Cantidad de inversiones de un arreglo

Cantidad de inversiones de un arreglo

Entrada: Un arreglo X que contiene los números $1, 2, 3, \ldots, N$ en algún orden arbitrario

Salida: Cantidad de inversiones, es decir, número de pares i,j correspondientes a índices del arreglo que cumplen que $i<j$ y $X_i > X_j$

Ejemplos: {1, 2, 3, 4} = ? 0
{4, 3, 2, 1} = ? 6

Aplicaciones: filtrado colaborativo, recomendación

¿Cuál es la cantidad máxima de inversiones en un arreglo de tamaño N? N(N-1)/2

Cantidad de inversiones de un arreglo

Primera solución: búsqueda por fuerza bruta!

```
inv = 0
for i=0 to N-2:
    for j=i+1 to N-1:
        if Xi > Xj:
            inv++
print(inv)
```

¿Cuál es la eficiencia? O(N^2)

¿Se puede hacer mejor? 

Solución mediante Divide & Vencerás

Denominar una inversión (par i, j con $i<j$) como:

- Izquierda si $i, j \leq N/2$
- Derecha si $i, j > N/2$

Se pueden resolver recursivamente

- Dividida si $i \leq N/2 < j$

Requieren otro procedimiento

Ejemplo:

{2, 1, 4, 3, 6, 5, 8, 7}

{2, 1, 4, 3}   {6, 5, 8, 7}

{2, 1} {4, 3} {6, 5} {8, 7}

Solución mediante Divide & Vencerás

```
function countInversions(X, N):
   if N ≤ 1:
      return 0
   else:
      a = countInversions(primera izquierda de X, N/2)
      b = countInversions(segunda derecha de X, N/2)
      c = countSplitInversions(X, n)
   return a + b + c
```

Objetivo: Hacer que el algoritmo tenga una eficiencia menor a $O(N^2)$, pero ¿Cómo hacerlo si el número de inversiones divididas puede ser cuadrático?

Solución mediante Divide & Vencerás

Truco: aprovecharse (descaradamente) del *MergeSort*

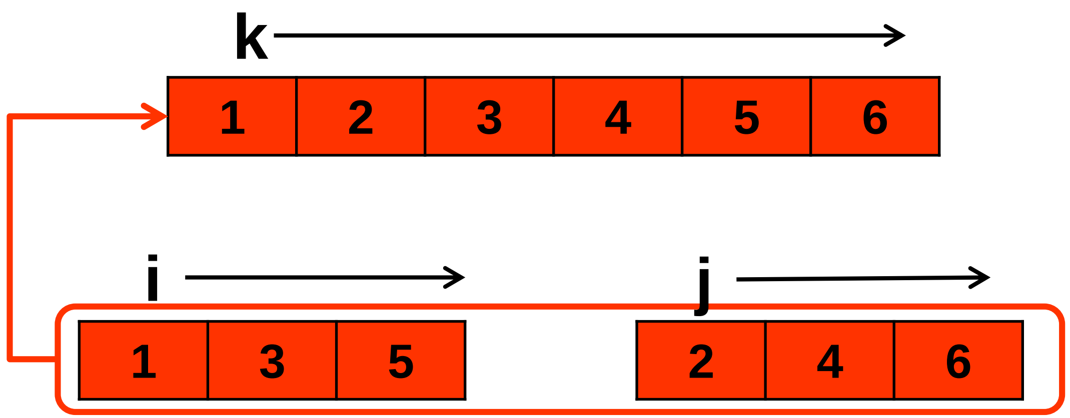
```
inv = 0
function countInversions(X, M):
   if M > 1:
      XL = mitad izquierda de X
      XR = mitad derecha de X
      return merge(countInversions(XL, M/2), countInversions(XR, M/2), M)
   else
      return X
```

¿Y que nos ganamos con esto?

¿Cómo al mezclar los dos sub-arreglos ordenados podemos contar la cantidad de inversiones?

Solución mediante Divide & Vencerás

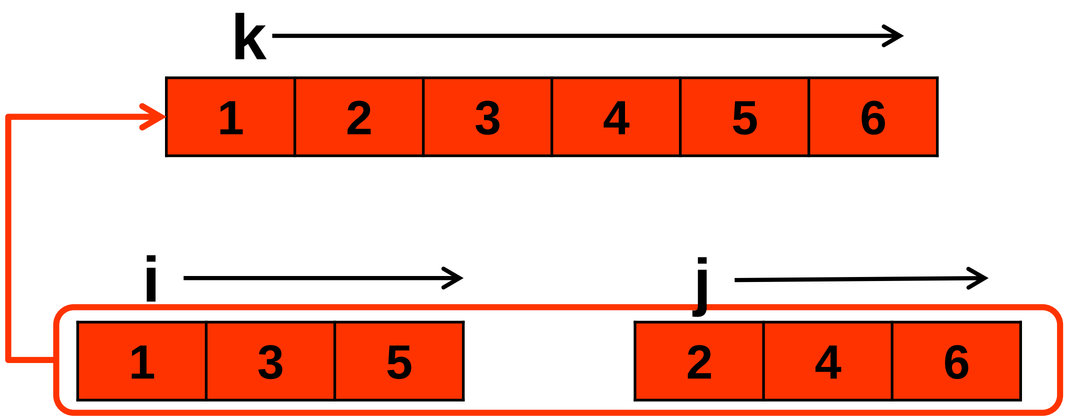
Recordemos que hace la función *merge*:



En este ejemplo, si el arreglo *XL* U *XR*, no tuviera inversiones divididas, ¿cómo sería *XL* respecto a *XR*?

Solución mediante Divide & Vencerás

Volviendo al ejemplo:



- Se copia el elemento 1 al arreglo mezclado
- Cuando el elemento 2 pasa al arreglo mezclado se “descubren” dos inversiones: (3,2) y (5,2)
- Se copia el elemento 3 al arreglo mezclado
- Cuando el elemento 4 pasa al arreglo mezclado se “descubre” una inversión: (5,4)
- Se copia el elemento 5 al arreglo mezclado
- Se copia el elemento 6 al arreglo mezclado

Idea: La cantidad de inversiones divididas que involucran a un elemento *d* del sub-arreglo *XR* es exactamente igual a la cantidad de elementos de *XL* que aún no se han revisado cuando se copia *d* al arreglo resultante.

```
Function merge(XL, XR, M){
   i,j,k = 0
   while k < M:
      if XL_i <= XR_j
         Z_k = XL_i
         i++
      else if XL_i > XR_j
         Z_k = XR_j
         j++
         inv += M/2 - i
      else if i >= M/2
         Z_k = XR_j
         j++
      else
         Z_k = XL_i
         i++
      k++
   return Z
```

¿Cuál es la eficiencia del algoritmo finalmente?

*O(N*log(N))*