Programación II: Guia 03

Prof. Dr. Hans H. Ccacyahuillca Bejar, INFO-UNSAAC 18/09/2023

Ejercicio 1

Implementar el algoritmo de Bubblesort (Complejidad $O(N^2)$):

```
procedimiento DeLaBurbuja (a_0, a_1, a_2, ..., a_(n-1))
  para i ← 1 hasta n hacer
    para j ← 0 hasta n − 2 hacer
      si a_(j) > a_(j+1) entonces
        aux ← a_(j)
        a_(j) ← a_(j+1)
        a_(j+1) ← aux
      fin si
    fin para
  fin para
fin procedimiento
```

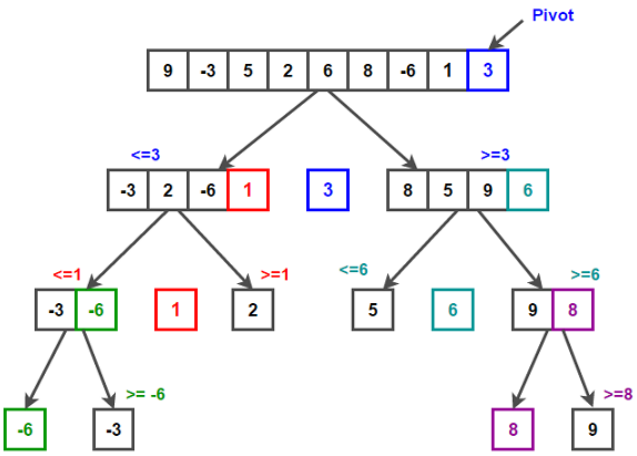
Ejercicio 2

Implementar el algoritmo de Mergesort (Complejidad $O(NlogN)$):.
El algoritmo tiene 2 fases:

- Partición: Se divide sucesivamente los elementos a ordenar en listas más pequeñas hasta llegar listas de un solo elemento.
- Se comienzan a mezclar las listas de elementos para formar listas ordenadas de 2 elementos, luego se mezclan las listas de 2 elementos para formar listas de 4 elementos ordenados y asi sucesivamente hasta terminar.

Ejercicio 3

Implementar el algoritmo de Quicksort (Complejidad $O(N^2)$)::



Ejercicio 4

Usar la función Random para generar arreglos de 10^2, 10^3 y 10^4, luego ordenarlos en forma ascendente y medir el tiempo en milisegundos que demora cada algoritmo.

Listing 1: Sample CSharp code – Random function.

```
1  Random random = new Random();
2  int[] values = new int[count];
3      for (int i = 0; i < count; ++i)
4      values[i] = random.Next(count);
```

Listing 2: Sample CSharp code – time function.

```
1  var watch = System.Diagnostics.Stopwatch.StartNew();
2  // colocar el codigo del algoritmo
3  watch.Stop();
4  Console.WriteLine($"Execution Time: {watch.ElapsedMilliseconds} ms");
```

Ejercicio 5

Se le proporcionan dos arreglos de números enteros **nums1** y **nums2**, ordenados ascendentemente, y dos números enteros **m** y **n**, que representan el número de elementos en nums1 y **nums2** respectivamente.

Fusione **nums1** y **nums2** en un único arreglo ordenado ascendentemente.

La función no debe devolver arreglo ordenado final, sino que debe almacenarse dentro de la matriz **nums1**. Para dar cabida a esto, nums1 tiene una longitud de $m+n$, donde los primeros m elementos denotan los elementos que deben fusionarse, y los últimos n elementos se establecen en **0** y deben ignorarse. **nums2** tiene una longitud de **n**.

Ejemplo 1:

Entrada: $nums1 = [1,2,3,0,0,0]$, $m = 3$, $nums2 = [2,5,6]$, $n = 3$

Salida: $[1,2,2,3,5,6]$

Explicación: Las matrices que estamos fusionando son [1,2,3] y [2,5,6].
El resultado de la fusión es [1,2,2,3,5,6] con los elementos de nums1.

Ejemplo 2:

Entrada: $nums1 = [1]$, $m = 1$, $nums2 = []$, $n = 0$

Salida: $[1]$

Explicación: Las matrices que estamos fusionando son $[1]$ y [].
El resultado de la fusión es $[1]$.

Ejemplo 3:

Entrada: $nums1 = [0]$, $m = 0$, $nums2 = [1]$, $n = 1$

Salida: $[1]$

Explicación: Las matrices que estamos fusionando son [] y $[1]$.
El resultado de la fusión es $[1]$. Tenga en cuenta que como $m = 0$, no hay elementos en nums1. El 0 solo está ahí para garantizar que el resultado de la fusión quepa en nums1.

Restricciones:

- $nums1.length == m + n$
- $nums2.length == n$
- $0 <= m, n <= 200$
- $1 <= m + n <= 200$
- $-109 <= nums1[i], nums2[j] <= 109$