Práctica 9. Algoritmos de Ordenación y Divide y Vencerás

Objetivos

- Practicar el diseño e implementación de algoritmos de ordenación.
- Realizar un estudio comparativo de algoritmos de ordenación.
- Practicar el diseño e implementación de algoritmos bajo el esquema de Divide y Vencerás.

Desarrollo

Se parte de una lista de identificadores de productos, los cuales vienen representados por códigos numéricos positivos. Se pide:

1. Diseñar e Implementar un Algoritmo de Ordenación

Implementar un algoritmo de ordenación por base (radix sort) que ordene la lista de códigos de producto. El número de dígitos que forman el código del producto debería ser un valor parametrizable. Por ejemplo, estableciendo tres dígitos tendríamos códigos de 000 a 999, con cuatro de 0000 a 9999, etc.

2. Realizar un Estudio Comparativo

Una vez implementado el algoritmo anterior, se pretende realizar un estudio comparativo de nuestro radix sort respecto a la ordenación por quicksort proporcionada por java con el método Arrays.Sort(int[] a).

Para ello habrá que ejecutar ambos algoritmos en distintas situaciones, variando dos parámetros: el tamaño del array y el número de dígitos de los códigos de productos. Para facilitar este proceso, se proporciona un pequeño fragmento de código con las instrucciones para medir tiempos y una función que permite generar arrays aleatorios de códigos de productos, dado el rango y tamaño del array.

3. Diseñar e Implementar un Algoritmo con el Esquema Divide y Vencerás

Partiendo del array de códigos de producto ya ordenado, se pretende implementar un algoritmo que siga el esquema Divide y Vencerás y que encuentre la coincidencia índice-valor.

La coincidencia índice-valor es aquel elemento cuyo valor numérico es igual al índice que ocupa dentro del array. Por ejemplo, en el siguiente caso la coincidencia índice-valor la tendríamos en el elemento 5:

0	1	2	3	4	5	6	7	8	9
-10	-8	-3	-1	3	5	13	23	43	123

En caso de no encontrar la coincidencia índice-valor exacta, devolveremos el primer valor que supere el valor del índice.

Sin embargo, en nuestros vectores de códigos de productos no tenemos valores negativos y en caso de aplicar este algoritmo, la coincidencia índice-valor la tendríamos en el primer elemento. Para solventar esto introduciremos un desplazamiento, que será introducido como parámetro y se le sumará a los índices.

Por ejemplo, en el siguiente caso la coincidencia índice-valor sin considerar desplazamiento la tendríamos en el elemento 0, que aunque no coincide su valor con el índice, sí que lo supera.

0	1	2	3	4	5	6	7	8	9
10	20	30	40	55	86	99	120	143	237

Los siguientes ejemplos consideran desplazamiento. Considerando un desplazamiento 20, la coincidencia exacta no la tendríamos, pero el algoritmo devolvería el elemento 2, ya que su valor (30) es superior al del índice más el desplazamiento (2+20). Con desplazamiento 60, el algoritmo devolvería el elemento 5...