MÉTODOS DE ORDENAMIENTO

Belén Miranda Y Andrés Monestel

QUICK SORT

Elegir un elemento de la lista, ordenar a cada lado del elemento elegido los demás elementos restantes, de modo que quede los menores a este a un lado y los mayores al otro. Repetir este proceso con cada sublista de los menores y los mayores de forma recursiva mientras las listas aun contenga más de un elemento.

BIN SORT

Consiste en crear sublistas con rangos, para dividir los dígitos de la lista principal en estas y colocarlos en la sublista en donde cumplan el rango solicitado. Posteriormente, ordenar cada sublista y luego unirlas.

RADIX SORT

Se empieza a comparar el dígito menos significativo(unidades) y se acomoda la lista de elementos en base en el dígito de esa posición, luego se cambia al siguiente dígito(decenas) y se vuelve a acomodar la lista en base a ese dígito y así consecutivamente hasta realizar el proceso con todos los dígitos.

HEAP SORT

Consiste almacenar todos los dígitos de izquierda a derecha en una estructura tipo genealógica como un "árbol", y en el que cada dígito tendrá dos "hijos" máximo, se tiene que comparar que cada dígito "padre" sea mayor/menor a sus hijos, si no es así remplazar el dígito "hijo" mayor/menor por el dígito "padre", de esta manera comparar los demás dígitos "padres" hasta tener al dígito "padre" principal que se colocara en la última posición de la lista y ahora el nuevo dígito "padre" principal será el último digito "hijo" de derecha a izquierda. Realizar consecutivamente hasta no tener más dígitos en el "árbol".

MELGE SORT

Se divide la lista por la mitad una y otra vez formando sublistas, hasta que cada sublista contenga un solo elemento de longitud. Luego esos elementos se vuelven a unir, se comparan los primeros dígitos de cada lista para conocer cual es el menor, se agrega el menor en la nueva lista y se aumenta el índice de la sublista de donde proviene el menor para así seguir comparando con los dígitos restantes.

BURBUJA

Se realizan múltiples pasadas por la lista comparando el primer dígito desde el segundo dígito hasta el último para encontrar el mayor, mientras sea mayor a los dígitos se colocará en la posición de estos, en el momento en que este sea menor a un digito con el que se comparó el nuevo dígito a comparar con lo que resta de la lista será el mayor.

¿CUÁL ES MEJOR?¿POR QUÉ?

Segun nuestra investigacion, se sugiere que los mejores metodos de ordenamiento son quickSort, mergeSort y heapSort, desde nuestro punto de vista el quickSort es mejor debido a que tiene menos limitaciones, por ejemplo el quickSort puede ordenar datos reales y no solo enteros, ademas de que requiere de menos espacio en memoria que los demas como lo es el mergeSort el cual requiere el doble de espacio que un quickSort.