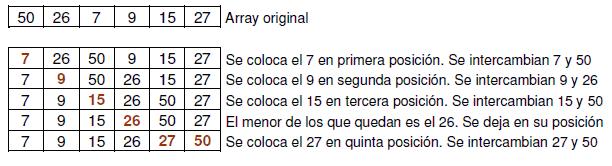
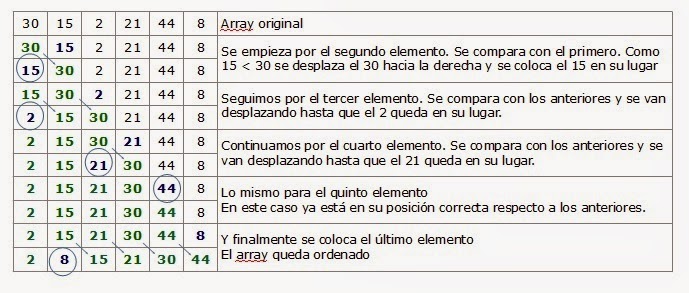
**Algoritmos de ordenamiento**

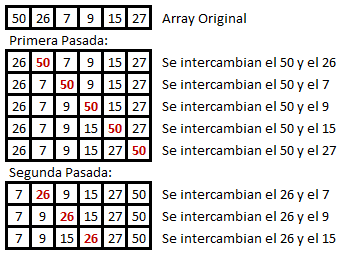
**Por selección:**Se busca el elemento más pequeño del array y se coloca en la primera posición. Entre los restantes, se busca el elemento más pequeño y se coloca en la segunda posición. Entre los restantes se busca el elemento más pequeño y se coloca en la tercera posición. Este proceso se repite hasta colocar el último elemento.

**Por Inserción:**El método de ordenación por inserción directa consiste en recorrer todo el array comenzando desde el segundo elemento hasta el final. Para cada elemento, se trata de colocarlo en el lugar correcto entre todos los elementos anteriores a él o sea entre los elementos a su izquierda en el array.

Dada una posición actual p, el algoritmo se basa en que los elementos A[0], A[1], ..., A[p-1] ya están ordenados.



**Método de la burbuja:**Consiste en comparar pares de elementos adyacentes en un array y si están desordenados intercambiarlos hasta que estén todos ordenados. Si A es el array a ordenar, se realizan A.length-1 pasadas. Si la variable i es la que cuenta el número de pasadas, en cada pasada i se comprueban los elementos adyacentes desde el primero hasta A.length-i-1 ya que el resto hasta el final del array están ya ordenados. Si los elementos adyacentes están desordenados se intercambian. Es uno de los peores algoritmos de ordenación en cuanto a tiempo de ejecución, solamente es recomendable su uso para ordenar listas con un número pequeño de elementos.



**QuickSort:**Se basa en la técnica divide y vencerás, que consiste en ir subdividiendo el array en arrays más pequeños, y ordenar éstos. Para hacer esta división, se toma un valor del array como pivote, y se mueven todos los elementos menores que este pivote a su izquierda, y los mayores a su derecha. A continuación, se aplica el mismo método a cada una de las dos partes en las que queda dividido el array.

Después de elegir el pivote se realizan dos búsquedas: Una de izquierda a derecha, buscando un elemento mayor que el pivote Otra de derecha a izquierda, buscando un elemento menor que el pivote. Cuando se han encontrado los dos elementos anteriores, se intercambian, y se sigue realizando la búsqueda hasta que las dos búsquedas se encuentran. La implementación del método de ordenación Quicksort es claramente recursiva.

La elección del pivote determinará la eficiencia de este algoritmo ya que determina la partición del array. Si consideramos que el array está desordenado, podemos elegir el primer elemento y el algoritmo funcionaría de forma eficiente. Pero si el array está casi ordenado, elegir el primer elemento como pivote sería una mala solución ya que obtendríamos un subarray muy pequeño y otro muy grande. Por la misma razón, elegir el último elemento del array como pivote también es una mala idea. Pretendemos conseguir que el tamaño de los subarrays sea lo más parecido posible.

