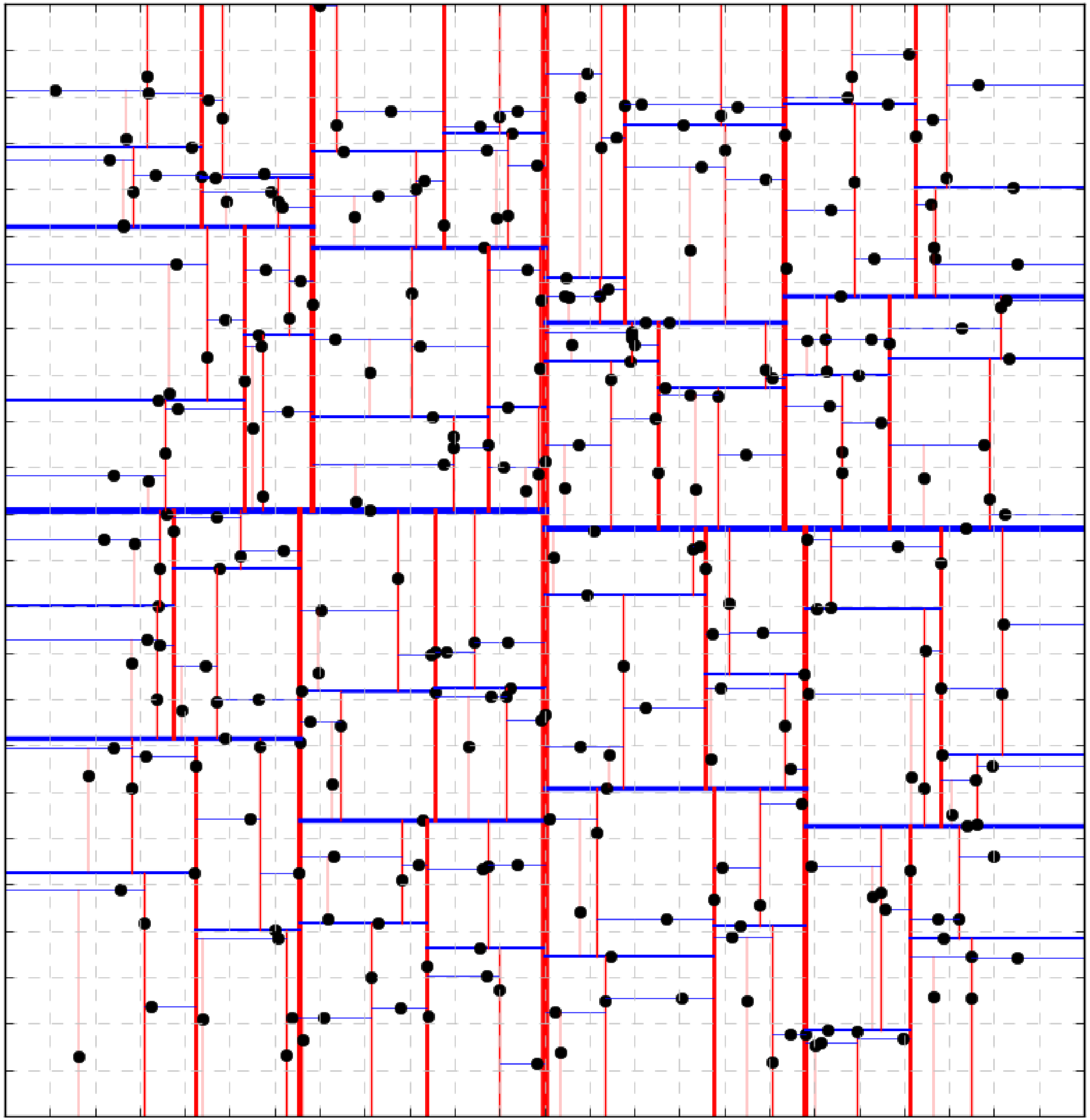
El set de coordenadas

- Se quiere procesar un set muy grande de coordenadas en 2D
- Para repartir la carga, se reparten los datos en zonas rectangulares
- La idea es que las zonas no se traslapen: deben particionar el espacio
- Queremos además que cada zona tenga la misma cantidad de datos

¿Cómo garantizamos todas estas condiciones se cumplen?



Mediana

¿Cómo encontrar la mediana de un conjunto de datos?

¿Cómo hacerlo sin recurrir a ordenar los datos?

Pensemos en la definición de mediana

Mediana

Set de datos:

7 3 5 2 1 9 7 8 6 4 3

Pivote: 6

Menores: 3 5 2 1 4 3

Mayores: 7 9 7 8

¿Cuál de los dos grupos contiene la mediana? ¿Por qué?

Mediana

Repitamos el proceso

3 5 2 1 4 3 6 7 9 7 8

Pivote: 2

Menores: 1

Mayores: 3 5 4 3

Mediana

Repitamos el proceso nuevamente

1 2 3 5 4 3 6 7 9 7 8

Pivote: 5

Menores: 3 4 3

Mayores:

Mediana

¿Como sabemos cuando encontramos la mediana?

1 2 3 4 3 5 6 7 9 7 8

$$\boldsymbol{partition(A,\ i,\ f):}$$

$$\boldsymbol{p \leftarrow un\ pivote\ aleatorio\ en\ A[i,\ f]}$$

$$\boldsymbol{m,\ M \leftarrow listas\ vacias}$$

$$\boldsymbol{for\ x \in A[i,\ f]\ excepto\ p:}$$

$$\boldsymbol{if\ x < p,\ insertar\ x\ en\ m}$$

$$\boldsymbol{else,\ insertar\ x\ en\ M}$$

$$\boldsymbol{A[i,\ f] \leftarrow concatenar\ m\ con\ p\ con\ M}$$

$$\boldsymbol{return\ i + |m|}$$

$median(A)$:

$i \leftarrow 0,\ \ f \leftarrow n - 1$

$x \leftarrow partition(A,\ i,\ f)$

$while\ x \neq \frac{n}{2}$:

$if\ x < \frac{n}{2}$, $i \leftarrow x + 1$

$else,\ \ f \leftarrow x - 1$

$x \leftarrow partition(A,\ i,\ f)$

$return\ A[x]$

Mediana

¿Cuál es la complejidad de este algoritmo?

¿En el mejor caso? ¿En el peor caso?

¿Cómo extenderlo para buscar algo en un índice arbitrario?

$\textit{quickselect}(A, j)$:

$i \leftarrow 0, \quad f \leftarrow n - 1$

$x \leftarrow \textit{partition}(A,\ i,\ f)$

$\textit{while}\ x \neq j$:

$\textit{if}\ x < j,\ \ i \leftarrow x + 1$

$\textit{else},\ \ f \leftarrow x - 1$

$x \leftarrow \textit{partition}(A,\ i,\ f)$

$\textit{return}\ A[x]$

Complejidad

¿Es la misma complejidad si hay elementos repetidos?

¿Cambia el caso promedio?

¿Cómo se puede mejorar?

Observación

En cada iteración, el pivote queda en su posición ordenada

¿Por qué?

¿Se puede usar esto para ordenar?

$\boldsymbol{quicksort(A, i, f)}$:

$\boldsymbol{if\ i \leq f}$:

$\boldsymbol{p \leftarrow partition(A,\ i,\ f)}$

$\boldsymbol{quicksort(A,\ i,\ p-1)}$

$\boldsymbol{quicksort(A,\ p+1,\ f)}$

Luego se llama $\boldsymbol{quicksort(A,\ 0,\ n-1)}$

Complejidad

¿Cual es la complejidad de QuickSort en el mejor caso?

¿Y en el peor caso?

¿Y en el caso promedio?

QuickSort en arreglos

En general, usar arreglos es más conveniente

Pero la operación de concatenar es muy cara en arreglos

Debemos reformular **partition** para que funcione en arreglos

$$\textit{partition}(A,\ i,\ f):$$

$$x \leftarrow un\ indice\ aleatorio\ en\ [i,\ f], \qquad p \leftarrow A[x]$$

$$A[x] \rightleftarrows A[f]$$

$$j \leftarrow i$$

$$\textit{for}\ k \in [i,\ f-1]:$$

$$\textit{if}\ A[k] < p:$$

$$A[j] \rightleftarrows A[k]$$

$$j \leftarrow j + 1$$

$$A[j] \rightleftarrows A[f]$$

$$\textit{return}\ j$$