UNIVERSIDAD SIGLO 21

TRABAJO PRÁCTICO

Estudiante: Daiana Micaela Arena

Cursada: EDH

DNI: 38629115

Consigna:

Los métodos a desarrollar son:

- el método ordenar de la clase QuickSort

```
void ordenar(int arr[], int low, int high)
```

- y el método ordenar de la clase MergeSort

```
void ordenar(int arr[], int l, int r)
```

Resolución:

Los algoritmos de ordenamiento son herramientas fundamentales en la informática y la ciencia de la computación. Dos de los métodos más eficientes y ampliamente utilizados son Merge Sort y Quick Sort. A continuación, exploraremos brevemente cómo funcionan estos algoritmos.

Para la resolución del trabajo práctico N°4 trabajé en mi editor de Visual Studio Code con las extensiones necesarias para ejecutar archivos .java

Dentro de mi directorio, cree un archivo App.java que cuenta con las siguientes funciones (obtenidas del archivo del TP4):

```java
public static void main(String args[])
{

    int arr[] = { 12, 11, 13, 5, 6, 7 };
    System.out.println(x:"\nArray original antes de MergeSort");
    MostrarArray(arr);
    MergeSort obMS = new MergeSort();
    obMS.ordenar(arr, l:0, arr.length - 1);
    System.out.println(x:"\nArray ordenado por Merge Sort");
    MostrarArray(arr);
    int arr2[] = { 12, 11, 13, 5, 6, 7 };
    System.out.println(x:"\nArray original antes de QuickSort");
    MostrarArray(arr2);
    int n = arr2.length;
    QuickSort obQS = new QuickSort();
    obQS.ordenar(arr2, low:0, n-1);
    System.out.println(x:"\nArray ordenado por Quick Sort");
    MostrarArray(arr2);
}
```

```java
/* Una función que sirve para mostrar un array de tamaño n */
static void MostrarArray(int arr[])
{
    int n = arr.length;
    for (int i = 0; i < n; ++i)

    System.out.print(arr[i] + " ");
}
// Main
```

En cuanto a los archivos pedidos, me basé en la estructura del módelo del TP4 que se encuentra en el canvas, añadiendo comentarios para hacerlo más legible. Adjunto a continuación las funciones pedidas:

ORDENAR - MERGESORT:

- **Descripción:** Merge Sort es un algoritmo de ordenamiento dividir y conquistar. Divide el arreglo en mitades iguales, ordena cada mitad y luego combina las mitades ordenadas en un solo arreglo ordenado.
- **Eficiencia:** Tiene un tiempo de ejecución promedio de O(n log n), lo que lo hace eficiente para grandes conjuntos de datos.

```java
// Función principal que ordena arr[l..r] utilizando el método mezclar()
public void ordenar(int arr[], int l, int r){

    // Verifica si hay más de un elemento en el arreglo
    if (l < r) {

        // Encuentra el punto medio del arreglo
        int m = l + (r - l) / 2;

        // Ordena la primera mitad del arreglo
        ordenar(arr, l, m);

        // Ordena la segunda mitad del arreglo
        ordenar(arr, m + 1, r);

        // Une las dos mitades ordenadas
        mezclar(arr, l, m, r);
    }
}
```

ORDENAR - QUICKSORT:

- **Descripción:** Quicksort también es un algoritmo de ordenamiento dividir y conquistar. Selecciona un "pivote," coloca los elementos menores que el pivote a la izquierda y los elementos mayores a la derecha. Luego, ordena recursivamente las subpartes izquierda y derecha.

- **Eficiencia:** Tiene un tiempo de ejecución promedio de O(n log n) y puede ser más rápido que Merge Sort en ciertos casos debido a su menor sobrecarga.

```java
/* Método principal que implementa quicksort
arr[] ⟶ Array a ser ordenado,
low ⟶ Comienzo de indice,
high ⟶ Fin de indice */
public void ordenar(int arr[], int low, int high){
    if (low < high) {
        // Encuentra el índice de partición
        int partitionIndex = particion(arr, low, high);

        // Ordena la parte izquierda del arreglo (elementos menores que el pivote)
        ordenar(arr, low, partitionIndex - 1);

        // Ordena la parte derecha del arreglo (elementos mayores que el pivote)
        ordenar(arr, partitionIndex + 1, high);
    }
}
```

Adjunto también un screen de la consola donde podemos validar que ejecuta el código correctamente:

```
PS C:\Users\daian\Desktop\codiguitos\java-algorithms-data-structures>  c:; cd 'c:\Users\daian\Desktop\codiguitos\java-algorithms-data-structures'; & 'C:\Program Files\Java\jdk-20\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\daian\Desktop\codiguitos\java-algorithms-data-structures\bin' 'Exercises.TP4.App'

Array original antes de MergeSort
12 11 13 5 6 7
Array ordenado por Merge Sort
5 6 7 11 12 13
Array original antes de QuickSort
12 11 13 5 6 7
Array ordenado por Quick Sort
5 6 7 11 12 13
PS C:\Users\daian\Desktop\codiguitos\java-algorithms-data-structures>
```

Conclusión:

Ambos algoritmos son altamente eficientes y ampliamente utilizados en la práctica. La elección entre Merge Sort y Quick Sort depende de las características específicas del problema a resolver y las preferencias del programador.

Muchas gracias por leer hasta aquí. Todo lo expuesto puede encontrarse en los archivo adjuntos en esta entrega. ¡Saludos!