Par de pontos mais próximos

Algoritmos geométricos

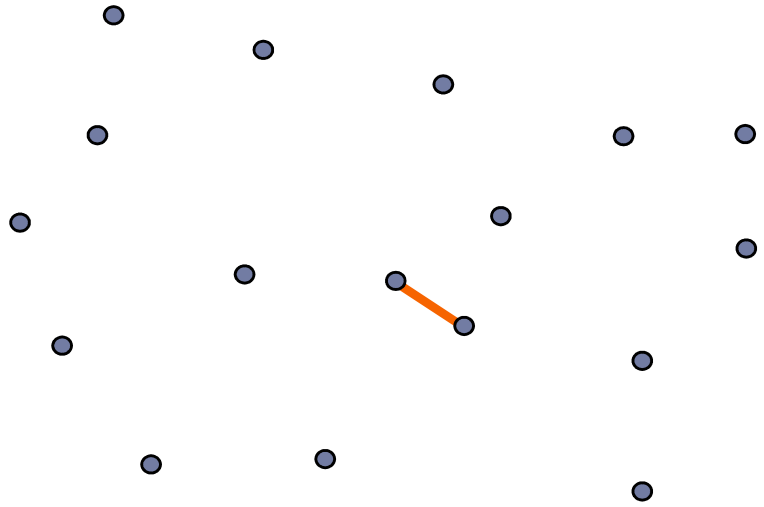
Prof. Aurélio Hoppe

aurelio.hoppe@gmail.com

PAR DE PONTOS MAIS PRÓXIMOS

- **Enunciado:** Dados n pontos queremos encontrar dois cuja distância entre eles é mínima.

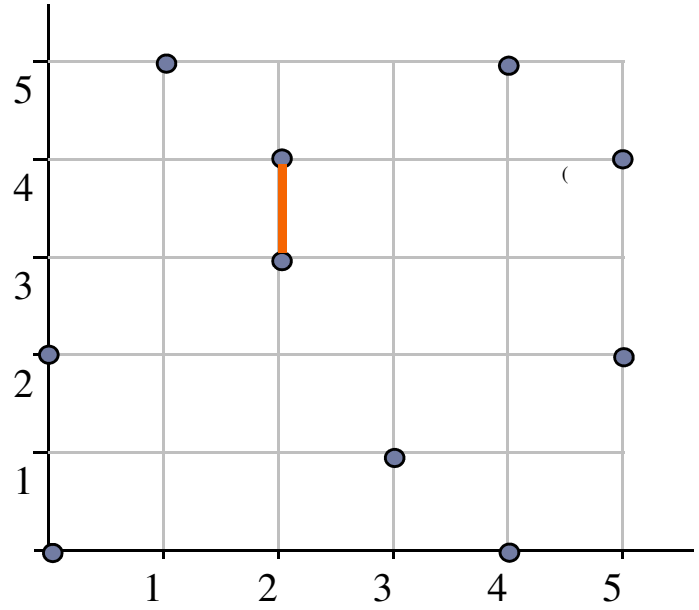
- **Aplicação:** uma aplicação prática deste problema está no controle de tráfego aéreos: os dois aviões que estão em maior perigo de colisão são aqueles que estão mais próximos.



PAR DE PONTOS MAIS PRÓXIMOS

- **Enunciado:** Dados n pontos queremos encontrar dois cuja distância entre eles é mínima.

- **Entrada:** (4,5),(3,1), (2,3), (0,2), (5,4), (5,2), (1,5), (2,4), (4,0), (0,0)

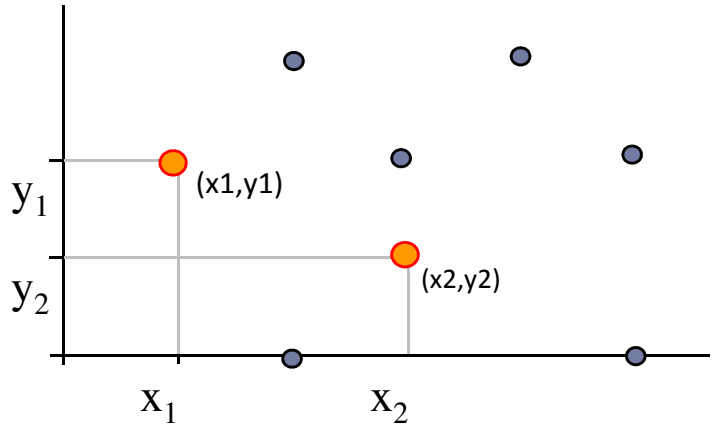


- **Saída:** (2,3) e (2,4) que estão a 1 de distância

FURB

PAR DE PONTOS MAIS PRÓXIMOS

- **Enunciado:** Dados n pontos queremos encontrar dois cuja distância entre eles é mínima.



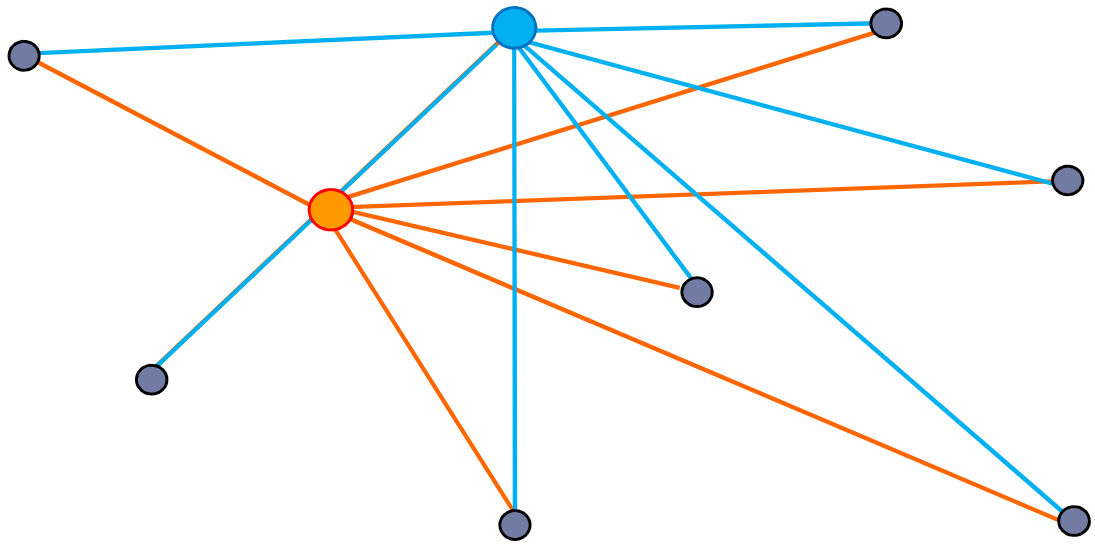
- **E agora? Como descobrir a distância entre dois pontos?**

Distância euclidiana: $$d(p,q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$$

FURB

PAR DE PONTOS MAIS PRÓXIMOS
[Algoritmo ingênuo]

- A **primeira ideia** de solução para este problema é:
  - Aplicar uma busca exaustiva em todos os pares de pontos da malha dada à procura da distância mínima.



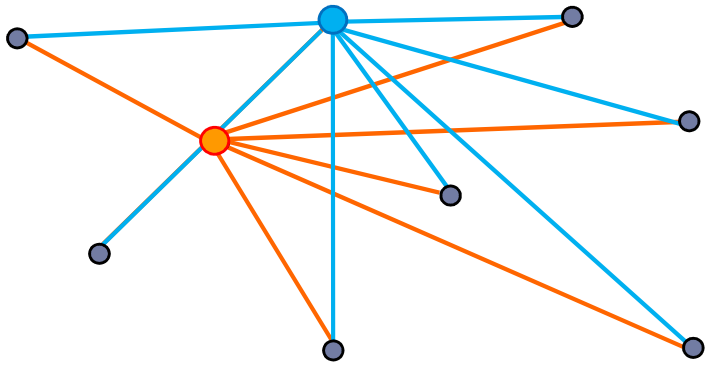
FURB

PAR DE PONTOS MAIS PRÓXIMOS
[Algoritmo ingênuo]

- Dada uma coleção $P = (p_1, p_2, \ldots, p_n)$ de n pontos determina a distância mínima entre pontos de P.

Algoritmo ingênuo ou força-bruta (PmP1)

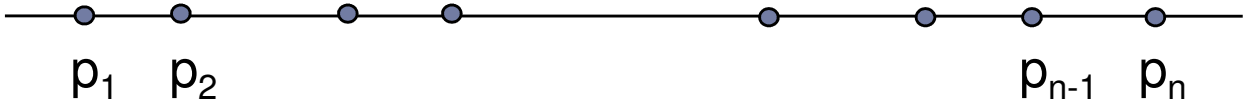
```
01.   D ⇦ ∞
02.   Para todo i em [1..n-1] faça
03.       Para todo j em [i+1..n] faça
04.           D ⇦ min{D, d(p_i,q_j)}
05.   Devolve D
```



- **É a melhor solução?** **Não!!!** ☹

PAR DE PONTOS MAIS PRÓXIMOS
[Procurando melhores ideias]

- **Primeira ideia**: Tentar fazer incursões em dimensões menores de modo a obter ideias que sugiram soluções eficientes para dimensões maiores (R^1)
  - Ordenar os pontos e procurar o par mais próximo através de um varrimento linear destes pontos
  - Nesta estratégia o par de pontos mais próximos é necessariamente **um par consecutivo** no conjunto ordenado é o que permite desenvolver este algoritmo.



PAR DE PONTOS MAIS PRÓXIMOS
[Procurando melhores ideias]

- Dada uma coleção $P = (p_1, p_2, \ldots, p_n)$ de n pontos em R^1 e devolve um par mais próximo {p,q} em P.

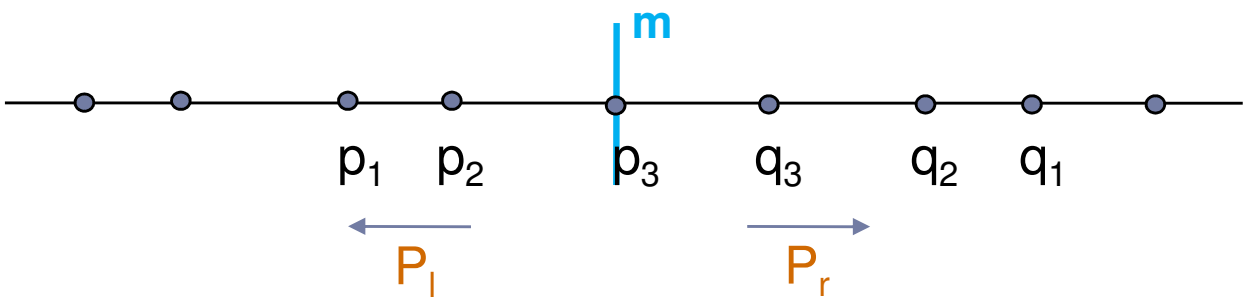
Algoritmo ingênuo2 (PmP2)

```
01.   Ordene p1,..,pn de tal forma que p1 ≤ .. ≤ pn
02.   D ⇦ ∞
03.   Para todo i em [1..n-1] faça
04.        Se distância(pi, pi + 1) < D então
05.             D ⇦ distância(pi, pi + 1)}
06.             p ⇦ pi;
07.             q ⇦ pi+1;
08.   Devolve {p,q}
```

É possível generalizar esta ideia para outras dimensões??? **Não!!!** ☹

PAR DE PONTOS MAIS PRÓXIMOS
[Procurando melhores ideias]

- **Segunda ideia**: Versão divisão-e-conquista (R^1)
  - Dividimos um conjunto P de n pontos em R^1 pela sua mediana*:
    - Temos dois subconjuntos disjuntos: P_l e P_r, (todos os pontos de P_l estão à esquerda da mediana e os de P_r à direita)
  - O par de ponto mais próximo em P será:
    - Um par contido em P_l ou
    - Um par contido em P_r ou
    - Um par muito particular de pontos de P_l x P_r (ponto mais a direta de P_l e do mais a esquerda de Pr, isto é, os dois pontos vizinhos da mediana).

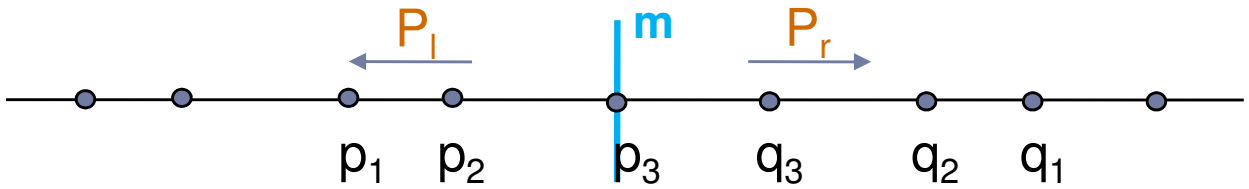


* a mediana será o elemento central (n+1)/2

FURB

PAR DE PONTOS MAIS PRÓXIMOS

[Divisão-e-conquista em R^1]



- **Dividir**
  - Compute a mediana m de P
  - Particione P em P_l e P_r tal que $p \leq m < q \mid p \in P_l$ e $q \in P_r$
- **Conquistar**
  - Resolva, separadamente, o Problema do Par-mai-proximo(Pmp) para P_l e P_r
    - Sejam {p1,p2} o PmP em P_l e {q1,q2} o PmP em P_r
- **Combinar**
  - Seja $D = \min\{|p_1-p_2|, |q_1,q_2|\}$, O PmP de P é
    - $\{p_1,p_2\}$ ou $\{q_1,q_2\}$ ou $\{p_3,q_3\}$, onde $p_3 \in P_l$ e $q_3 \in P_r$
  - Se PmP de P é $\{p_3,q_3\}$, então $|p_3-q_3| < D$
  - **Quantos pares de pontos são candidatos a $\{p_3,q_3\}$?**
    - **Resposta:** No máximo 1 ⇨ $p_3=m$ e $q_3=\min\{q \in P_r\}$

FURB

PAR DE PONTOS MAIS PRÓXIMOS
[Divisão-e-conquista em R^1]

- Dada uma coleção $P = (p_1, p_2, \ldots, p_n)$ de n pontos em R^1, devolve um par mais próximo $\{p_i, q_j\}$, $i \neq j$.

Algoritmo divisao-e-conquista R^1 (PmP3)

```
01.  Se n = 1 então devolve D ⇦ ∞
02.  senão Se n = 2 então devolve distância(p1,p2)
03.       senão
04.            m ⇦ mediana(P)
05.            Sejam P1 =(p∈P|p≤m) e Pr =(q∈P|q>m)
06.            D1 ⇦ PmP3(P1)
07.            Dr ⇦ PmP3(Pr)
08.            p3 ⇦ max(p∈P1)
09.            q3 ⇦ min(q∈Pr)
10.            D ⇦ min(D1, Dr, q3−p3)
11.  Devolve D
```

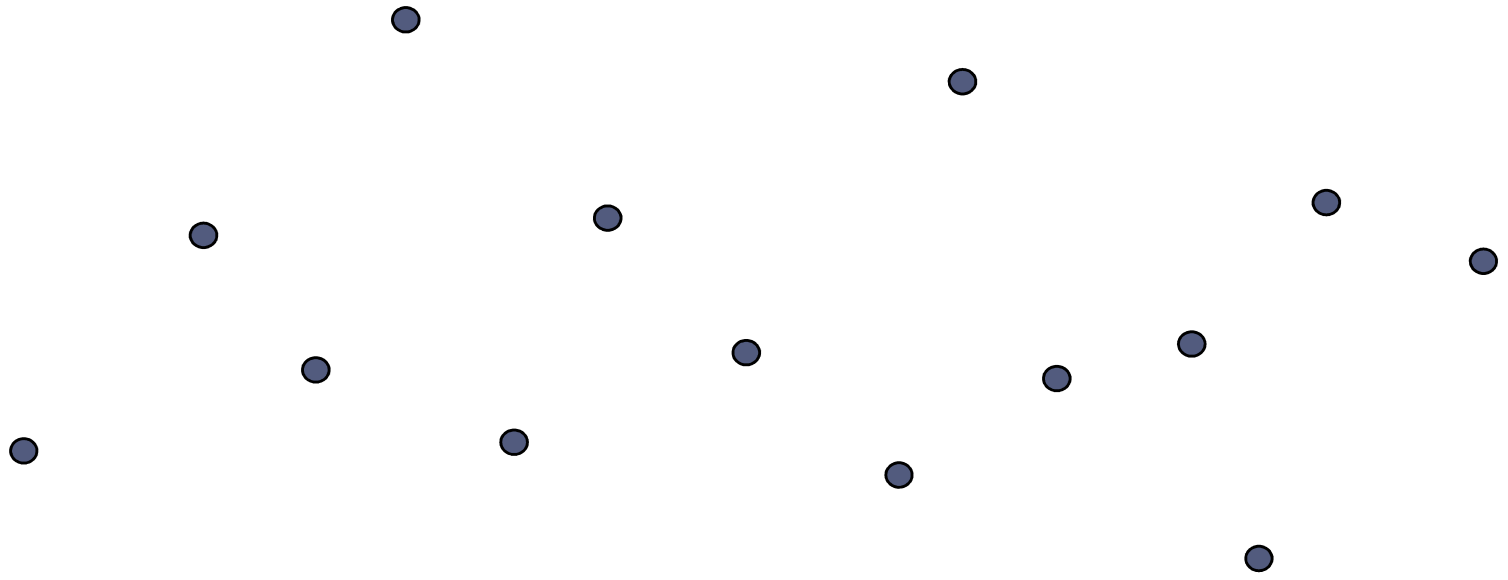
PAR DE PONTOS MAIS PRÓXIMOS

É possível aplicar o algoritmo divisão-e-conquista para o problema PmP em R^2?

Sim!!! ☺

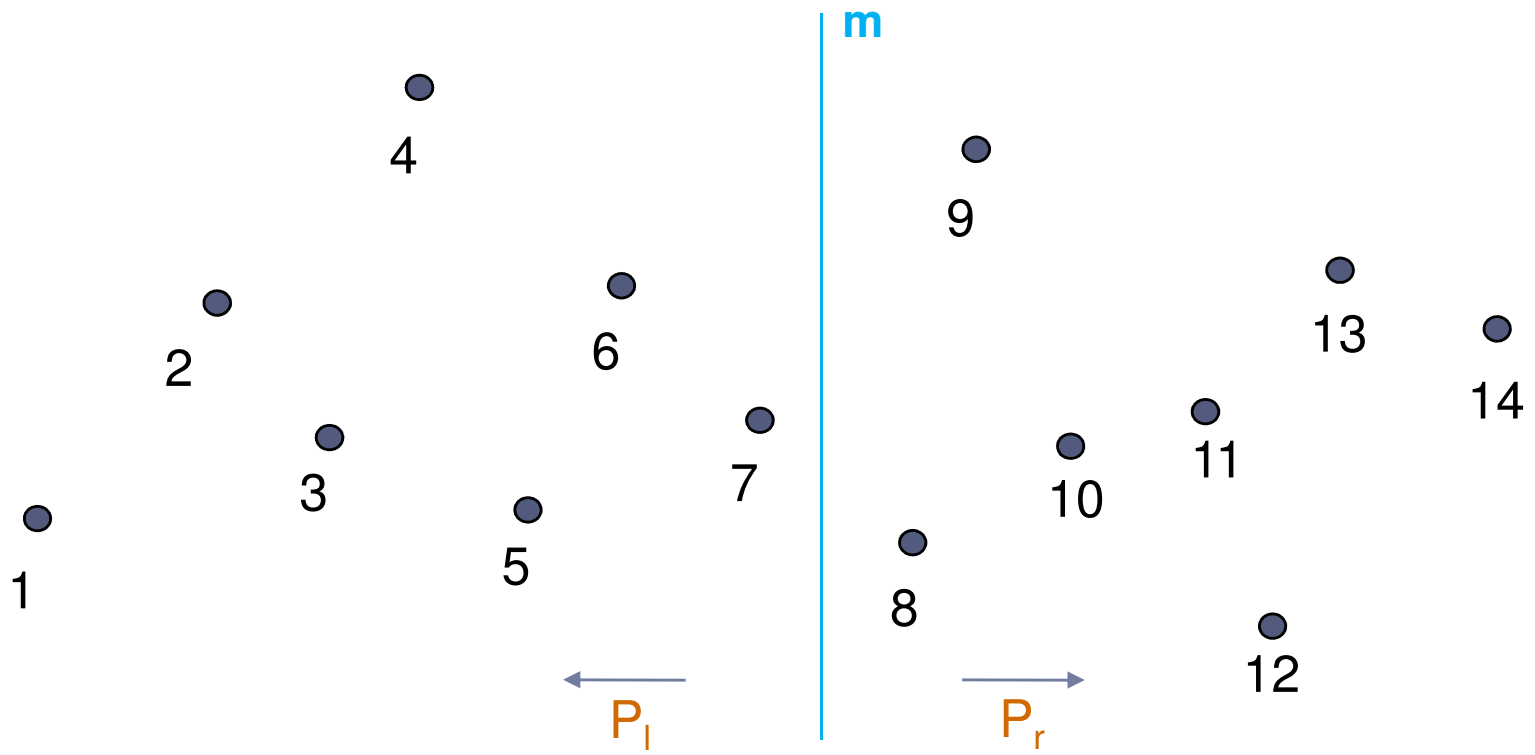
PAR DE PONTOS MAIS PRÓXIMOS
[Divisão-e-conquista em R^2]

- Seja $P=\{p_1, p_2,...,p_n\}$ o conjunto dos n pontos em R^2
  - Se o número de pontos for menor que três, então o problema é resolvido através do algoritmo de força-bruta
  - Caso contrário em cada nível de recursividade executar as fases Dividir, Conquistar e Combinar, como passamos a descreve a continuação



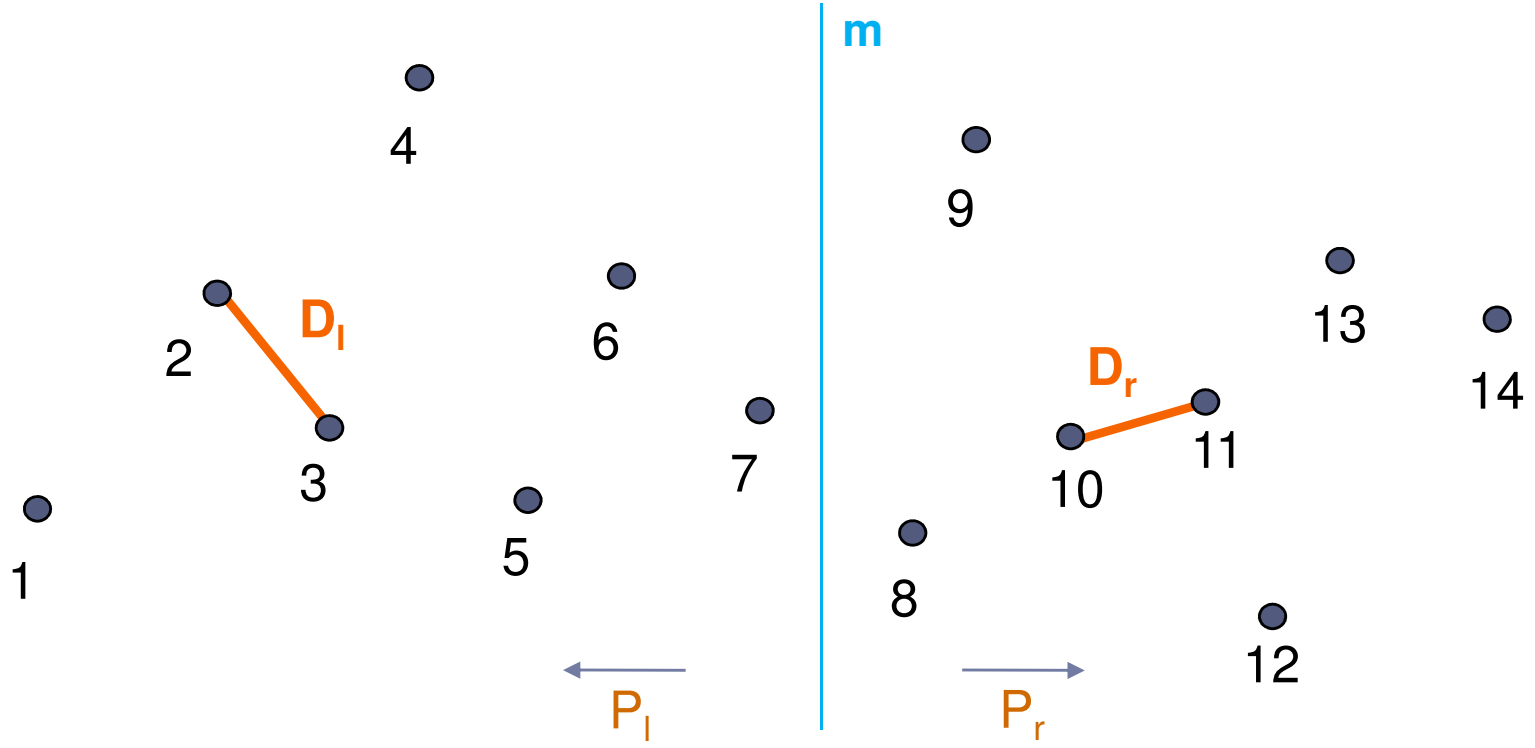
PAR DE PONTOS MAIS PRÓXIMOS
[Divisão-e-conquista em R^2]

- **Dividir**
  - Particione P em P_l e P_r (mediada "m")
  - Em **pre-processamento ordene** os pontos de P segundo as suas x-coordenadas



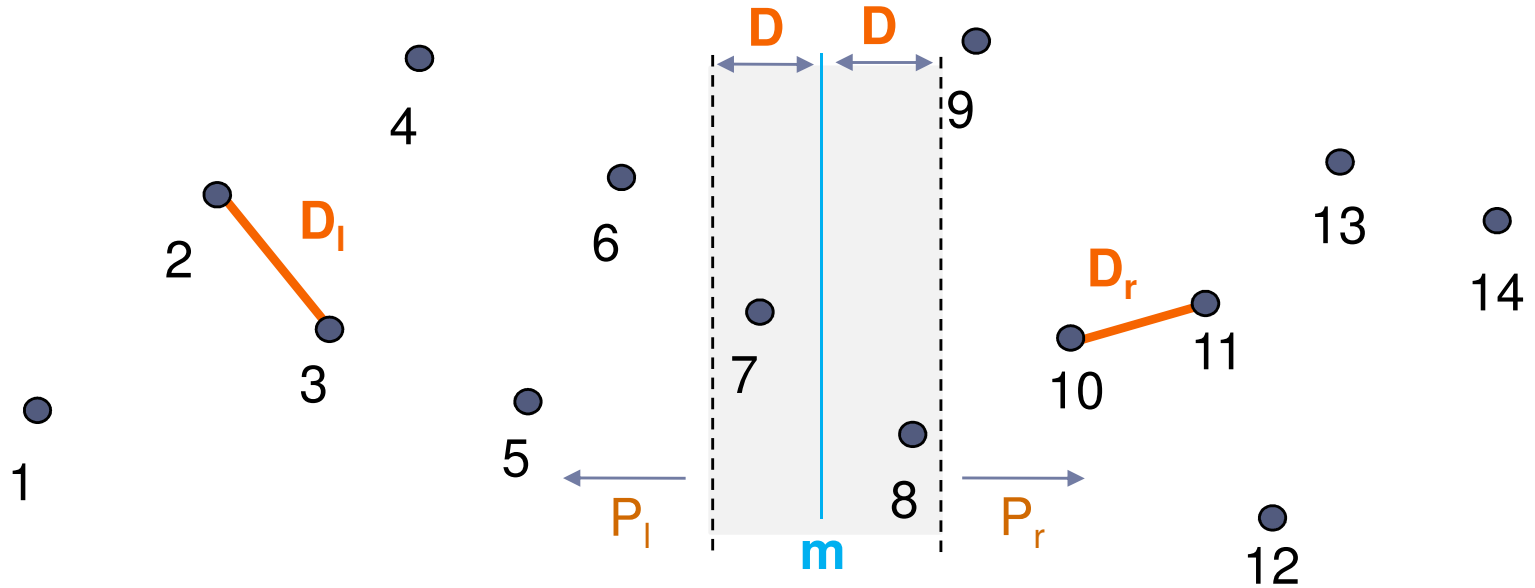
PAR DE PONTOS MAIS PRÓXIMOS
[Divisão-e-conquista em R^2]

- **Conquistar**
  - O problema deve ser resolvido recursivamente para P_l e P_r , obtendo assim D_l e D_r, as distâncias mínimas entre pares de pontos $\{p_1,p_2\}$ em P_l e $\{q_1,q_2\}$ em P_r respectivamente.



PAR DE PONTOS MAIS PRÓXIMOS
[Divisão-e-conquista em R^2]

- **Combinar**
  - Seja D = min(D_l , D_r) e seja m uma reta tal que todo ponto de P_l está à esquerda ou sobre m e todo ponto de P_r está à direita ou sobre m
  - Pode ocorrer que o par mais próximo de {p,q} de P seja tal que $p \in P_l$ e $q \in P_r$
    - Este par deve estar na faixa de largura 2D que tem como centro a reta m



PAR DE PONTOS MAIS PRÓXIMOS
[Divisão-e-conquista em R^2]

- **Combinar**
  - Seja P_l'- subconjunto dos pontos de P_l que estão a uma distância menor do que D da linha m e seja P_r' a correspondente subconjunto de pontos de P_r
  - Para cada $p \in P_l$ $\min(D_l , D_r)$ e seja m uma reta tal que todo ponto de P_l' determinar os pontos $q_i \in P_r$' tal que $d(p,q_i) < D$
  - Os $q_i \in R_p$' (pontos de P_r' que estão no retângulo R_p de altura $2D$ e largura D

No máximo quantos pares de pontos de P_r' podem estar contidos no tal retângulo????

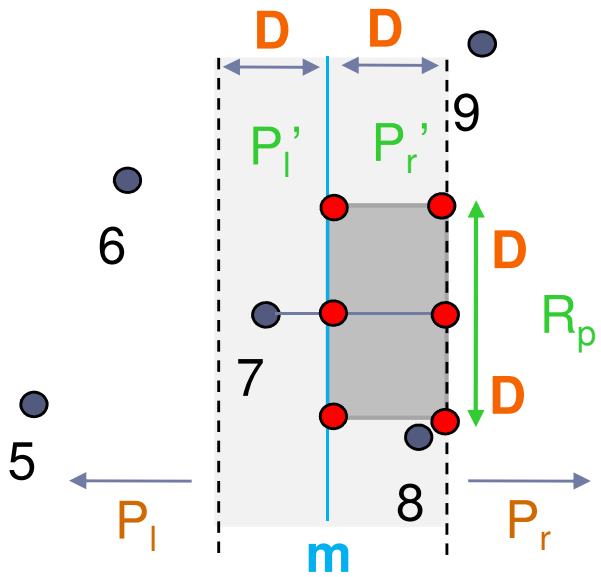
- **No máximo 6!!!**

Em caso contrario existe um par de pontos em P_r cuja distância seria menor que D o que não pode ocorrer.

FURB

PAR DE PONTOS MAIS PRÓXIMOS
[Divisão-e-conquista em R^2]

- **Combinar**
  - Seja P_l'- subconjunto dos pontos de P_l que estão a uma distância menor do que D da linha m e seja P_r' a correspondente subconjunto de pontos de P_r
  - Para cada p ∈ P_l min(D_l , D_r) e seja m uma reta tal que todo ponto de P_l' determinar os pontos q_i ∈ P_r' tal que $d(p,q_i) < D$
  - Os q_i ∈ R_p' (pontos de P_r' que estão no retângulo R_p de altura 2D e largura D



PAR DE PONTOS MAIS PRÓXIMOS
[Divisão-e-conquista em R^2]

- **Algoritmo PmP4(S)**
  - Dado um conjunto $S = (p_1, p_2, \ldots, p_n)$ de n pontos em R^2, devolve a distância mínima entre pontos de S
  - **Ordenar** os pontos de S por abscissa e armazene num vetor V_x
  - **Ordenar** os pontos de S por abscissa e armazene num vetor V_Y
  - **Retornar** o par obtido pelo algoritmo Pmp4-Aux(S)

FURB

PAR DE PONTOS MAIS PRÓXIMOS
[Divisão-e-conquista em R^2]

Algoritmo divisao-e-conquista R^2 (PmP4-Aux)

01. Se|S|= 2 então devolver (p_1, p_2)

02. Senão calcular a mediana m_x das abscissas de S. Seja r a reta vertical com abscissa m_x

03. Dividir S em duas coleções S_1 e S_2 com [n/2] e [n/2] pontos, respectivamente. Todos os pontos de S_1 estão à esquerda de (ou sobre) a reta r e os pontos de S_2 estão à direita (ou sobre) a reta r

    Obter recursivamente o Par mais Próximo

04. De S_1: (p_{i1} ,p_{j1}) = PmP4-Aux(S_1)

    De S_2: (p_{i2} ,p_{j2}) = PmP4-Aux(S_2)

05. Seja D = min{d(p_{i1}, p_{j1}), d(p_{i2}, p_{j2})}

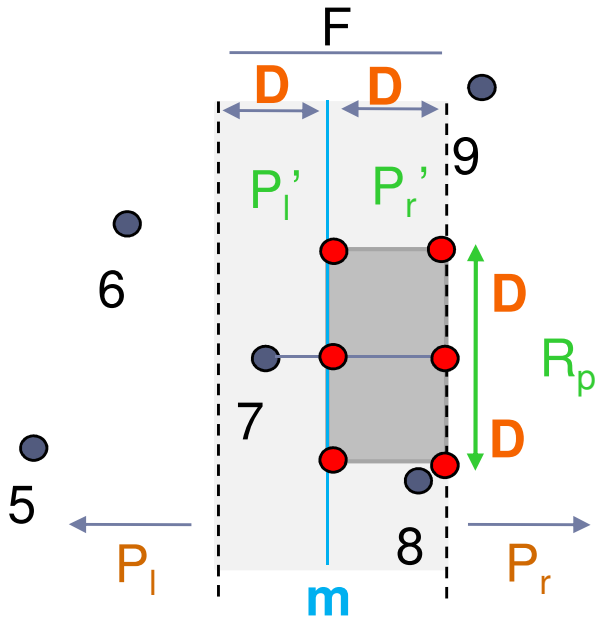
06. Seja F a faixa de largura 2D centrada na reta m. Procure por varredura vertical, conforme detalhado abaixo, o par de pontos $p_k \in S_1 \cap F$ e $p_1 \in S_2 \cap F$ mais próximos

07. Devolver: min{d(p_{i1},p_{j1}), d(p_{i2},p_{j2}), d(p_1,p_k))}

FURB

PAR DE PONTOS MAIS PRÓXIMOS
[Divisão-e-conquista em R^2]

Detalhamento do passo 06	
6.1	Utilizando o vetor ordenado V_y, percorra os pontos $S_1 \cap F$ em ordem ascendente de ordenadas. Para cada ponto p visitado, tomamos em V_y os três pontos de $S_2 \cap F$ acima e os três pontos abaixo de p. Como observamos anteriormente, o vizinho mais próximo de p em S_2 dentro da faixa F tem que ser um destes seis pontos, devido à escolha da largura D da faixa F
6.2	Devolva o para (p_k, p_l) que minimiza as distâncias encontradas



PAR DE PONTOS MAIS PRÓXIMOS
[Divisão-e-conquista em R^2]

- **Sumário**

Algoritmo divisao-e-conquista (PmP4-Aux)

Pré-processamento

Construir P_x e P_y como listas ordenadas pelas coordenadas x e y

Divisão

Quebrar P em P_l e P_r

Conquista

D_l = PmP4-Aux(P_l)

D_r = PmP4-Aux(P_r)

Combinação

D = min(**D_l**, **D_r**)

Determinar faixa divisória e pontos

Verificar se tem algum par com distancia < D na faixa divisória

FURB