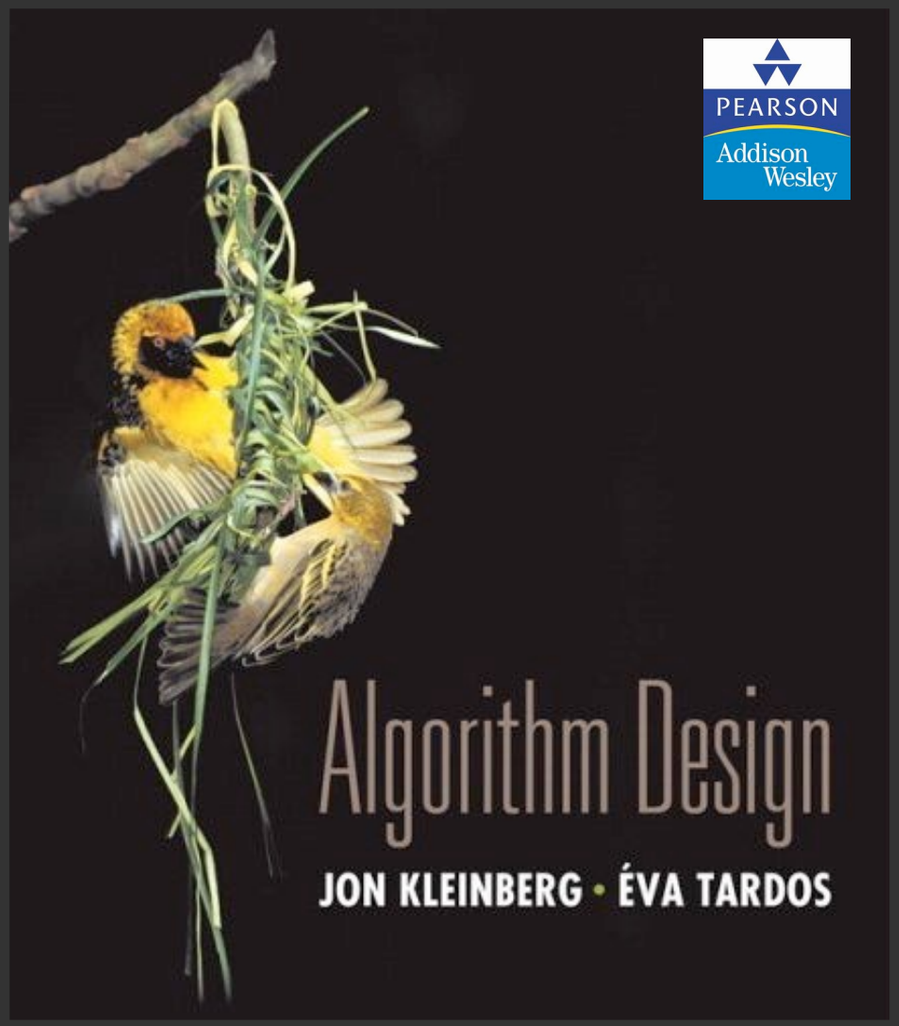


5. Divide y vencerás

Lecture slides by Kevin Wayne

http://www.cs.princeton.edu/~wayne/kleinberg-tardos

Last updated on 12/9/23 10:56

Paradigma de Divide-y-Vencerás (Divide&Conquer)

Divide-y-Vencerás.

- Dividir un problema en múltiples sub-problemas.
- Resolver cada subproblema de forma recursiva.
- Combinar las soluciones de los subproblemas en una general.

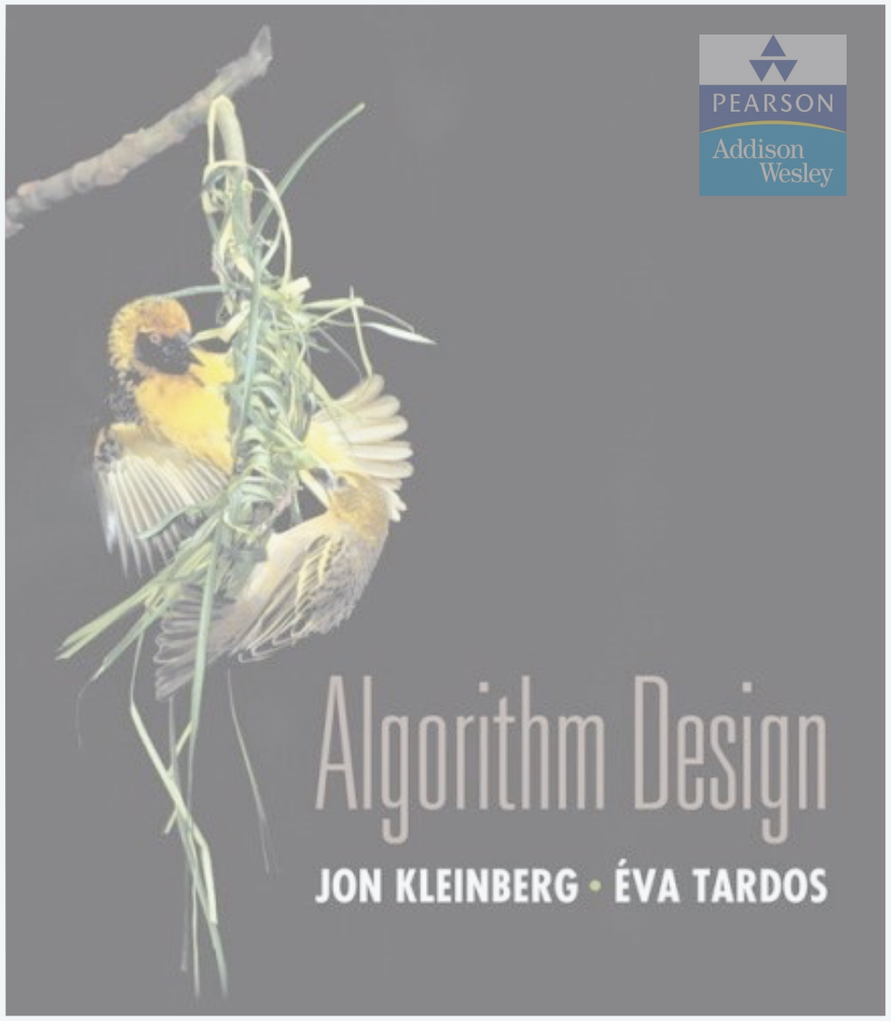
Uso más común.

- Dividir el problema de tamaño n en dos subproblemas (del mismo tipo) de tamaño $n/2$ en tiempo lineal.
- Resolver los dos subproblemas de forma recursiva.
- Combinar las dos soluciones en una general en tiempo lineal.

Consecuencia.

- Fuerza bruta: $\Theta(n^2)$.
- Divide-y-vencerás: $\Theta(n \log n)$.

atribuída a Julio César

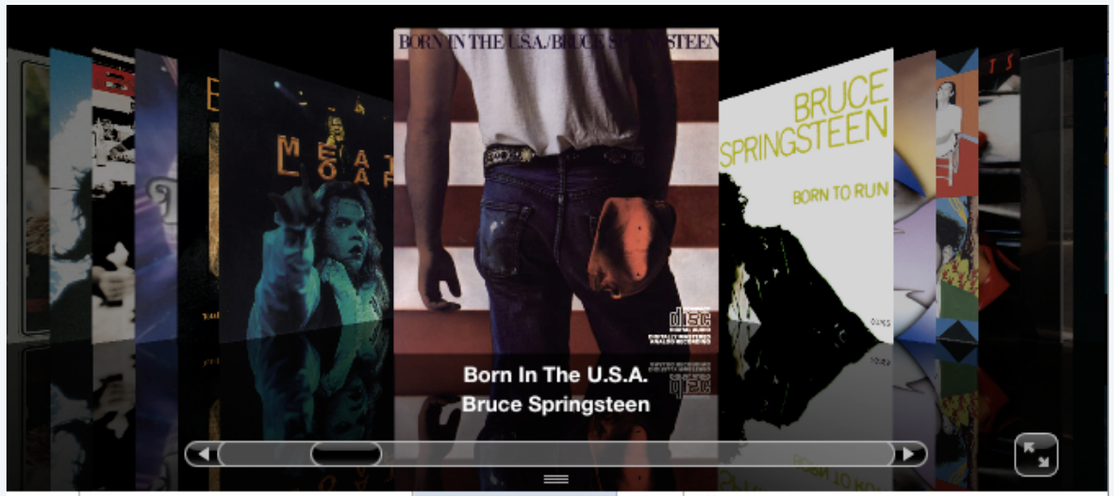


5. Divide y Vencerás

- *mergesort*
- *contar inversiones*
- *par de puntos más cercanos*

Problema de ordenamiento

Problema. Dada una lista de n elementos de un universo totalmente ordenado, reorganizarlos en orden ascendente.



	Name	Artist	Time	Album
12	Let It Be	The Beatles	4:03	Let It Be
13	Take My Breath Away	BERLIN	4:13	Top Gun – Soundtrack
14	Circle Of Friends	Better Than Ezra	3:27	Empire Records
15	Dancing With Myself	Billy Idol	4:43	Don't Stop
16	Rebel Yell	Billy Idol	4:49	Rebel Yell
17	Piano Man	Billy Joel	5:36	Greatest Hits Vol. 1
18	Pressure	Billy Joel	3:16	Greatest Hits, Vol. II (1978 – 1985) (Disc 2)
19	The Longest Time	Billy Joel	3:36	Greatest Hits, Vol. II (1978 – 1985) (Disc 2)
20	Atomic	Blondie	3:50	Atomic: The Very Best Of Blondie
21	Sunday Girl	Blondie	3:15	Atomic: The Very Best Of Blondie
22	Call Me	Blondie	3:33	Atomic: The Very Best Of Blondie
23	Dreaming	Blondie	3:06	Atomic: The Very Best Of Blondie
24	Hurricane	Bob Dylan	8:32	Desire
25	The Times They Are A-Changin'	Bob Dylan	3:17	Greatest Hits
26	Livin' On A Prayer	Bon Jovi	4:11	Cross Road
27	Beds Of Roses	Bon Jovi	6:35	Cross Road
28	Runaway	Bon Jovi	3:53	Cross Road
29	Rasputin (Extended Mix)	Boney M	5:50	Greatest Hits
30	Have You Ever Seen The Rain	Bonnie Tyler	4:10	Faster Than The Speed Of Night
31	Total Eclipse Of The Heart	Bonnie Tyler	7:02	Faster Than The Speed Of Night
32	Straight From The Heart	Bonnie Tyler	3:41	Faster Than The Speed Of Night
33	Holding Out For A Hero	Bonny Tyler	5:49	Meat Loaf And Friends
34	Dancing In The Dark	Bruce Springsteen	4:05	Born In The U.S.A.
35	Thunder Road	Bruce Springsteen	4:51	Born To Run
36	Born To Run	Bruce Springsteen	4:30	Born To Run
37	Jungleland	Bruce Springsteen	9:34	Born To Run

Mergesort

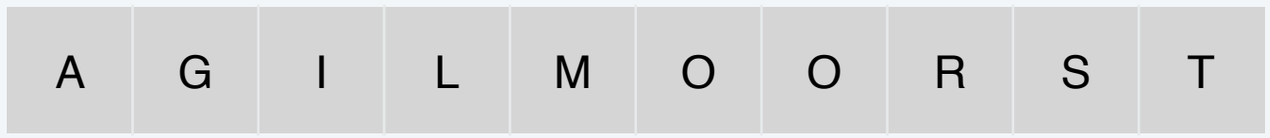
- Ordenar de forma recursiva la mitad izquierda.
- Ordenar de forma recursiva la mitad derecha.
- Fusionar (*merge*) las dos mitades para ordenar el conjunto.

entrada

ordenar mitad izquierda

ordenar mitad derecha

fusionar los resultados

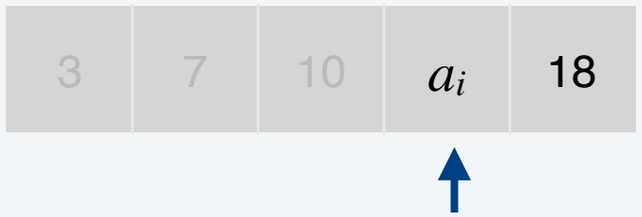


Fusionar

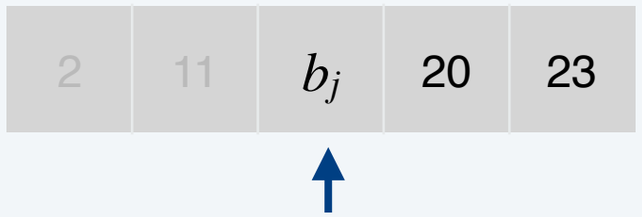
Objetivo. Combinar dos listas ordenadas A y B en una lista completa C.

- Leer A y B de izquierda a derecha.
- Comparar a_i y b_j.
- Si $a_i \leq b_j$, agregar a_i a C (no es más grande que ningún elem. restante en B).
- If $a_i > b_j$, agregar b_j a C (más chico que cualquier elemento restante en A).

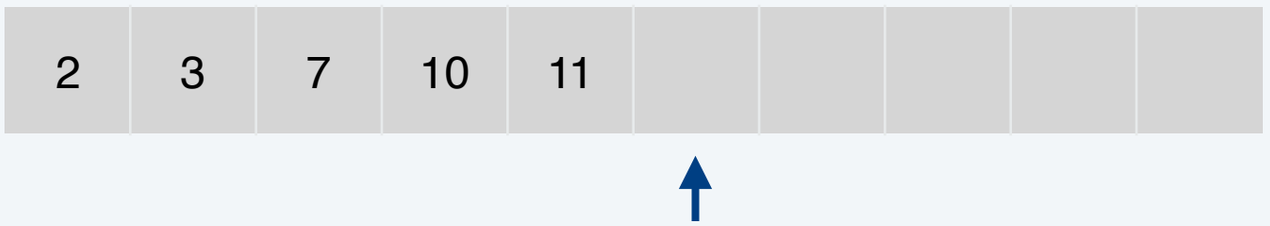
Lista ordenada A



Lista ordenada B



fusionar para formar la lista ordenada C



Una relación de recurrencia útil

Def. $T(n)$ = máximo número de comparaciones realizadas por el algoritmo de mergesort para una lista de largo $\leq n$.

Recurrencia del Mergesort.

$$T(n) \leq \begin{cases} 0 & \text{si } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n & \text{en otro caso} \end{cases}$$

Solución. $T(n)$ es $O(n \log_2 n)$.

Diferentes pruebas. Describimos varias formas de resolver esta recurrencia. Inicialmente asumimos que n es una potencia de 2 y reemplazamos $\leq$ con $=$ en la recurrencia.

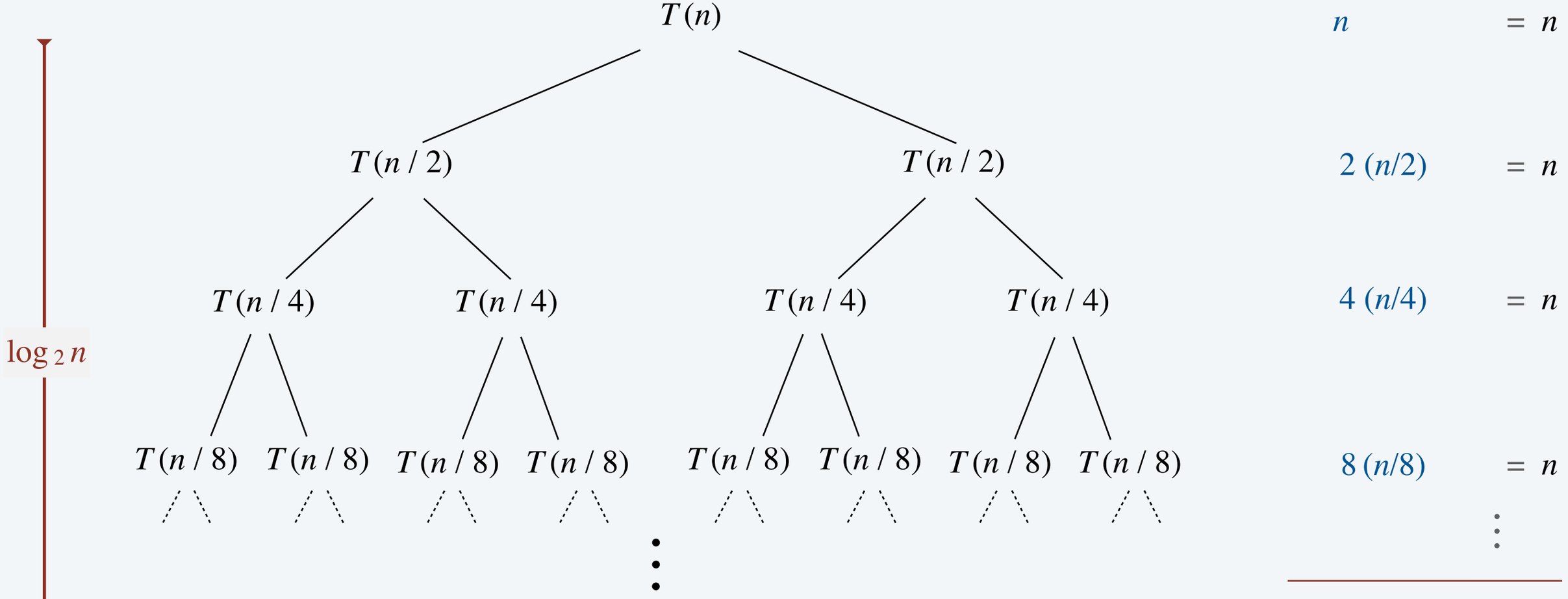
Recurrencia de divide-y-vencerás: prueba por árbol de recursión

Proposición. Si $T(n)$ satisface la siguiente recurrencia, entonces $T(n) = n \log_2 n$.

$$T(n) = \begin{cases} 0 & \text{si } n = 1 \\ 2\,T(n/2) + n & \text{en otro caso} \end{cases}$$

asumiendo que n es una potencia de 2

Dem 1.



Prueba por inducción

Proposición. Si $T(n)$ satisface la siguiente recurrencia, entonces $T(n) = n \log_2 n$.

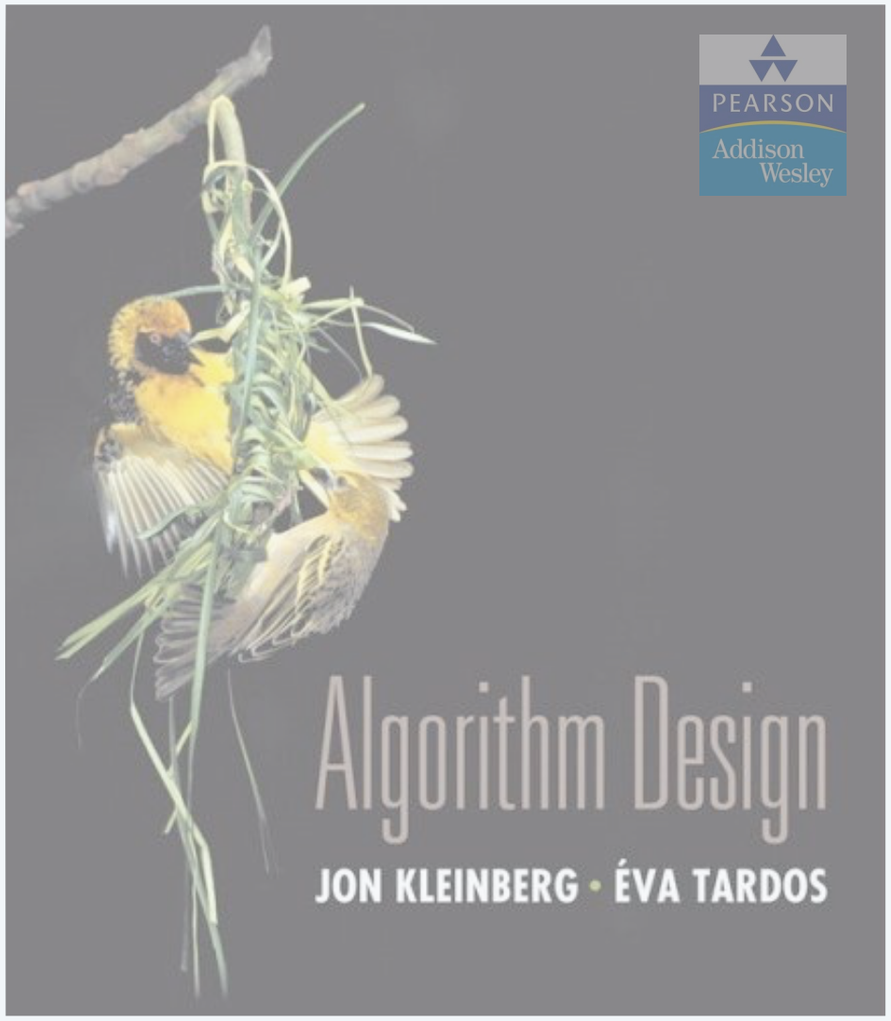
$$T(n) = \begin{cases} 0 & \text{si } n = 1 \\ 2\,T(n/2) + n & \text{en otro caso} \end{cases}$$

asumiendo que n es una potencia de 2

Dem 2. [por inducción en n]

- Caso base: cuando $n = 1$, $T(1) = 0 = n \log_2 n$.
- Hipótesis inductiva: asumimos que $T(n) = n \log_2 n$.
- Objetivo: mostrar que $T(2n) = 2n \log_2 (2n)$.

$$\begin{aligned} T(2n) &= 2\,T(n) + 2n \\ &= 2\,n \log_2 n + 2n = 2n\,(\log_2 n + 1 - 1) + 2n \\ &= 2n\,(\log_2 n + \log_2 2 - 1) + 2n = 2\,n\,(\log_2 (2n) - 1) + 2n \\ &= 2\,n \log_2 (2n) - 2\,n + 2n = 2\,n \log_2 (2n). \quad \blacksquare \end{aligned}$$



5. Divide y Vencerás

- *mergesort*
- *contar inversiones*
- *par de puntos más cercanos*

Contar inversiones

Un sitio de música quiere medir qué tan similares son las preferencias musicales de un usuario con las de otros.

- Rankear n canciones.
- El sitio de música consulta su base de datos para encontrar usuarios con gusto similar.

Métrica de similitud: número de inversiones entre dos rankings.

- Mi ranking: $1, 2, \ldots, n$.
- Otro ranking: $a_1, a_2, \ldots, a_n$.
- Las canciones i y j están invertidas si $i < j$, pero $a_i > a_j$.

	A	B	C	D	E
yo	1	2	3	4	5
otro	1	3	4	2	5

2 inversiones: 3-2, 4-2

Fuerza bruta: verificar los $\Theta(n^2)$ pares.

Contar inversiones: divide-y-vencerás

- Dividir: separar la lista en dos mitades A y B.
- Vencer: contar las inversiones en cada lista de forma recursiva.
- Combinar: contar inversiones (a, b) con $a \in A$ y $b \in B$.
- Retornar la suma de los tres totales.

entrada

1	5	4	8	10	2	6	9	3	7

contar inv. en la mitad izq. de A

1	5	4	8	10

5-4

contar inv. en la mitad der. de B

2	6	9	3	7

6-3 9-3 9-7

contar inv. (a, b) con a ∈ A y b ∈ B

1	5	4	8	10

2	6	9	3	7

4-2 4-3 5-2 5-3 8-2 8-3 8-6 8-7 10-2 10-3 10-6 10-7 10-9

salida 1 + 3 + 13 = 17

Contar inversiones: ¿cómo combinar dos subproblemas?

Q. Cómo contar inversiones (a, b) donde $a \in A$ y $b \in B$?

A. Fácil si A y B están ordenados!

Algoritmo de calentamiento.

- Ordenar A y B.
- Para cada elemento $b \in B$,
  - Búsqueda binaria en A para encontrar cuántos elementos en A son mayores que b.

lista A

7	10	18	3	14

lista B

20	23	2	11	16

ordenar A

3	7	10	14	18

ordenar B

2	11	16	20	23

búsqueda binaria para encontrar inversiones (a, b) con a ∈ A y b ∈ B

3	7	10	14	18

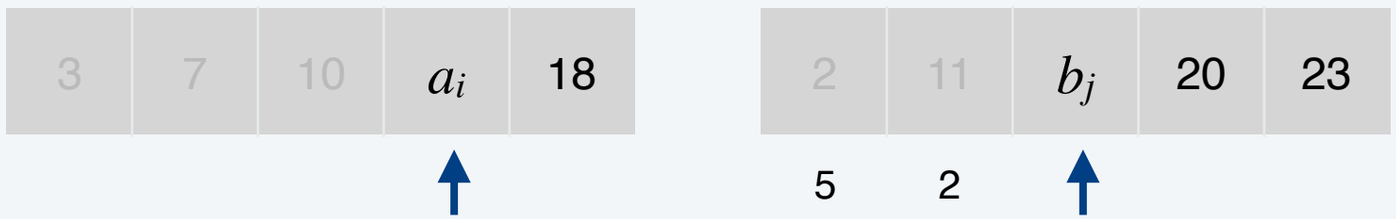
2	11	16	20	23
5	2	1	0	0

Contar inversiones: cómo combinar dos subproblemas?

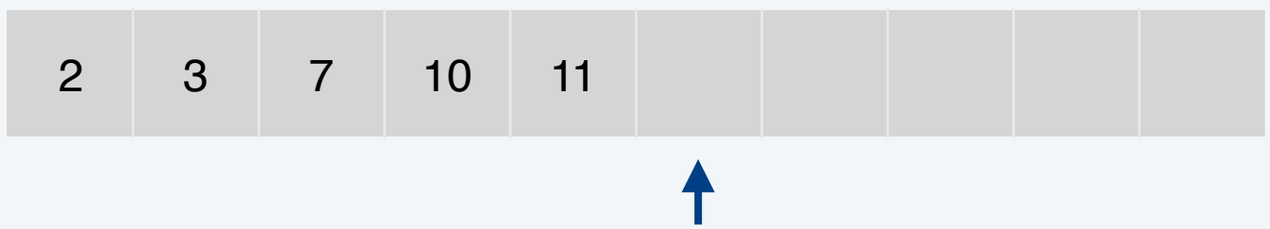
Contar inversiones (a, b) con $a \in A$ y $b \in B$, asumiendo que A y B están ordenados.

- Leer A y B de izquierda a derecha.
- Comparar a_i y b_j.
- Si $a_i < b_j$, entonces a_i no está invertido con ninguno de los elementos restantes en B.
- Si $a_i > b_j$, entonces b_j está invertido con cada elemento restante en A.
- Agregar el elemento menor a la lista ordenada C.

contar inversiones (a, b) con a ∈ A y b ∈ B



merge para formar la lista ordenada C



Contando inversiones: implementación del algoritmo divida&vencerás

Entrada. Lista L.

Salida. Número de inversiones en L y lista ordenada de elementos L'.

SORT-AND-COUNT (L)

IF *lista L tiene un único elemento*

RETURN $(0, L)$.

DIVIDIR la lista en dos mitades A *y* B.

$(r_A , A) \leftarrow$ *SORT-AND-COUNT*(A).

$(r_B , B) \leftarrow$ *SORT-AND-COUNT*(B).

$(r_{AB} , L') \leftarrow$ *MERGE-AND-COUNT*(A, B).

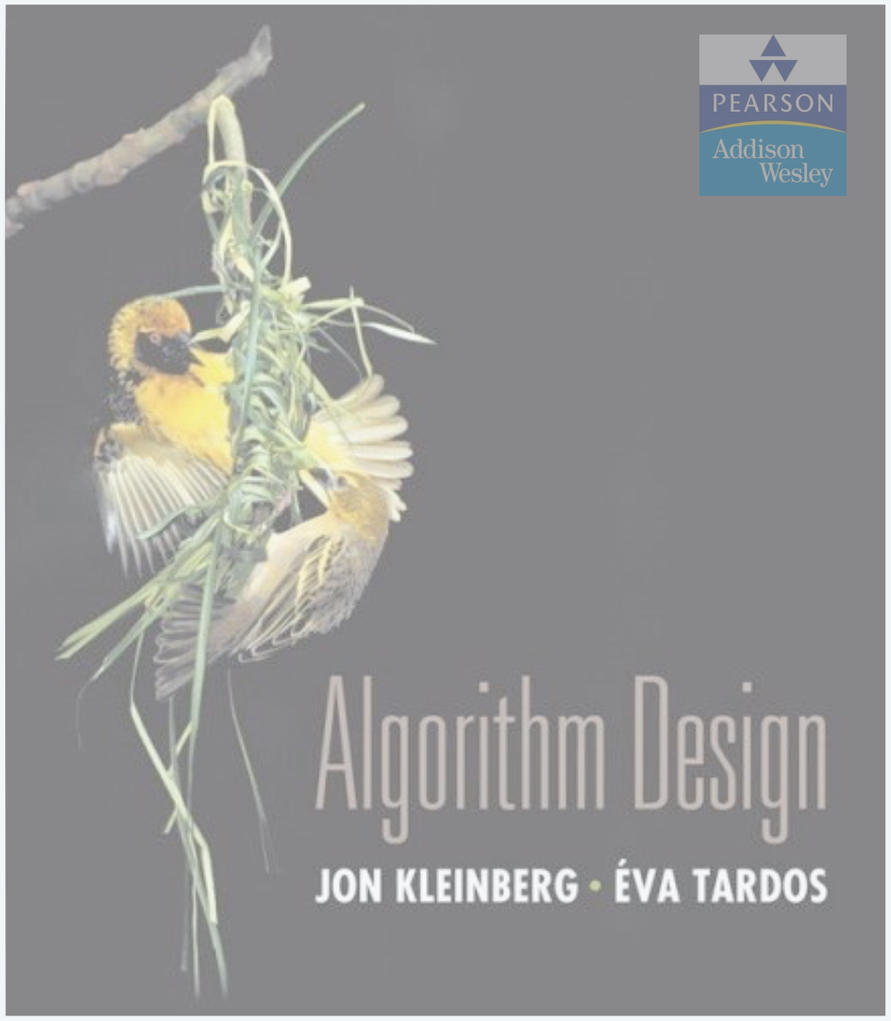
RETURN $(r_A + r_B + r_{AB} , L')$.

Contando inversiones: análisis del algoritmo divide y vencerás

Proposición. El algoritmo de sort-and-count cuenta el número de inversiones en una permutación de tamaño n en tiempo $O(n \log n)$.

Dem. El tiempo de ejecución en el peor caso $T(n)$ satisface la siguiente recurrencia:

$$T(n) = \begin{cases} \Theta(1) & \text{si } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) & \text{en otro caso} \end{cases}$$



5. Divide y Vencerás

- *mergesort*
- *contar inversiones*
- *par de puntos más cercanos*

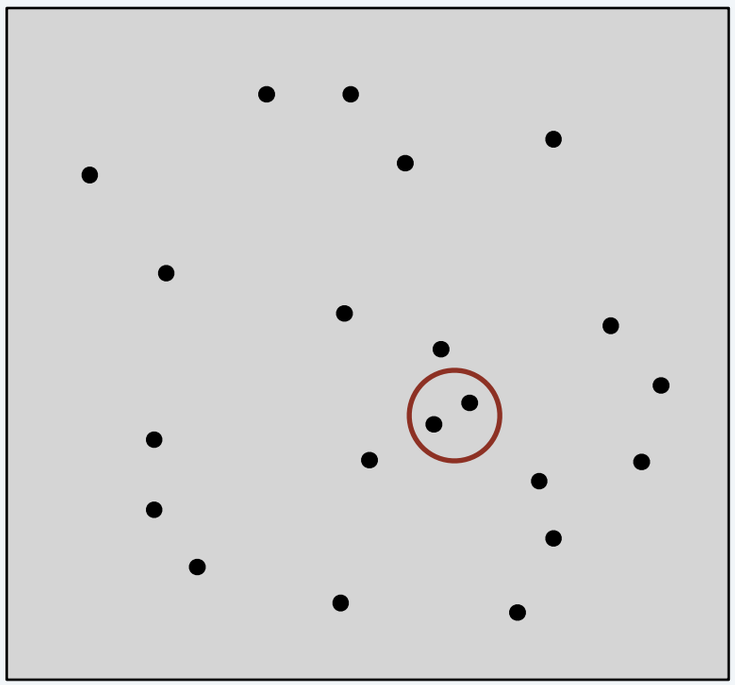
Par de puntos más cercanos

Problema de los puntos más cercanos. Dados n puntos en un plano, encontrar el par de puntos con la menor distancia Euclidiana entre ellos.

Primitiva geométrica fundamental.

- Sistemas de información geográfica, modelado molecular, control de tráfico aéreo.
- Caso especial del vecino más cercano, MST Euclideano, Voronoi.

el algoritmo eficiente para encontrar el par de puntos más cercano
inspiró algoritmos rápidos para estos problemas



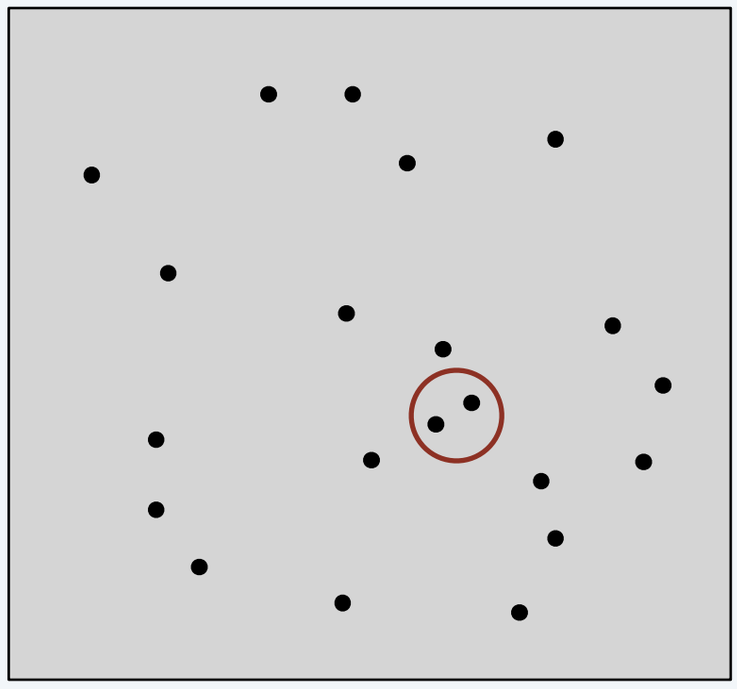
Par de puntos más cercanos

Problema de los puntos más cercanos. Dados n puntos en un plano, encontrar el par de puntos con la menor distancia Euclidiana entre ellos.

Fuerza bruta. Verificar todos los pares con $\Theta(n^2)$ cálculos de distancias.

Versión 1d. Fácil, algoritmo $O(n \log n)$ si todos los puntos están en una línea.

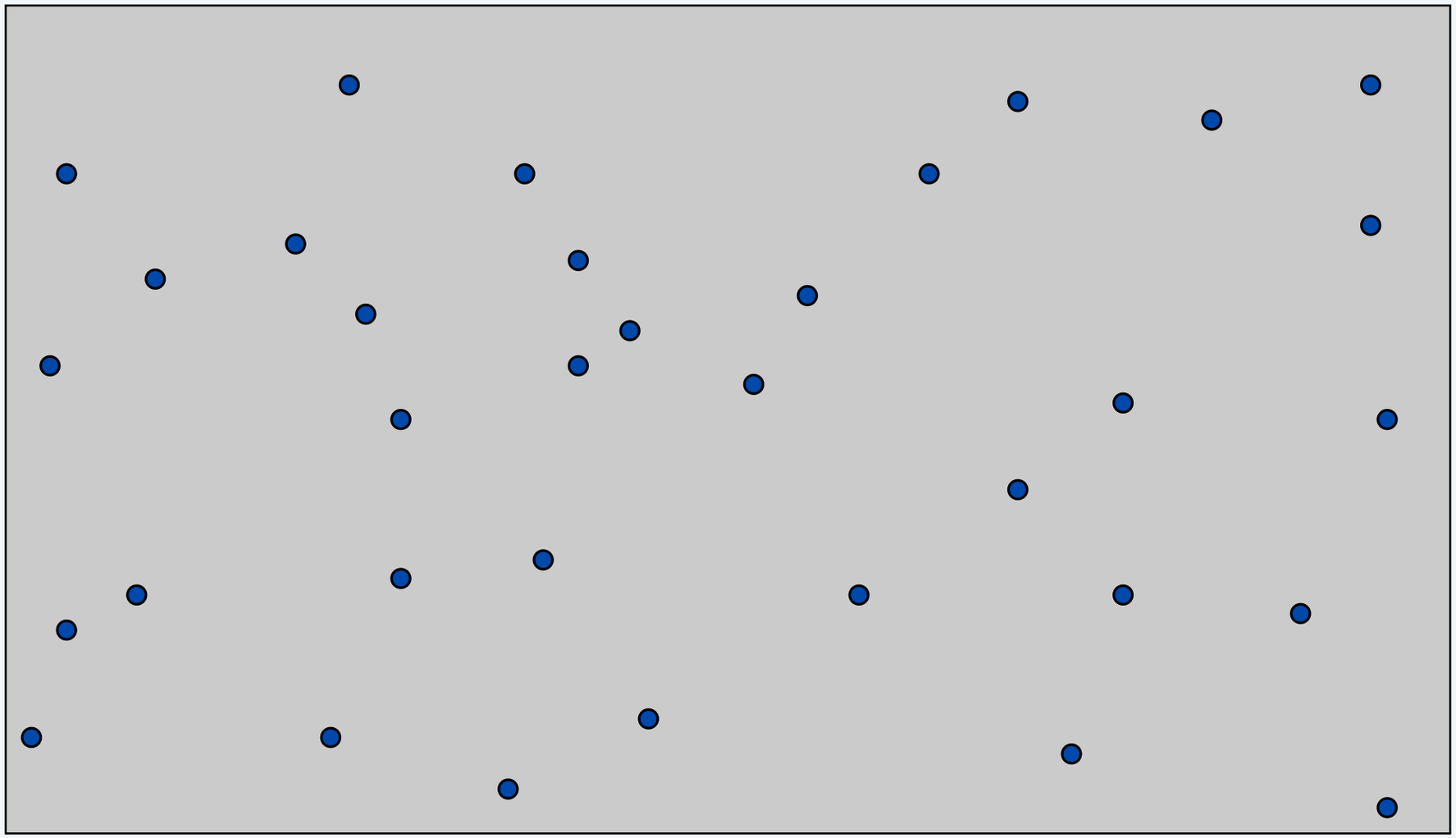
Asumimos. No hay dos puntos con la misma coordenada en x.



Par de puntos más cercanos: Un primer intento

Solución ordenando.

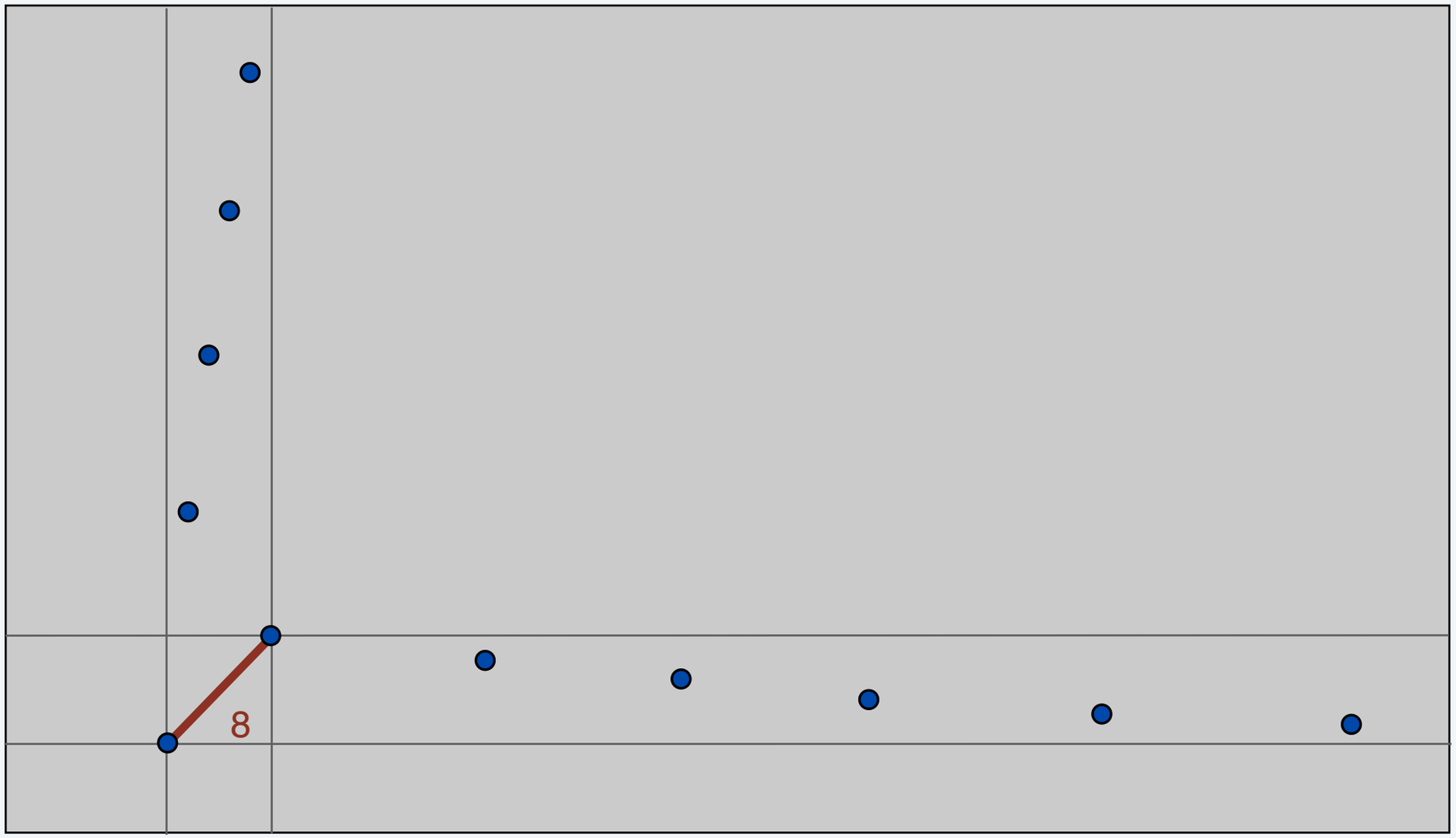
- Ordenamos por la coordenada x y consideramos puntos cercanos.
- Ordenamos por la coordenada y y consideramos puntos cercanos.



Par de puntos más cercanos: Un primer intento

Solución ordenando.

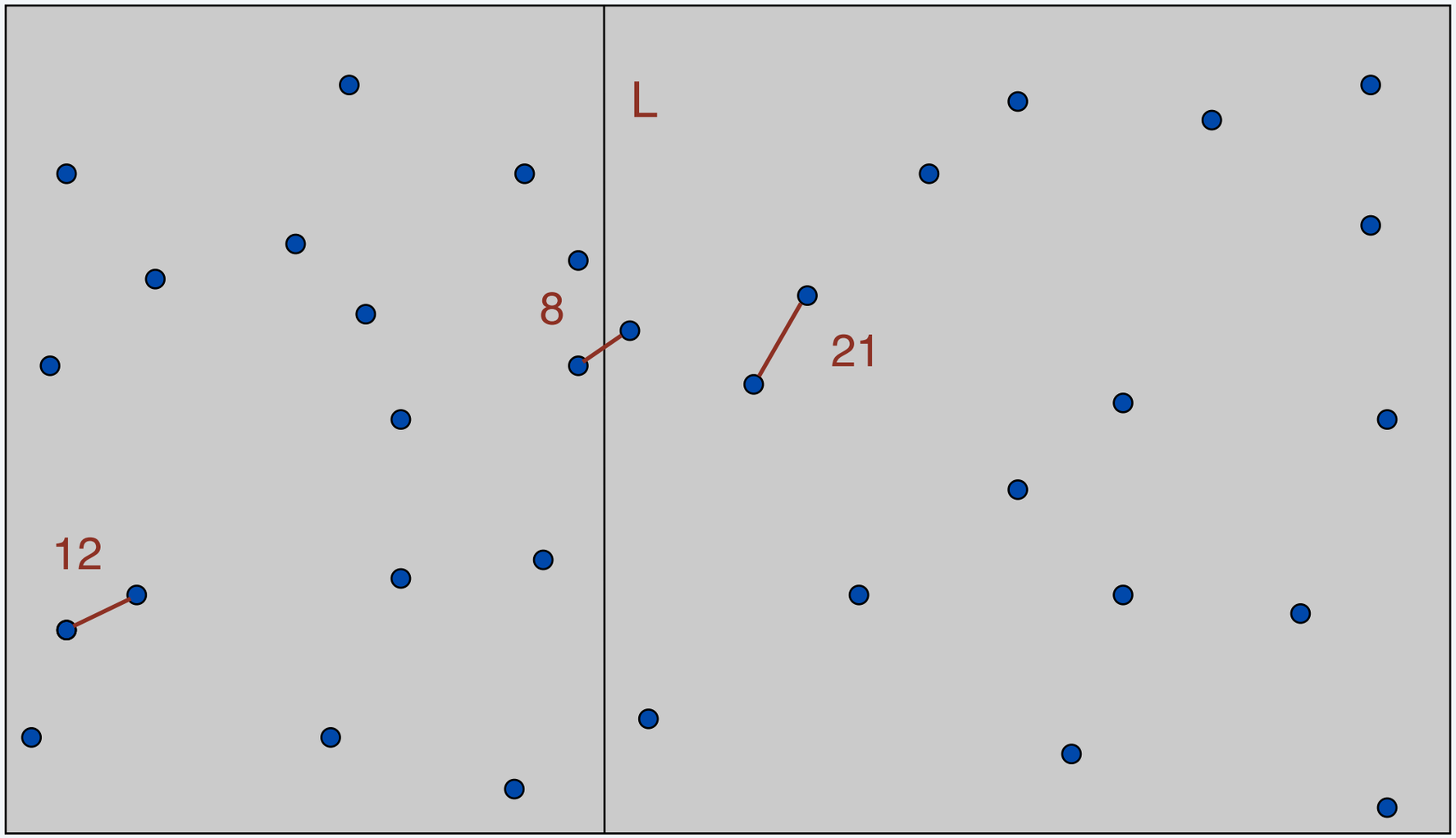
- Ordenamos por la coordenada x y consideramos puntos cercanos.
- Ordenamos por la coordenada y y consideramos puntos cercanos.
- Podemos ver un contraejemplo donde los puntos más cercanos en la coordenada x y en la coordenada y, no coinciden con los más cercanos en el plano.



Par de puntos más cercanos: algoritmo divide-y-vencerás

- Dividir: dibujar una línea vertical L tal que hayan $n/2$ puntos en cada lado.
- Vencer: Encontrar el par de puntos más cercano en cada mitad de forma recursiva.
- Combinar: Encontrar el par más cercano con un punto en cada lado.
- Retornar la mejor de las 3 soluciones.

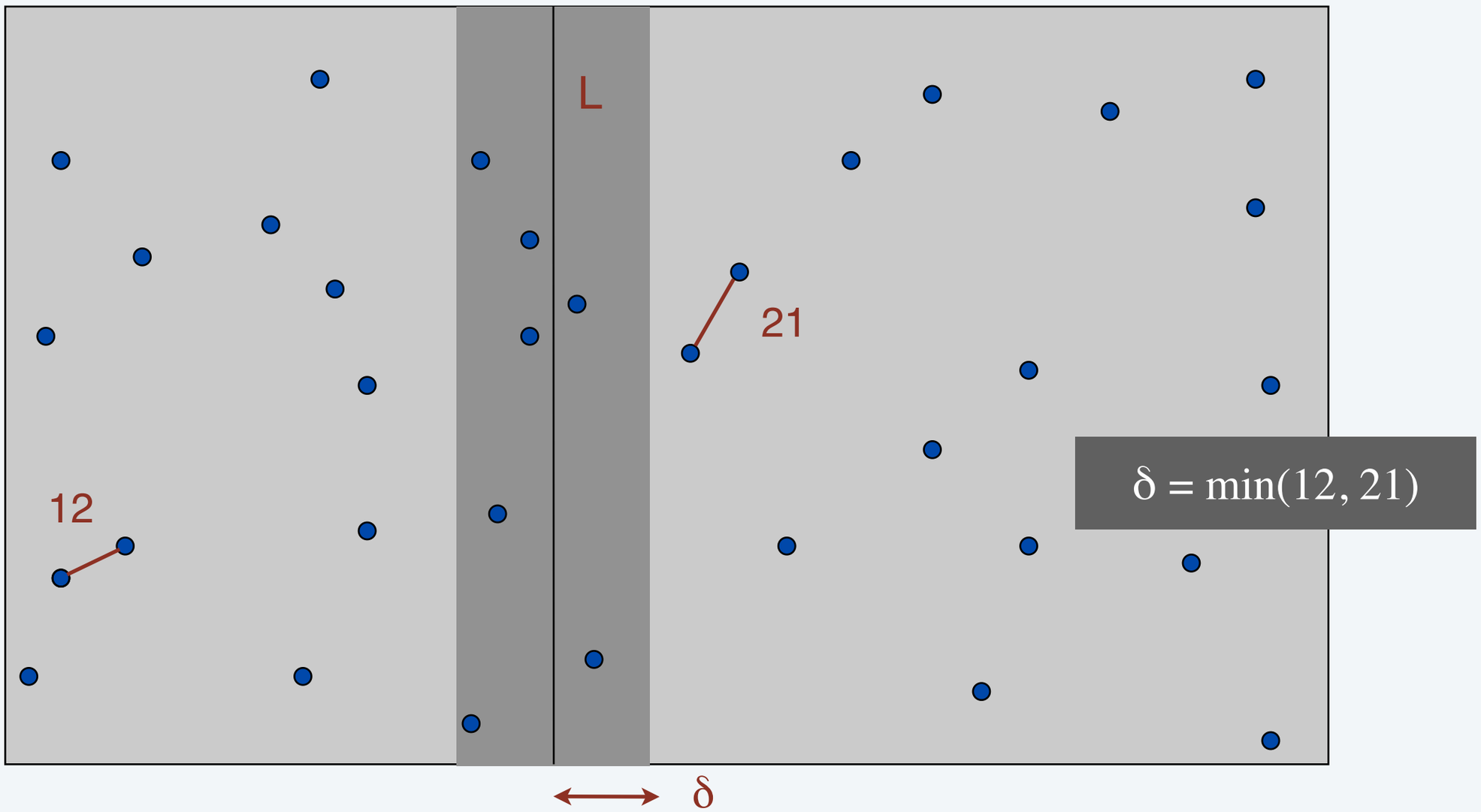
parecería ser $\Theta(N^2)$



¿Cómo encontramos el par más cercano con un punto en cada lado?

Encontrar el par más cercano con un punto en cada lado, asumiendo que la distancia es $< \delta$.

- Observación: solo necesitamos considerar los puntos con distancia menor a δ de la línea L.

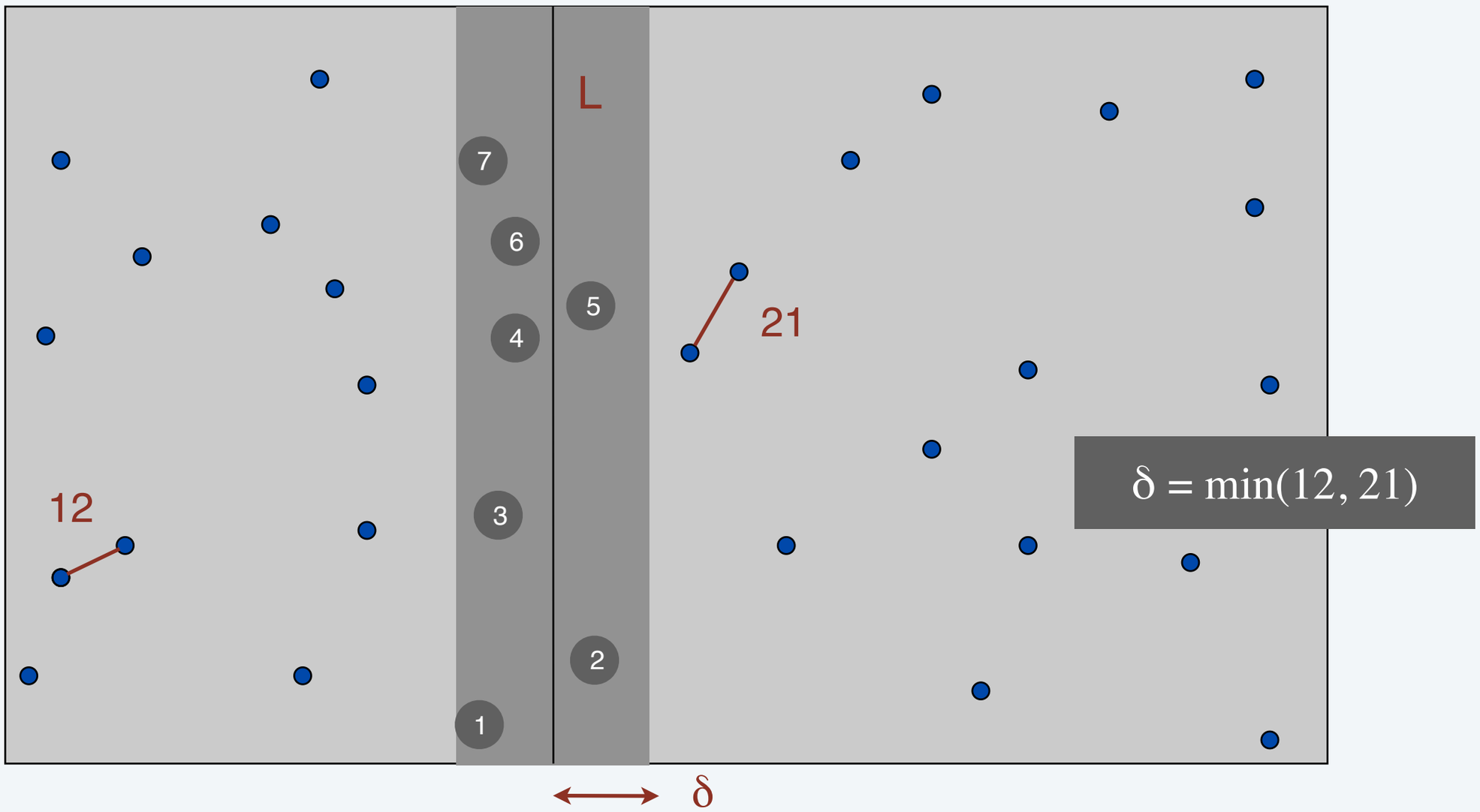


¿Cómo encontramos el par más cercano con un punto en cada lado?

Encontrar el par más cercano con un punto en cada lado, asumiendo que la distancia es $< \delta$.

- Observación: solo necesitamos considerar los puntos con distancia menor a δ de la línea L.
- Ordenar los puntos en la franja $2\,\delta$ por su coordenada y.
- Solo chequear las distancias de aquellos que están entre 11 posiciones.

por qué 11?



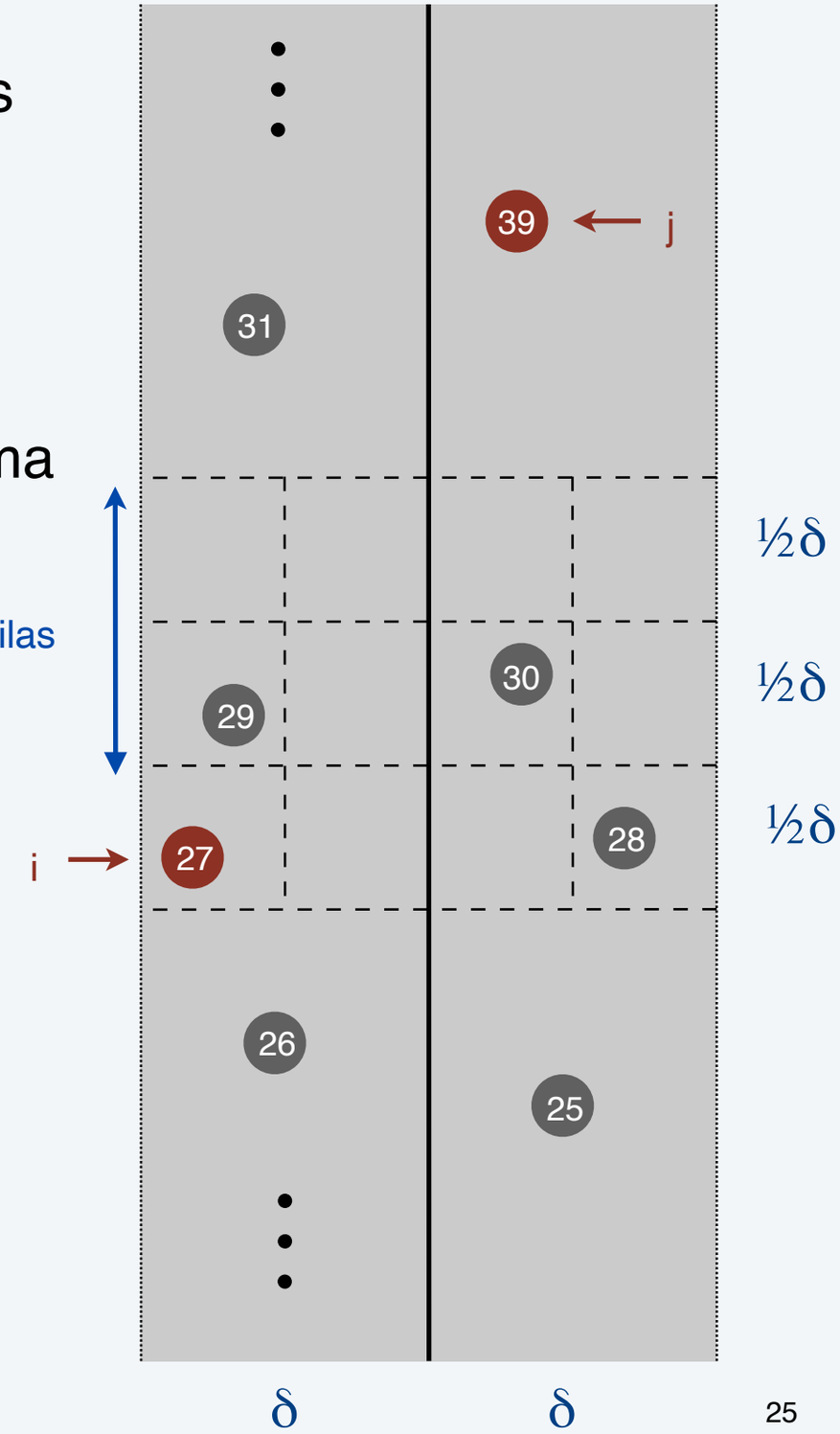
¿Cómo encontramos el par más cercano con un punto en cada lado?

Def. Sea s_i el punto de la franja $2\,\delta$, con la *i-ésima* menor coordenada y.

Prop. Si $|i - j| \geq 12$, entonces la distancia entre s_i y s_j es mayor o igual δ.

Dem.

- No pueden haber dos puntos que caigan en la misma caja de tamaño ½ δ-por-½ δ. Sino habrían dos puntos a menor distancia que δ en algún plano.
- Dos puntos separados por al menos 2 filas tienen distancia ≥ 2 (½ δ). ▪



Par de puntos más cercanos: algoritmo divide-y-vencerás

PAR-MÁS-CERCANO $(p_1, p_2, \ldots, p_n)$

Computar la línea vertical de separación L tal que la mitad de los puntos esté de cada lado. ← $O(n \log n)$

$\delta_1 \leftarrow$ PAR-MÁS-CERCANO (puntos en la mitad izquierda).

$\delta_2 \leftarrow$ PAR-MÁS-CERCANO (puntos en la mitad derecha). ← $2\,T(n / 2)$

$\delta \leftarrow \min \{ \delta_1 , \delta_2 \}$.

Eliminar los puntos a más de δ de la línea L. ← $O(n)$

Ordenar los puntos restantes por su coordenada y. ← $O(n \log n)$

Procesar los puntos en el orden de y y comparar la distancia entre cada punto y sus siguientes 11 vecinos. Si alguna de esas distancias es menor que δ, actualizar δ. ← $O(n)$

RETURN δ.

Par de puntos más cercanos: análisis

Teorema. El algoritmo de divide-y-vencerás para encontrar el par de puntos más cercanos en el plano puede ser implementado en tiempo $O(n \log^2 n)$.

$$T(n) = \begin{cases} \Theta(1) & \text{si } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + O(n \log n) & \text{en otro caso} \end{cases}$$

Algoritmo del par más cercano mejorado

Q. ¿Podemos mejorar a $O(n \log n)$?

A. Si. No ordenar desde cero los puntos en la franja cada vez.

- Cada llamado recursivo retorna dos listas: todos los puntos ordenados por la coordenada x, y todos los puntos ordenados por la coordenada y.
- Ordenar haciendo merge de dos listas ordenadas.

Teorema. [Shamos 1975] El algoritmo de divide-y-vencerás para encontrar el par de puntos más cercanos en un plano puede ser implementado en tiempo $O(n \log n)$.

Dem.

$$T(n) = \begin{cases} \Theta(1) & \text{si } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) & \text{en otro caso} \end{cases}$$