ALGORITMOS DE ORDENAMIENTO

DESCRIPCIÓN

Durante la sesión, de manera individual, deberás desarrollar una aplicación que aplica el uso de apuntadores en la implementación de algunos algoritmos de ordenamiento.

MODALIDAD

Individual

INSTRUCCIONES

A patir del archivo ***sorts.h***, implementa los algoritmos de ordenamiento que se describen a continuación.

int* count_sort(int *arr, int size)

Devuelve un nuevo arreglo con el resultado de efectuar el algoritmo de conteo sobre ***arr***. La función no debe modificar el arreglo ***arr***.

Descripción del algoritmo: La idea básica es que para cada elemento ***arr[i]*** en el arreglo ***arr***, debemos contar el número de elementos en el arreglo que son menores a ***arr[i]*** (***count***). Entonces colocamos ***arr[i]*** en un arreglo temporal usando como índice (posición) el valor de ***count***. Este algoritmo presenta un pequeño problema cuando tratamos con arreglos que tienen valores repetidos, dado que pueden ser asignados a la misma localidad en el arreglo temporal. Una forma fácil de resolver este punto es revisar la posición de los valores cuando estos son iguales. Si ***arr[i] == arr[j]*** y ***j < i***, entonces incrementamos el contador.

Liga: https://en.wikipedia.org/wiki/Sorting_algorithm#Counting_sort

int* bucket_sort(int *arr, int size)

Devuelve un nuevo arreglo con el resultado de efectuar el algoritmo de cubetas sobre ***arr***. La función no debe modificar el arreglo ***arr***.

Descripción del algoritmo: Empieza con diez arreglos vacíos numerados de 0 a 9. Recorremos todos los elementos de ***arr***, colocando cada elementos en el arreglo que le corresponde: los elementos del 0 al 9 van en el arreglo 0, los elementos del 10 al 19 van en el arreglo 1, y así sucesivamente. Posteriormente ordena individualmente cada uno de los arreglos (puedes utilizar algunos de los otros algoritmos de ordenamientos que se van a implementar). Finamente regresa el resultado de concatenar en orden los arreglos del 0 al 9.

Liga: https://en.wikipedia.org/wiki/Sorting_algorithm#Bucket_sort

int* merge_sort(int *arr, int size)

Devuelve un nuevo arreglo con el resultado de efectuar el algoritmo de mezcla sobre ***`arr`***. La función no debe modificar el arreglo ***`arr`***.

Descripción del algoritmo: Divide el arreglo arr en n subarreglos de 1 elementos (un arreglo de un elemento se considera ordenado. Repetidamente mezcla dos subarreglos ordenados para generar un nuevo subarreglo. Este último paso de repite hasta tener el arreglo final ordenado.

Liga: https://en.wikipedia.org/wiki/Merge_sort