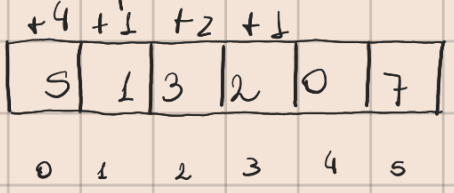
Ejercicio 7 *(DesordenSort)* ★

La cantidad de parejas en desorden de un arreglo $A[1 \dots n]$ es la cantidad de parejas de posiciones $1 \leq i < j \leq n$ tales que $A[i] > A[j]$. Dar un algoritmo que calcule la cantidad de parejas en desorden de un arreglo y cuya complejidad temporal sea estrictamente mejor que $O(n^2)$ en el peor caso. **Hint:** Considerar hacer una modificación de un algoritmo de sorting.

Puedo modificar el mergeSort para que en el caso base o en el combine cuente 1



Divido: voy dividiendo mi arreglo en 2 → 1ro 5 1 3 2do 2 0 7

5 | 1 3 2 | 0 7

5 | 1 | 3 2 | 0 | 7

5 | 1 | 3 2 | 0 | 7

+2 +1

1 3 5 | 0 2 7

+3+2

0 1 2 3 5 7

2+1+3+2

8 parejas en desorden

Quiero contarlas + eficientemente

Vamos que la # de parejas en desorden es la # de pos que tengo que moverme para acomodar el número

→ vamos a modificar el merge del mergeSort

↳ si hago tal cual el mergesort no solo voy a tener un cb que devuelve un array si no que voy a necesitar devolver un número

↳ para modificar un merge y que devuelva un número si o si le voy a tener que pasar de entrada arrays

↳ voy a hacer 2 funciones → 1 que devuelva <array, int> y otra que la llame a esta y devuelva el int

DS1 (arreglo A

if |A| = 1

return <A, 0> → ponemos 0 porque tenemos que si o si devolver una tupla porque es lo que vamos a devolver con el merge → necesitamos que coincida

else

A1 ← DS1(A[0, ..., L|/2|]) → además un array de 1 elemento no tiene pares desordenados

A2 ← DS1 (A[⌈|/2⌉, ..., |A|])

return mergeModificado (π_1 A1, π_1 A2)

DS (A)

return π_2 DS1 (A)

mergeModificado (arreglo A, arreglo B)

res = < ∅, 0)

voy mergeando A con B y a medida que meto elementos sumo 1

return res