ESTRUCTURAS DE DATOS Y ALGORITMOS II

PRACTICA # 3: ALGORITMOS DE ORDENAMIENTO PARTE 3

Objetivo: El estudiante identificará la estructura de los algoritmos de ordenamiento MergeSort, Counting-sort y Radix-sort

Objetivo de clase: El alumno implementará casos particulares de estos algoritmos para entender mejor su funcionamiento a nivel algorítmico.

ACTIVIDADES PARA EL DESARROLLO DE LA PRÁCTICA

Ejercicio 1.- Merge Sort.- Responde las siguientes preguntas acerca de MergeSort

1.- Qué diferencias hay entre el algoritmo de merge sort proporcionado y el pseudo código visto en clase

2.- En segundo archivo (Practica3.java) escribe la función principal del programa para crear un arreglo de 20 elementos y ordenarlo con mergesort (agrega aquí una captura del código y de la ejecución)

3.- Para qué sirven n1 y n2

4.- Para qué sirven los arreglos L y R dentro de Merge Sort (agrega en el código instrucciones para imprimir los arreglos L y R en cada iteración para ayudar a responder esta pregunta)

5.- Consideras adecuado que los métodos de MergeSort sean estáticos, (si o no y por qué?)

6.- Dónde consideras que inicia y termina una “Iteración” de MergeSort. Agrega en tu programa impresiones de esas “iteraciones”

7.- Agrega a la función principal las instrucciones para que el arreglo se llene con valores aleatorios de 1 a 100

Ejercicio 2. CountingSort

En el lenguaje de tu preferencia (C o Java), realiza la implementación de Counting Sort aplicado a un caso específico. Toma en cuenta los siguientes requerimientos:

1) Utiliza un arreglo de 20 elementos, los cuales serán solicitados al usuario (asume que el usuario los va a ingresar correctamente)

Para ello considera solo números enteros del 5 al 20

2) Crea un segundo arreglo donde cada posición será asociada a uno de los posibles valores del rango indicado (arreglo para hacer la cuenta)

3) En una primera pasada al arreglo que llenó el usuario, realiza la cuenta de las apariciones de cada uno de los valores. (Muestra en pantalla el arreglo de la cuenta al finalizar la primera pasada)

4) Realiza la suma de los elementos del arreglo; en cada índice se considera la cantidad de elementos actuales y los anteriores. (Muestra en pantalla la suma del arreglo)

5) Ingresa en un tercer arreglo los elementos ordenados de acuerdo con el funcionamiento del algoritmo, realizando una segunda pasada al primer arreglo, partiendo desde el final y para cada elemento, verifica la posición que le corresponde en el segundo arreglo y establece su posición final en el tercero.

Para verificar el funcionamiento y los pasos del algoritmo, agrega impresiones en pantalla en e acomodo del arreglo final ordenado

Ejercicio 3. RadixSort

En el lenguaje de tu preferencia (C o Java). Realiza la implementación de radix-sort basada en un caso particular similar al visto en clase, con las siguientes restricciones:

1) Los datos de entrada serán números de longitud máxima de 4 , de entre los cuales solo aparecerán dígitos del 0 al 4

2) Deberás implementar una cola para cada uno de estos dígitos (0,1,2,3,4)

3) Se deberán solicitar al usuario los valores a ordenar (15 elementos), los cuales podrás almacenar en un arreglo o una lista (ejemplo de entrada: 4203, 0313, 2302, 1031, 2021, 1143, etc.)

4) Es necesario realizar 4 iteraciones sobre esta lista o arreglo, y en cada una de ellas utilizar las colas para almacenar los elementos de acuerdo con su posición (unidades, decenas, etc). El programa deberá mostrar la lista resultante en cada iteración

5) Al finalizar, el programa deberá mostrar la lista ordenada

CONCLUSIONES

Escribe las conclusiones de tu práctica en la cual, incluye comentarios generales de la implementación de estos casos particulares