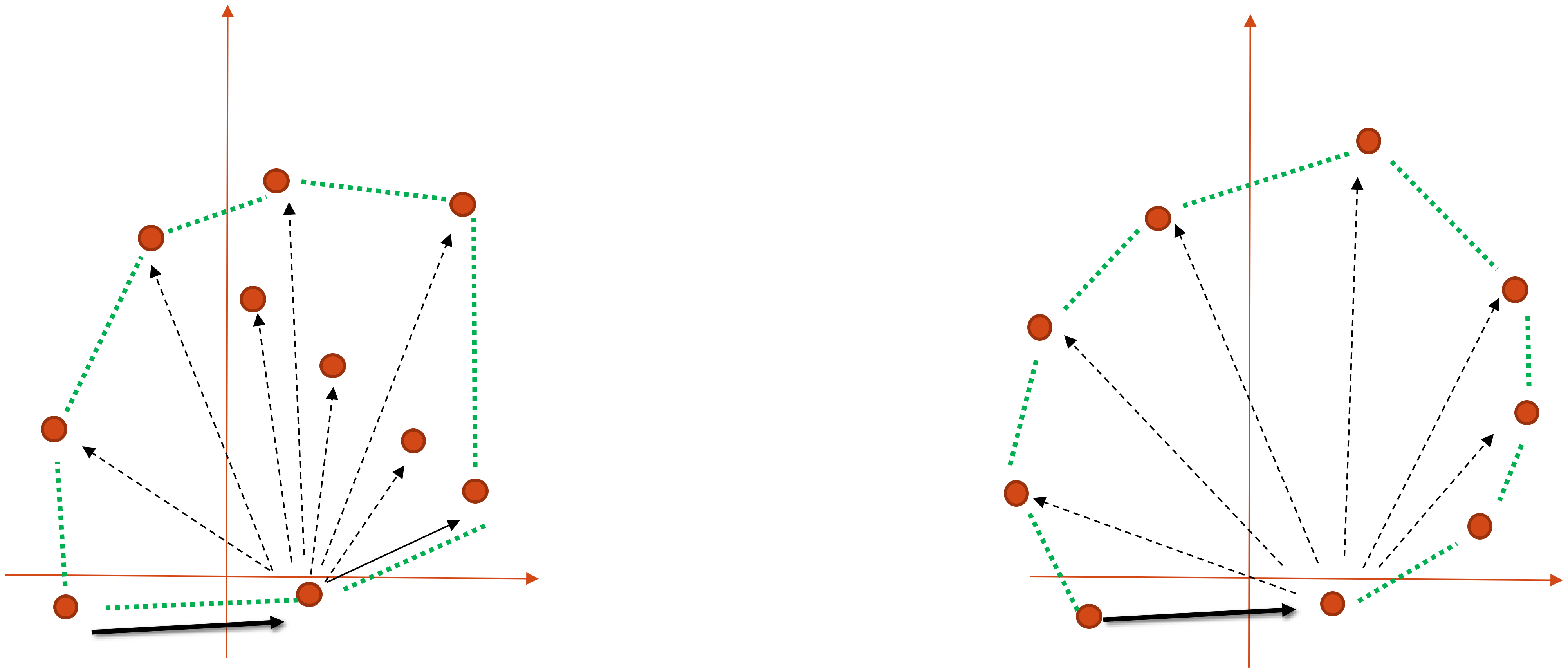
Jarvis March (Embrulho de presente)

```
Embrulho(X, Y, n)
1.  h ← 0
2.  H[0] ← min{i ∈ [1 . . n] : X[i] ≤ X[j], 1 ≤ j ≤ n}
3.  Repita
4.      i ← (H[h] mod h) + 1 //qualquer ponto distinto de H[h]
5.      para j ← 1 até n faça
6.          se Dir(X, Y, H[h], i, j) então i ← j
7.      h ← h + 1
8.      H[h] ← i
9.  até que i = H[0] // fechou o polígono
10. devolva (H, h)
```

Nas linhas 4-6, o algoritmo determina o sucessor do ponto de índice H[h] na fronteira do fecho convexo no sentindo anti-horário. Quando este sucessor é o ponto de índice H[0], significa que H[1 . . h] está completo.

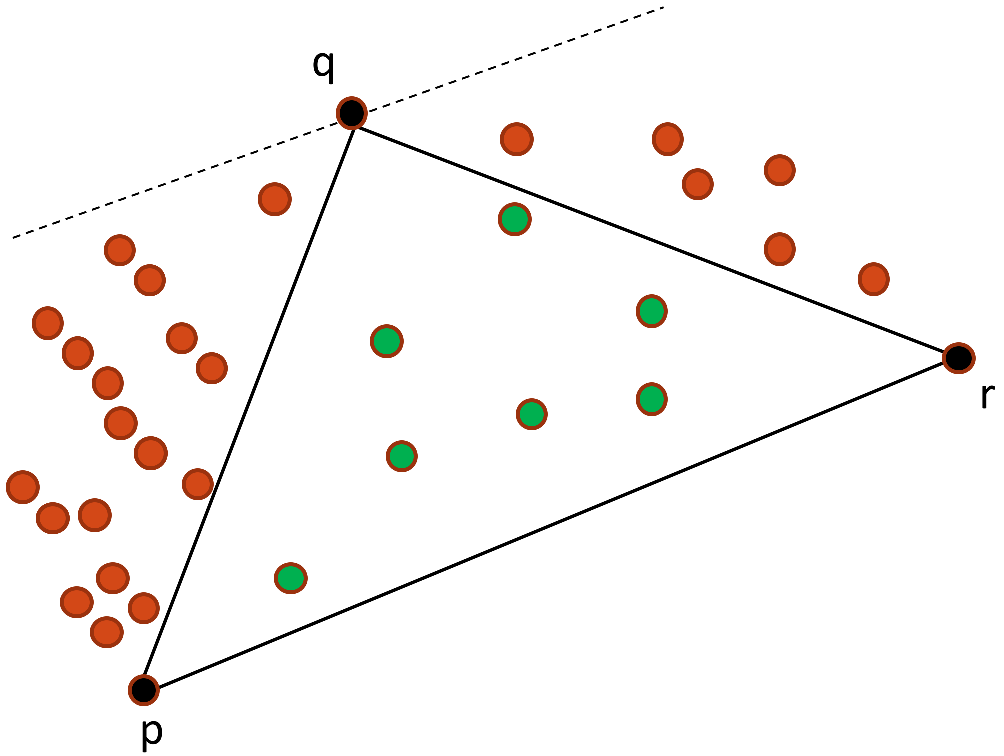
Se h é o número de pontos extremos do fecho convexo dos pontos X[1 . . n], Y [1 . . n], então o bloco de linhas 4-8 é executado h vezes. Para cada uma dessas h execuções, a linha 6 é executada n vezes. Portanto, o consumo de tempo do algoritmo é **Θ(hn).**

Exemplo caso médio $\Theta(hn)$ e pior caso $O(n^2)$



QuickHull

Possui estrutura semelhante ao algoritmo de ordenação QuickSort. Este algoritmo consiste em encontrar os pontos mais à esquerda e direita. Dada a reta traçada por esses dois pontos, faz a divisão do conjunto de modo recursivo.



QuickHull

Dado os conjuntos com coleções X[1 . . n], Y [1 . . n].

Encontrar os dois pontos estremos e em seguida encontrar o terceiro ponto, o mais distante da coleção em relação à reta que passa pelos dois extremos iniciais. Para tal, é necessario calcular área do triangulo.

Área *(X, Y, i, j, k)*

1. ***devolva |DET(X[i], Y [i], X[j], Y [j], X[k], Y [k])| / 2***

Na DET(x1, y1, x2, y2, x3, y3) o valor absoluto é duas vezes a área do triângulo de pontas (x1, y1), (x2, y2) e (x3, y3). Devolve a área do triângulo cujas pontas são os pontos da coleção de índices i, j e k.

QuickHull

O algoritmo abaixo recebe uma coleção de pelo menos três pontos X[p . . r], Y [p . . r] em posição geral e, usando Área, devolve o índice de um ponto extremo da coleção distinto de p e r. Este algoritmo será usado no algoritmo Particione.

PontoExtremo (X, Y, p, r)

1. ***q ← p + 1***
2. ***max ← Área(X, Y, p, r, q)***
3. ***para i ← p + 2 até r − 1 faça***
4. ***se Área(X, Y, p, r, i) > max***
5. ***então q ← i***
6. ***max ← Área(X, Y, p, r, q)***
7. ***devolva q***

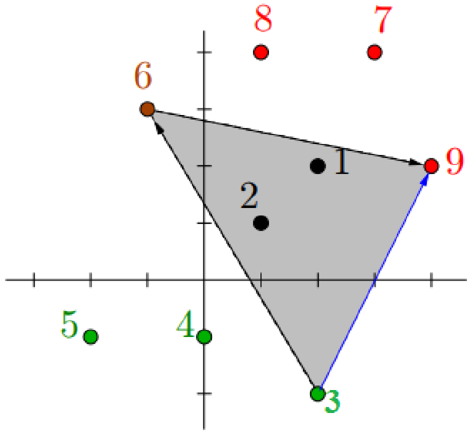
Consome tempo $\Theta(n)$, onde $n := r - p + 1$.

PARTICIONE (X, Y, p, r)

1. q ← PONTOEXTREMO(X, Y, p, r)
2. (X[p+1], Y [p+1]) ↔ (X[q], Y [q])
3. p′ ← r q ← r
4. para k ← r − 1 decrescendo até p + 2 faça
5.   se ESQ(X, Y, p, p+1, k)
6.   então p′ ← p′ − 1 (X[p′], Y [p′]) ↔ (X[k], Y [k])
7.   senão se ESQ(X, Y, p+1, r, k)
8.   então p′ ← p′ − 1 q ← q − 1
9.   (X[q], Y [q]) ↔ (X[k], Y [k])
10.   se p′ ≠ q então (X[k], Y [k]) ↔ (X[p′], Y [p′])
11. p′ ← p′ − 1 q ← q − 1
12. (X[q], Y [q]) ↔ (X[p+1], Y [p+1])
13. se p′ ≠ q então (X[p′], Y [p′]) ↔ (X[p+1], Y [p+1])
14. p′ ← p′ − 1 (X[p′], Y [p′]) ↔ (X[p], Y [p])
15. devolva (p′, q)

	p	$p{+}1$						k	p' q r
X	2	−1	3	0	1	1	−2	2	4
Y	−2	3	4	−1	4	1	−1	2	2

	p		p'			q			r
X	2	1	2	0	−2	−1	3	1	4
Y	2	1	−2	−1	−1	3	4	4	2
	1	2	3	4	5	6	7	8	9



PARTICIONE (X, Y, p, r)

```
1.  q ← PONTOEXTREMO(X, Y, p, r)
2.  (X[p+1], Y [p+1]) ↔ (X[q], Y [q])
3.  p' ← r q ← r
4.  para k ← r – 1 decrescendo até p + 2 faça
5.      se ESQ(X, Y, p, p+1, k)
6.          então p' ← p' – 1 (X[p'], Y [p']) ↔ (X[k], Y [k])
7.          senão se ESQ(X, Y, p+1, r, k)
8.              então p' ← p' – 1 q ← q – 1
9.                  (X[q], Y [q]) ↔ (X[k], Y [k])
10.                 se p' ≠ q então (X[k], Y [k]) ↔ (X[p'], Y [p'])
11. p' ← p' – 1 q ← q – 1
12. (X[q], Y [q]) ↔ (X[p+1], Y [p+1])
13. se p' ≠ q então (X[p'], Y [p']) ↔ (X[p+1], Y [p+1])
14. p' ← p' – 1 (X[p'], Y [p']) ↔ (X[p], Y [p])
15. devolva (p', q)
```

$\Theta(n)$

$\Theta(1)$

O bloco de linhas 5-10 é executado n – 3 vezes e cada execução consome tempo $\Theta(1)$. Logo o consumo de tempo de Particione é $\Theta(n)$.

$\Theta(1)$

QuickHull

```
QUICKHULL (X, Y, n)
se n = 1
então h ← 1 H[1] ← 1
senão k ← min{i ∈ [1 . . n] : Y [i] ≤ Y [j], 1 ≤ j ≤ n}            Θ(1)
      (X[1], Y [1]) ↔ (X[k], Y [k])
      i ← 2
      para j ← 3 até n faça
          se DIR(X, Y, 1, i, j) então i ← j                        Θ(n)
          (X[n], Y [n]) ↔ (X[i], Y [i])
          (H, h) ← QUICKHULLREC(X, Y, 1, n)                        Θ(n log n)
devolva (H, h)
```

QuickHull

QuickHullRec (X, Y, p, r)

```
se p = r – 1 // há exatamente dois pontos na coleção
      então h ← 2 H[1] ← r H[2] ← p
senão (p ' , q) ← Particione(X, Y, p, r)          → Θ(n)
       (H1, h1) ← QuickHullRec(X, Y, q, r, H, h)  → T(n1)
      (H2, h2) ← QuickHullRec(X, Y, p' , q, H, h) → T(n2)
       // remove uma cópia do q
       remove(H2[0])
       h ← h1+h2
       H ← H1+H2
 devolva (H, h)
```

QuickHull

QuickHullRec (X, Y, p, r)

```
se p = r – 1 // há exatamente dois pontos na coleção
    então h ← 2 H[1] ← r H[2] ← p
senão (p ' , q) ← Particione(X, Y, p, r)
    (H1, h1) ← QuickHullRec(X, Y, q, r, H, h)
    (H2, h2) ← QuickHullRec(X, Y, p', q, H, h)
    // remove uma cópia do q
    remove(H2[0])
    h ← h1+h2
    H ← H1+H2
devolva (H, h)
```

$$\mathbf{T(n) = T(n_1) + T(n_2) + \Theta(n)}$$

Se a partição é balanceada então **$T(n_1) = n/2$ e $T(n_2) = n/2$.** . Assim, podemos descrever o algoritmo por meio da recorrência:

$$T(n) = \begin{cases} T(1) = 1 \\ T(n) = 2\,T(n/2) + n \end{cases}$$

Utilizando-se o teorema master, pelo caso 2, a solução da recorrência resulta no caso médio $\Theta(n \log n)$.

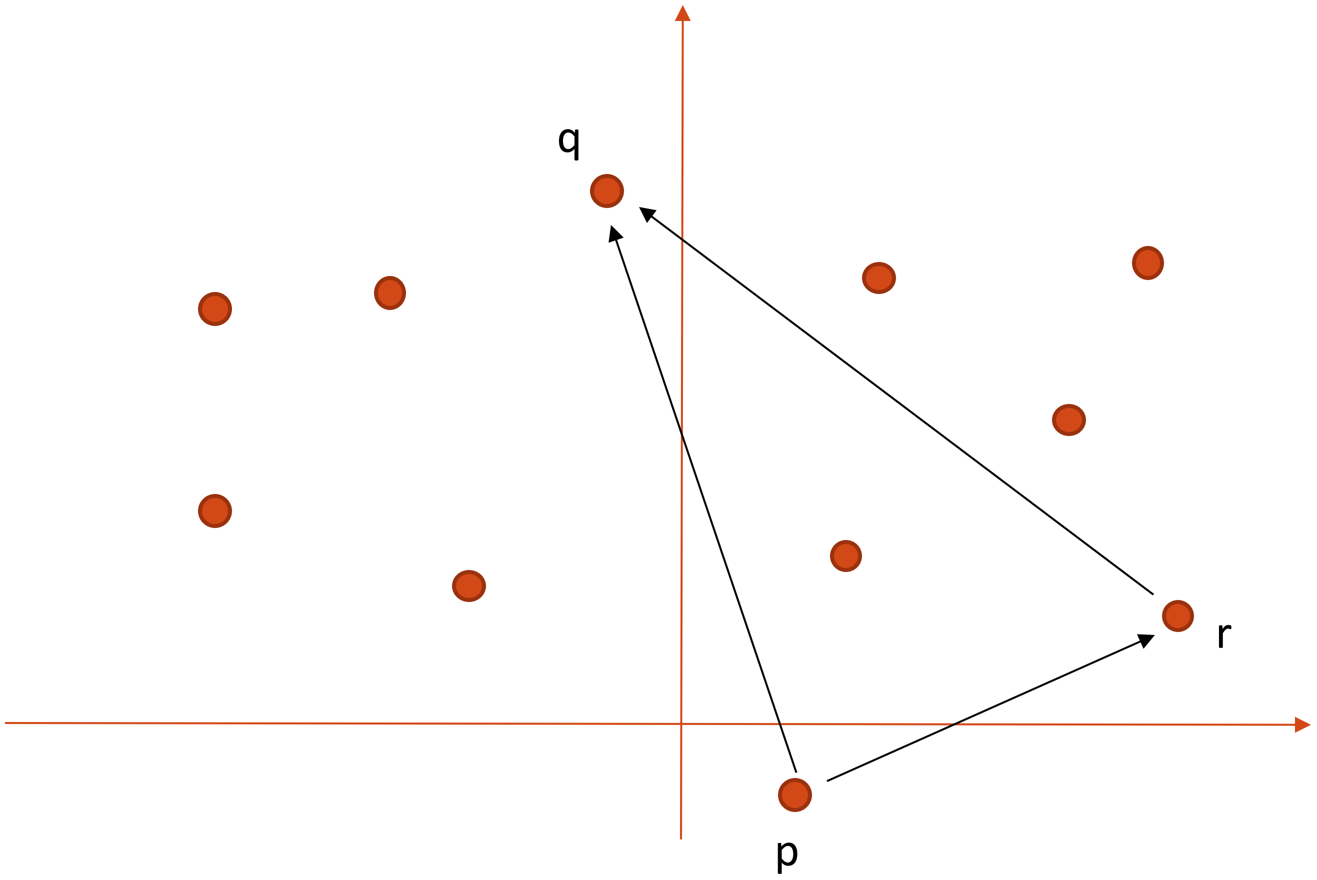
Etapas do QuickHull

Dividir: ocorre quando o método *particione(X, Y, p, r)* é chamado na linha 3.

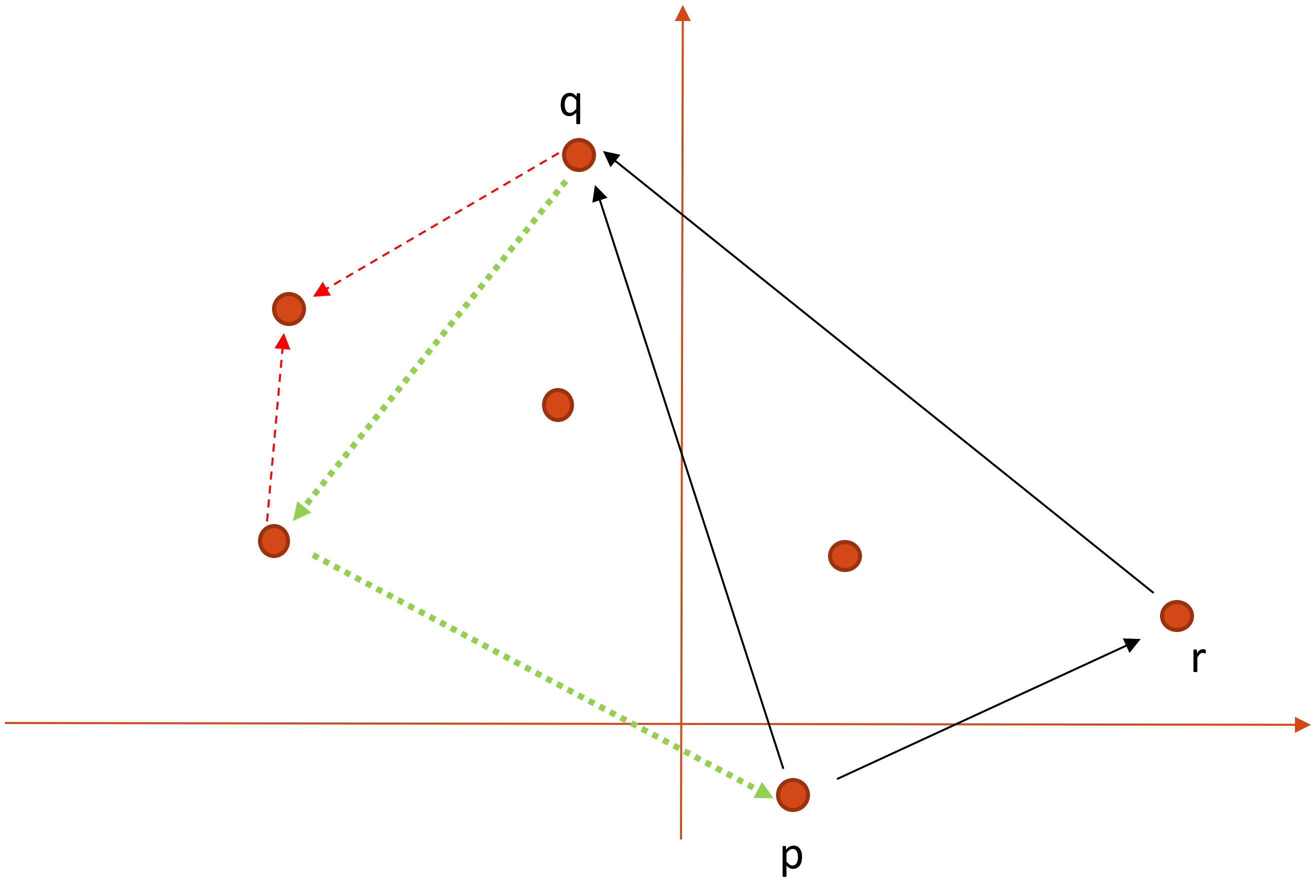
Conquistar: ocorre nas chamadas recursivas do método *quickHullRec(X, Y, p, r)* nas linhas 4 e 5.

Combinar: ocorre na linha 9 quando *H1* e *H2* são concatenados.

Exemplo caso médio Θ(n log n)



Exemplo pior caso $O(n^2)$



$T(n) = T(n-1) + n$

$$
\begin{array}{ccccc}
 & & & n & \\
 & & n-1 & & 1 \\
 & n-2 & & 1 & \\
1 & & 1 & &
\end{array}
$$

Exemplo pior caso $O(n^2)$

Método de Expansão

$T(n) = T(n-1) + n$

$= T(n-2) + n-1 + n$

$= T(n-3) + n-2 + n-1 + n$

// generalizar para k

$= T(n-k) + n-k-1 + n-k-2 + ... + n-1 + n$

// já que $n-k = 1$ então $k = n-1$ e $T(1) = 1$

$= 1 + 2 + ... + n$

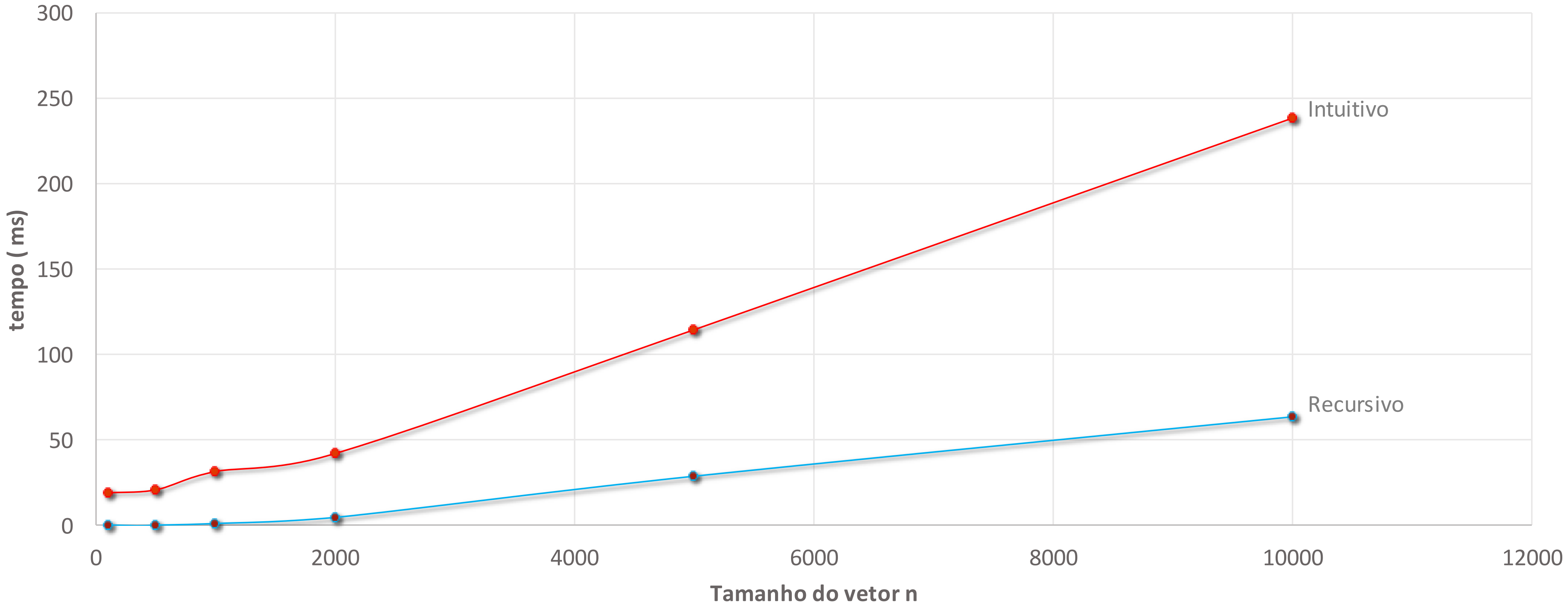
$= [n(n+1)]/2$

$= n^2/2 + n/2$

// tomamos o termo dominante que é n^2

$= O(n^2)$

Análise de curva de tempo (ms)



Dúvidas?