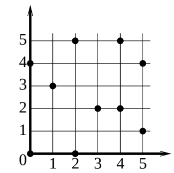
Par de pontos mais próximos

Problema: Dados n pontos no plano, determinar dois deles que estão a distância mínima.

Par de pontos mais próximos

Problema: Dados n pontos no plano, determinar dois deles que estão a distância mínima.

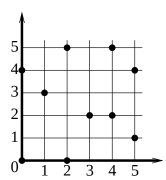
Entrada:
$(4,5),(3,1),(2,3),(0,2),(5,4),(5,2),(1,5),(2,4),(4,0),(0,0)$



Par de pontos mais próximos

Problema: Dados n pontos no plano, determinar dois deles que estão a distância 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 distância 1.

Par de pontos mais próximos

Problema: Dados n pontos no plano, determinar dois deles que estão a distância mínima.

Primeira solução: algoritmo quadrático,
que testa todos os pares de pontos.

Par de pontos mais próximos

Problema: Dados n pontos no plano, determinar dois deles que estão a distância mínima.

Primeira solução: algoritmo quadrático,
que testa todos os pares de pontos.

Vamos considerar o problema na reta.

Par mais próximo na reta

Problema: Dados n pontos numa reta, determinar dois deles que estão a distância mínima.

Primeira solução: ordene os pontos, e encontre os dois consecutivos mais próximos.
Tempo consumido: $O(n \lg n)$.

Par mais próximo na reta

Problema: Dados n pontos numa reta, determinar dois deles que estão a distância mínima.

Primeira solução: ordene os pontos, e encontre os dois consecutivos mais próximos.
Tempo consumido: $O(n \lg n)$.

Solução melhor: use divisão e conquista!

Divisão e conquista

Esse paradigma envolve os seguintes passos:

Divisão e conquista

Esse paradigma envolve os seguintes passos:

Divisão: dividir a instância do problema em instâncias menores do problema.

Divisão e conquista

Esse paradigma envolve os seguintes passos:

Divisão: dividir a instância do problema em instâncias menores do problema.

Conquista: resolver o problema nas instâncias menores recursivamente (ou diretamente, se elas forem pequenas o suficiente).

Divisão e conquista

Esse paradigma envolve os seguintes passos:

Divisão: dividir a instância do problema em instâncias menores do problema.

Conquista: resolver o problema nas instâncias menores recursivamente (ou diretamente, se elas forem pequenas o suficiente).

Combinação: combinar as soluções das instâncias menores para gerar uma solução da instância original.

Par mais próximo na reta

Entrada: vetor $x[1 \ldots n]$
Pré-processamento:
ordene x de modo que $x[1] \leq \cdots \leq x[n]$

Par mais próximo na reta

Entrada: vetor $x[1 \dots n]$
Pré-processamento:
ordene x de modo que $x[1] \leq \cdots \leq x[n]$

PAR+PROX (x, n)
se $n = 1$ **então devolva** ∞
$m \leftarrow \lfloor n/2 \rfloor$
$x_e[1\,..\,m] \leftarrow x[1\,..\,m]$
$x_d[1\,..\,n-m] \leftarrow x[m+1\,..\,n]$
$\delta_e \leftarrow$ PAR+PROX (x_e, m)
$\delta_d \leftarrow$ PAR+PROX $(x_d, n-m)$
$\delta \leftarrow \min\{\delta_e, \delta_d, x[m+1] - x[m]\}$
devolva δ

Par mais próximo na reta

Entrada: vetor $x[1 \dots n]$
Pré-processamento:
ordene x de modo que $x[1] \leq \cdots \leq x[n]$

PAR+PROX (x, n)
se $n = 1$ **então devolva** ∞
$m \leftarrow \lfloor n/2 \rfloor$
$x_e[1\,..\,m] \leftarrow x[1\,..\,m]$
$x_d[1\,..\,n-m] \leftarrow x[m+1\,..\,n]$
$\delta_e \leftarrow$ PAR+PROX (x_e, m)
$\delta_d \leftarrow$ PAR+PROX $(x_d, n-m)$
$\delta \leftarrow \min\{\delta_e, \delta_d, x[m+1] - x[m]\}$
devolva δ

Tempo consumido: $O(n \lg n)$ pelo pré-processamento
e $O(n)$ pelo PAR+PROX.

Par mais próximo no plano

Como generalizar essa idéia para o plano?

Par mais próximo no plano

Como generalizar essa idéia para o plano?

Discussão...

Par-Mais-Próximo

PAR+PROX (x, y, n)
para $i \leftarrow 1$ **até** n **faça**
  $p_x[i] \leftarrow i$
  $p_y[i] \leftarrow i$
MERGE-SORT(p_x, n, x) ▷ ordenação indireta
MERGE-SORT(p_y, n, y) ▷ ordenação indireta
devolva PAR+PROX-REC(x, y, p_x, p_y, n)

Par-Mais-Próximo

PAR+PROX (x, y, n)
para $i \leftarrow 1$ **até** n **faça**
  $p_x[i] \leftarrow i$
  $p_y[i] \leftarrow i$
MERGE-SORT(p_x, n, x) ▷ ordenação indireta
MERGE-SORT(p_y, n, y) ▷ ordenação indireta
devolva PAR+PROX-REC(x, y, p_x, p_y, n)

PAR+PROX-REC (x, y, p_x, p_y, n)
se $n \leq 3$
  então ▷ resolva o problema diretamente
  senão $(p_x^e, p_y^e, n^e, p_x^d, p_y^d, n^d) \leftarrow$ DIVIDE(x, y, p_x, p_y, n)
    $(i^e, j^e) \leftarrow$ PAR+PROX-REC $(x, y, p_x^e, p_y^e, n^e)$
    $(i^d, j^d) \leftarrow$ PAR+PROX-REC $(x, y, p_x^d, p_y^d, n^d)$
    devolva COMBINA $(x, y, i^e, j^e, i^d, j^d, p_x, p_y, n)$

Par-Mais-Próximo

DIVIDE (x, y, p_x, p_y, n)

1 $n^e \leftarrow \lfloor n/2 \rfloor$
2 $n^d \leftarrow n - n^e$
3 $p_x^e[1 \,.\,.\, n^e] \leftarrow p_x[1 \,.\,.\, n^e]$
4 $p_x^d[1 \,.\,.\, n^d] \leftarrow p_x[n^e + 1 \,.\,.\, n]$
5 $x_c \leftarrow x[p_x[n^e]]$
6 $i \leftarrow 0$
7 $j \leftarrow 0$
8 **para** $k \leftarrow 1$ **até** n **faça**
9   **se** $x[p_y[k]] \leq x_c$
10   **então** $i \leftarrow i + 1$
11             $p_y^e[i] \leftarrow p_y[k]$
12   **senão** $j \leftarrow j + 1$
13             $p_y^d[j] \leftarrow p_y[k]$
14 **devolva** $(p_x^e, p_y^e, n^e, p_x^d, p_y^d, n^d)$

Par-Mais-Próximo

COMBINA $(x, y, i^e, j^e, i^d, j^d, p_x, p_y, n)$

1 $\delta \leftarrow \min\{$DIST(x, y, i^e, j^e), DIST$(x, y, i^d, j^d)\}$
2 $x_c \leftarrow x[p_x[\lfloor n/2 \rfloor]]$
3 $m \leftarrow \delta$ $t \leftarrow 0$
4 **para** $k \leftarrow 1$ **até** n **faça**
5   **se** $|x[p_y[k]] - x_c| \leq \delta$
6   **então** $t \leftarrow t + 1$ $f_y[t] \leftarrow p_y[k]$
7 **para** $i \leftarrow 1$ **até** $t - 1$ **faça**
8   **para** $j \leftarrow i + 1$ **até** $\min\{i + 7, t\}$ **faça**
9         $d \leftarrow$ DIST$(x, y, f_y[i], f_y[j])$
10   **se** $d < m$
11   **então** $m \leftarrow d$
12                 $p_1 \leftarrow f_y[i]$ $p_2 \leftarrow f_y[j]$
13 **se** $m = \delta$
14   **então devolva**
argmin$\{$DIST(x, y, i^e, j^e), DIST$(x, y, i^d, j^d)\}$
15   **senão devolva** (p_1, p_2)

Consumo de tempo

Quanto tempo consome o DIVIDE em função de n?

linha	consumo de todas as execuções da linha
1-2	$\mathrm{O}(1)$
3-4	$\mathrm{O}(n)$
5-7	$\mathrm{O}(1)$
8-9	$\mathrm{O}(n)$
10–13	$\mathrm{O}(n)$
14	$\mathrm{O}(1)$
total	$\mathrm{O}(3n + 3) = \mathrm{O}(n)$

Consumo de tempo

Quanto tempo consome o COMBINA em função de n?

linha	consumo de todas as execuções da linha	
1-3	$\mathrm{O}(1)$	
4-6	$\mathrm{O}(n)$	
7	$\mathrm{O}(n)$	
8–12	$\mathrm{O}(n)$	$\triangleright\ 7\,\mathrm{O}(n) = O(n)$
13–15	$\mathrm{O}(1)$	
total	$\mathrm{O}(3n + 2) = \mathrm{O}(n)$	

Consumo de tempo

Para determinar o consumo de tempo do algoritmo PAR+PROX-REC, que é recursivo, precisamos derivar uma recorrência que descreva o seu consumo de tempo.

Seja $T(n)$ o tempo consumido pelo PAR+PROX-REC.

Consumo de tempo

Para determinar o consumo de tempo do algoritmo PAR+PROX-REC, que é recursivo, precisamos derivar uma recorrência que descreva o seu consumo de tempo.

Seja $T(n)$ o tempo consumido pelo PAR+PROX-REC.

Vale a seguinte recorrência para $T(n)$:

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \mathrm{O}(n)$$

Consumo de tempo

Para determinar o consumo de tempo do algoritmo PAR+PROX-REC, que é recursivo, precisamos derivar uma recorrência que descreva o seu consumo de tempo.

Seja $T(n)$ o tempo consumido pelo PAR+PROX-REC.

Vale a seguinte recorrência para $T(n)$:

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \mathrm{O}(n)$$

Essa é a mesma recorrência que a do MERGE-SORT, e portanto temos que $T(n) = \mathrm{O}(n \lg n)$.

Consumo de tempo

Para determinar o consumo de tempo do algoritmo PAR+PROX-REC, que é recursivo, precisamos derivar uma recorrência que descreva o seu consumo de tempo.

Seja $T(n)$ o tempo consumido pelo PAR+PROX-REC.

Vale a seguinte recorrência para $T(n)$:

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \mathrm{O}(n)$$

Essa é a mesma recorrência que a do MERGE-SORT, e portanto temos que $T(n) = \mathrm{O}(n \lg n)$.

Disso é fácil concluir que o consumo de tempo do PAR+PROX também é $\mathrm{O}(n \lg n)$.

Versão um pouco diferente

O algoritmo COMBINA pode ser escrito de maneira um pouco diferente.

Na versão apresentada a seguir, o ajustamos para devolver apenas a distância mínima, e não um par de pontos a distância mínima.

Naturalmente o algoritmo PAR+PROX-REC tem que ser ajustado de acordo, mas omitimos isso aqui.

Versão alternativa do Combina

COMBINA $(x, y, \delta^e, \delta^d, p_x, p_y, n)$

$\delta \leftarrow \min\{\delta^e, \delta^d\}$
$x_c \leftarrow x[p_x[\lfloor n/2 \rfloor]]$
$m \leftarrow \delta$ $\quad$ $t \leftarrow 0$
para $k \leftarrow 1$ **até** n **faça**
  se $|x[p_y[k]] - x_c| \leq \delta$
    então $t \leftarrow t + 1$ $\quad$ $f_y[t] \leftarrow p_y[k]$
para $i \leftarrow 1$ **até** $t - 1$ **faça**
  $j \leftarrow i + 1$
  enquanto $j \leq t$ **e** $y[f_y[j]] - y[f_y[i]] < \delta$ **faça**
    se DIST$(x, y, f_y[i], f_y[j]) < m$
      então $m \leftarrow d$
    $j \leftarrow j + 1$
devolva m