DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Puntos más cercanos

Análisis y Diseño de Algoritmos

Puntos más cercanos

Problema:

Dados n puntos en el plano, encontrar la pareja de puntos con la menor distancia euclídea entre ellos.
[Caso particular del "vecino más cercano" y de MST]

Aplicaciones:

Informática gráfica, visión por computador, sistemas de información geográfica (GIS)…

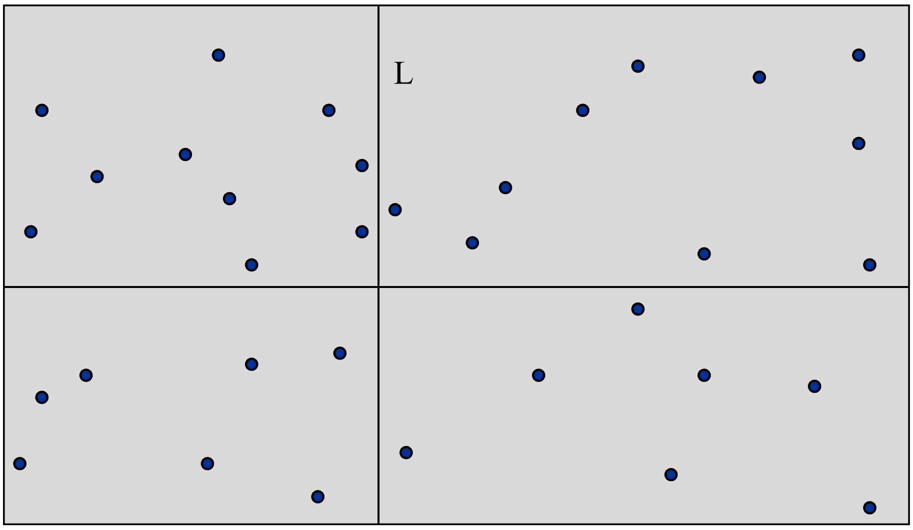
Algoritmo por fuerza bruta:

Comprobar todos los pares de puntos p y q: $\Theta(n^2)$.

- Versión 1-D: O(n log n).

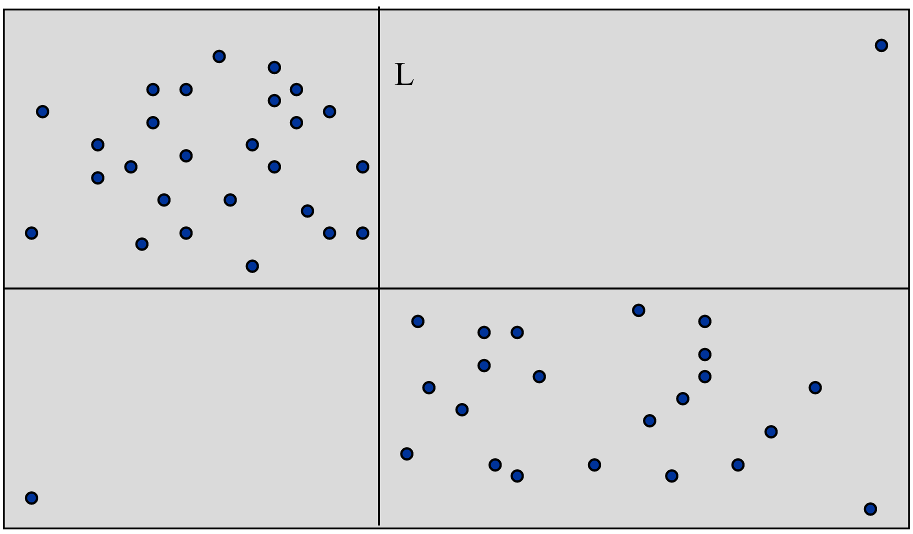
Puntos más cercanos

Algoritmo divide y vencerás:
Primer intento: División en 4 cuadrantes…



Puntos más cercanos

Algoritmo divide y vencerás:
Primer intento: División en 4 cuadrantes…
¡Resulta imposible asegurar n/4 puntos por cuadrante!

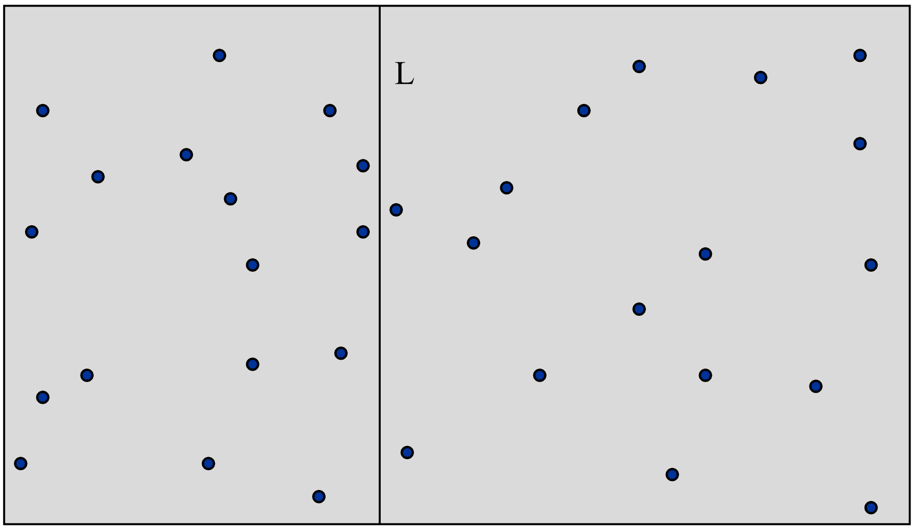


Puntos más cercanos

Algoritmo divide y vencerás:

Segundo intento:

Línea vertical con n/2 puntos a cada lado…

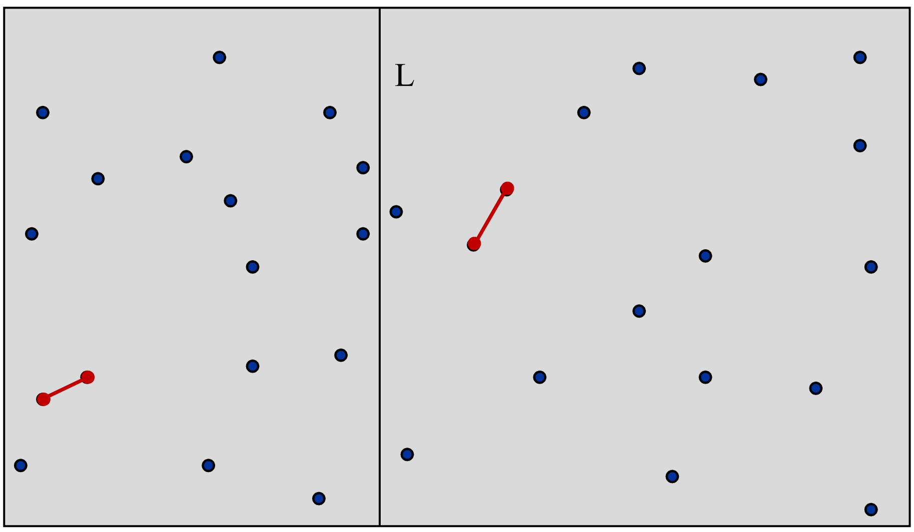


Puntos más cercanos

Algoritmo divide y vencerás:

División:

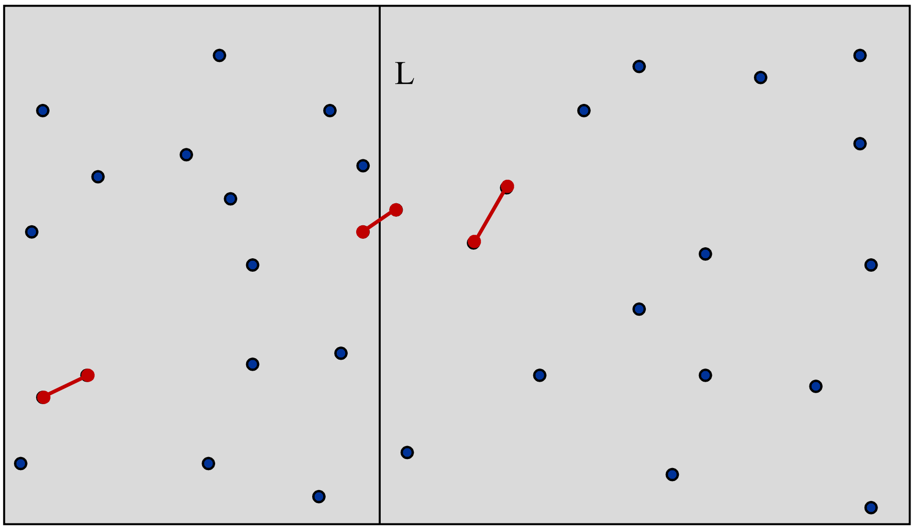
Encontrar la pareja más cercana en cada mitad…



Puntos más cercanos

Algoritmo divide y vencerás:

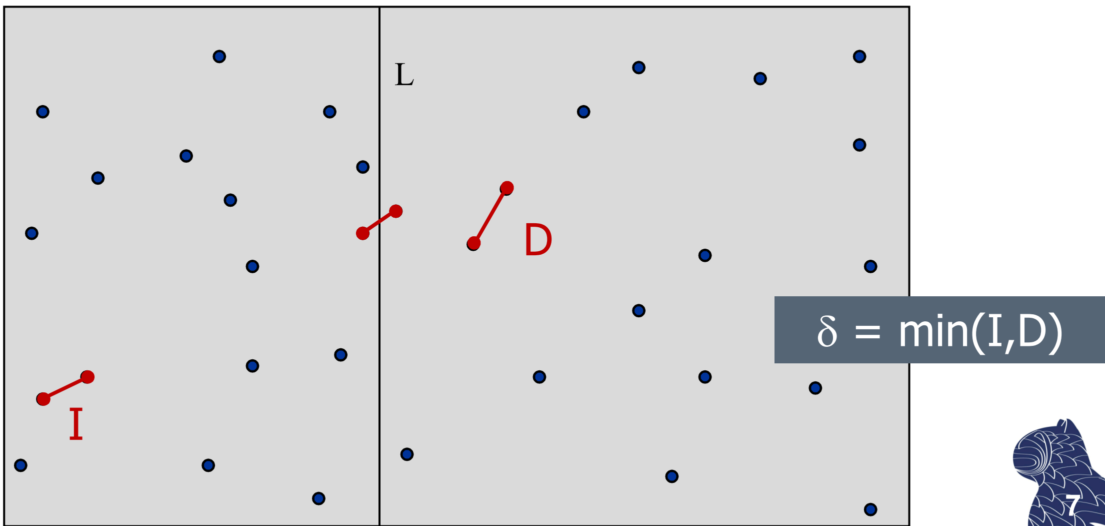
Combinación: Encontrar la pareja más cercana con un punto a cada lado de la línea…



Puntos más cercanos

Algoritmo divide y vencerás:

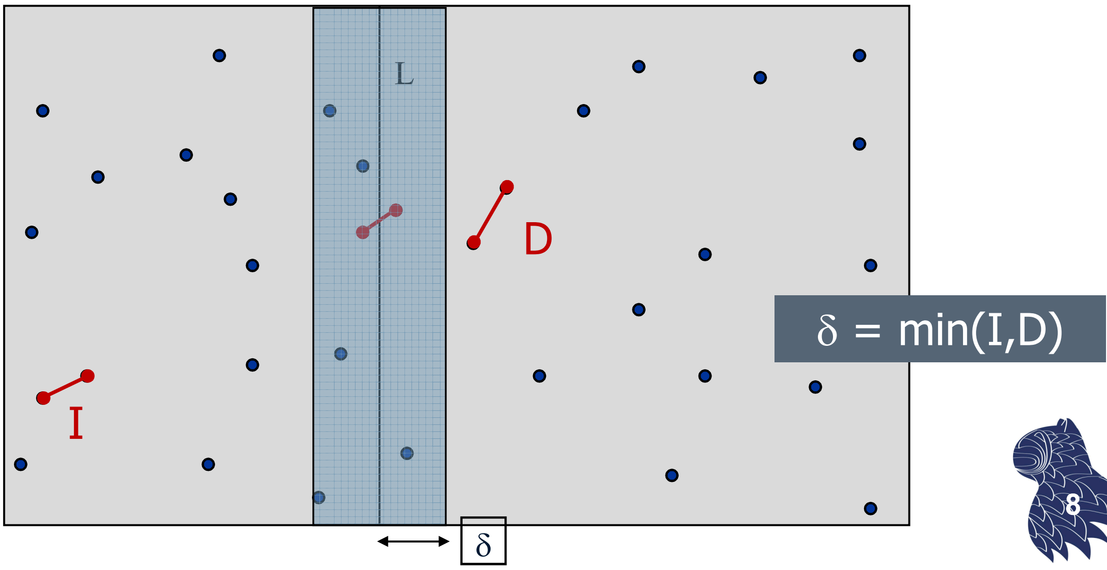
Observación: Sólo nos interesan las parejas con un punto a cada lado de la línea si están a menos de δ de **L**



Puntos más cercanos

Algoritmo divide y vencerás:

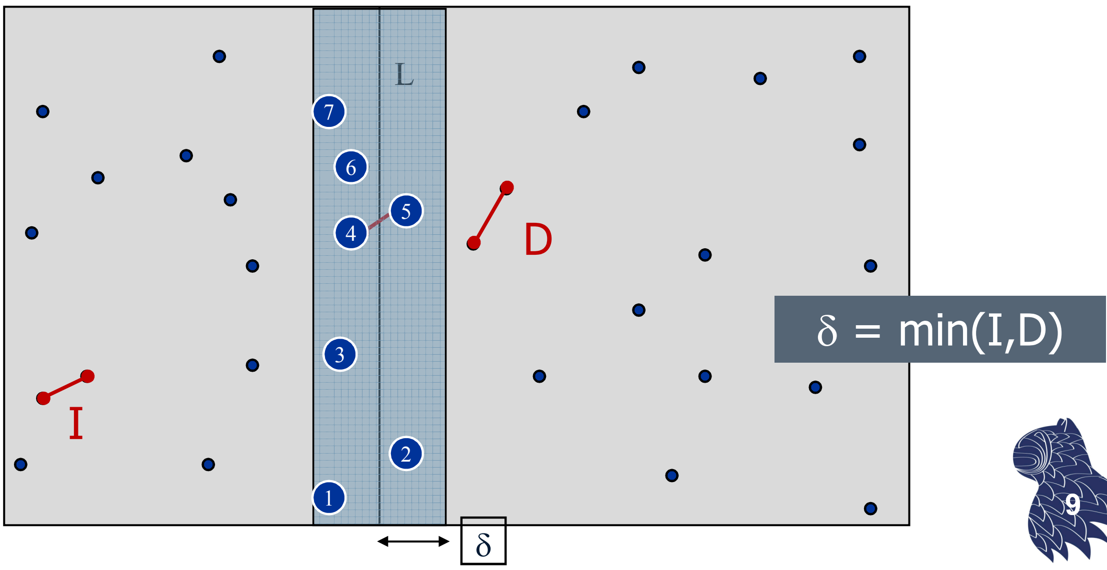
Observación: Sólo nos interesan las parejas con un punto a cada lado de la línea si están a menos de δ de **L**



Puntos más cercanos

Algoritmo divide y vencerás:

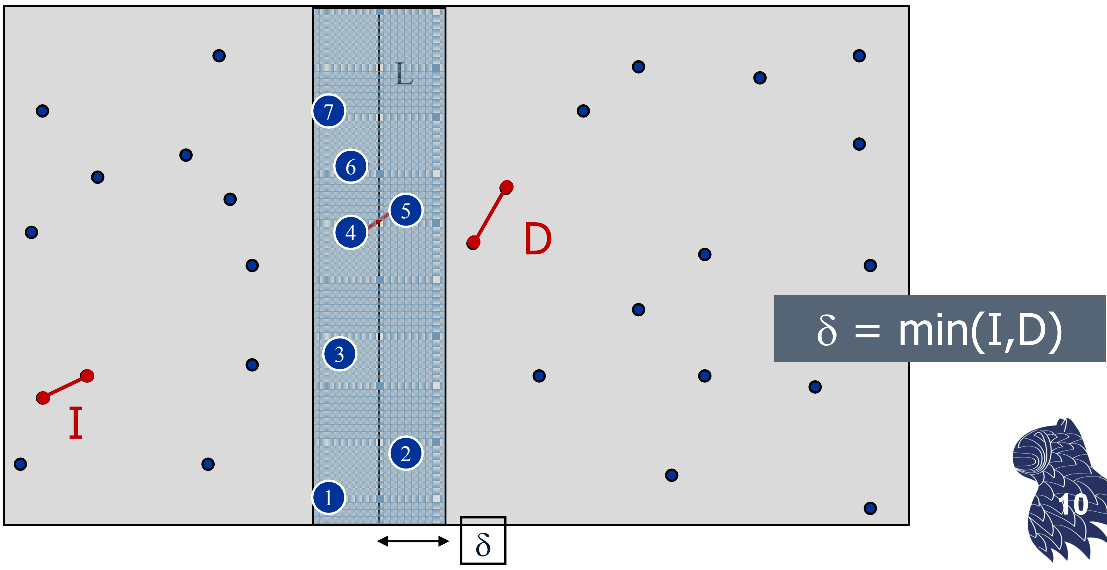
Solución eficiente: Ordenar los elementos de la franja **2**δ en función de su coordenada y...



Puntos más cercanos

Algoritmo divide y vencerás:

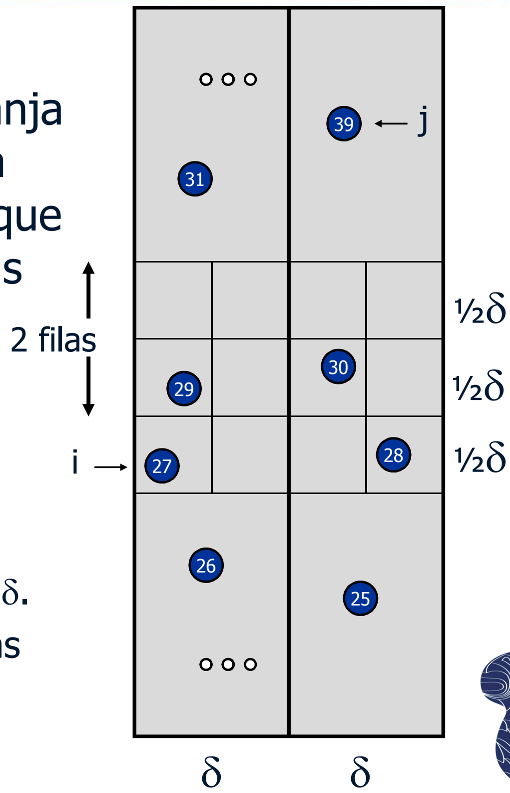
… y sólo comprobar aquéllos que están a menos de 12 posiciones en la lista ordenada en función de y.



Puntos más cercanos

Propiedad

Si s y s’ son puntos de la franja 2δ tales que d(s,s’)< δ, para encontrarlos no tendremos que calcular más de 11 distancias por punto si los ordenamos por su coordenada y.



Demostración:

- No hay dos puntos en la misma región de tamaño ½δ x ½δ.
- Dos puntos separados por dos filas están a una distancia ≥ 2(½δ).

Algoritmo

```
Closest-Pair (p1, …, pn)
{
   Compute separation line L such that half the points      O(n log n)
   are on one side and half on the other side.

   δ1 = Closest-Pair(left half)                              2T(n / 2)
   δ2 = Closest-Pair(right half)
   δ  = min(δ1, δ2)

   Delete all points further than δ from separation line L   O(n)

   Sort remaining points by y-coordinate.                    O(n log n)

   Scan points in y-order and compare distance between       O(n)
   each point and next 11 neighbors. If any of these
   distances is less than δ, update δ.

   return δ.
}
```


Eficiencia

$$T(n) \le 2T(n/2) + O(n \log n) \quad \Rightarrow \quad T(n) = O(n \log^2 n)$$

NOTA

Se puede mejorar la eficiencia del algoritmo si no se reordenan desde cero los puntos de la franja 2δ.

¿Cómo? Haciendo que cada llamada recursiva devuelva los puntos ordenados tanto por la coordenada y como por la coordenada x [y mezclando las listas ordenadas en O(n)].

$$T(n) \le 2T(n/2) + O(n) \quad \Rightarrow \quad T(n) = O(n \log n)$$