Ciclo de competencia: 2023 – 2024

Hoja de ejercicios

Tema: **Insertion sort, merge sort, quick sort**

Elaborador: Cristian Israel Donato Flores

Fecha: 30/03/2023

1. Escriba la complejidad en el tiempo de los siguientes algoritmos de ordenación:
   - Insert Sort:
   - Merge Sort:
   - Quick Sort:
2. Escriba el código del algoritmo de ordenación de *Insert Sort* en el lenguaje de su preferencia. Puede utilizar el siguiente seudocódigo como apoyo.

```
funcion insertionSort(lista)
   para i de 1 a longitud(lista) hacer
      valorActual = lista[i]
      j = i - 1
      mientras j >= 0 y lista[j] > valorActual hacer
         lista[j+1] = lista[j]
         j = j - 1
      fin mientras
      lista[j+1] = valorActual
   fin para
   retornar lista
fin funcion
```

3. Escriba el código del algoritmo de ordenación de *Merge Sort* en el lenguaje de su preferencia. Puede utilizar el siguiente seudocódigo como apoyo.

```
funcion mergeSort(lista)
   si longitud(lista) > 1 entonces
      mitad = longitud(lista) / 2
      izquierda = sublista(lista, 0, mitad)
      derecha = sublista(lista, mitad, longitud(lista))

      izquierda = mergeSort(izquierda)
      derecha = mergeSort(derecha)

      retornar fusionar(izquierda, derecha)
   sino
      retornar lista
   fin si
fin funcion
```

```
funcion fusionar(izquierda, derecha)
   resultado = []
   mientras longitud(izquierda) > 0 y longitud(derecha) > 0 hacer
      si primero(izquierda) < primero(derecha) entonces
         resultado.agregar(primero(izquierda))
         izquierda = sublista(izquierda, 1, longitud(izquierda))
      sino
         resultado.agregar(primero(derecha))
         derecha = sublista(derecha, 1, longitud(derecha))
      fin si
   fin mientras

   si longitud(izquierda) > 0 entonces
      resultado.extend(izquierda)
   sino
      resultado.extend(derecha)
   fin si

   retornar resultado
fin funcion
```

4. Escriba el código del algoritmo de ordenación de *Quick Sort* en el lenguaje de su preferencia. Puede utilizar el siguiente seudocódigo como apoyo.

```
funcion quickSort(lista, inicio, fin)
   si inicio < fin entonces
      indicePivote = particion(lista, inicio, fin)
      quickSort(lista, inicio, indicePivote-1)
      quickSort(lista, indicePivote+1, fin)
   fin si
   retornar lista
fin funcion

funcion particion(lista, inicio, fin)
   pivote = lista[fin]
   i = inicio
   para j de inicio a fin-1 hacer
      si lista[j] <= pivote entonces
         intercambiar(lista, i, j)
         i = i + 1
      fin si
   fin para
   intercambiar(lista, i, fin)
   retornar i
fin funcion

funcion intercambiar(lista, i, j)
   temp = lista[i]
   lista[i] = lista[j]
   lista[j] = temp
fin funcion
```