










## RELACIONA LAS COLUMNAS

1	El algoritmo de ordenamiento que asume que cada uno de los $n$ elementos del arreglo de entrada es un entero en el rango de 0 a $k$ . La idea es determinar, para cada elemento de entrada $x$ , el número de elementos menores que $x$ . Esta información puede ser usada para colocar al elemento $x$ directamente en su posición final en el arreglo de salida. Se asume que la entrada es un arreglo $A$ de longitud $n$ cuyos elementos son enteros no negativos.	QuickSort
2	El algoritmo de ordenamiento que es una variante del método de ordenación por selección, donde la búsqueda del elemento mínimo de un arreglo se realiza mediante técnicas basadas en la construcción de un montículo o heap. Se construye un montículo ascendente, o min heap, por lo tanto la raíz $v[1]$ será el menor elemento, y se intercambia con $v[n]$ . Luego se construye un montículo, y el menor elemento se $v[1]$ se intercambia con $v[n-1]$ , y así sucesivamente hasta que solo quede un elemento en el montículo.	ShellSort
3	El algoritmo de ordenamiento que divide en arreglo en dos subarreglos, que se pueden ordenar de modo independiente. Se selecciona un elemento específico del arreglo llamado <i>pivote</i> .	Radixsort
4	El algoritmo de ordenamiento que divide el vector por la posición central, ordena cada una de las mitades y después realiza la <i>mezcla</i> ordenada de las dos mitades. El caso base es aquel que recibe un vector con ningún elemento, o con 1 solo elemento, ya que obviamente está ordenado.	Counting sort
5	El algoritmo de ordenamiento realiza comparaciones entre elementos NO consecutivos, separados por una <i>distancia o salto</i> . El valor salto al principio es $n/2$ y va decreciendo en cada iteración hasta llegar a valer 1. Cuando salto vale 1 se comparan elementos consecutivos. El valor se encontrará ordenado cuando salto valga 1 y no se puedan intercambiar elementos consecutivos porque están en orden.	Merge Sort
6	En el algoritmo de ordenamiento, la manera de ordenar, es como un conjunto de fichas : se forman <i>montones</i> de fichas, cada uno caracterizado por tener sus componentes un mismo dígito en la misma posición. Inicialmente se forman los montones por las unidades (dígito de menor peso); estos montones se recogen y agrupan en orden ascendente, desde el montón del dígito 0 al montón del dígito 9. Entonces las fichas están ordenadas respecto a las unidades, a continuación, se vuelve a distribuir las fichas en montones, según el dígito de las decenas. El proceso de distribuir las fichas por montones y posterior acumulación en orden se repite tantas veces como número de dígitos tiene la ficha de mayor valor.	InsertionSort
7	El algoritmo de ordenamiento <i>separa la secuencia en dos grupos</i> : una parte con los valores ordenados (inicialmente con un solo elemento) y otra con los valores por ordenar (inicialmente todo el resto). Luego vamos pasando uno a uno los valores a la parte ordenada, asegurándose que se vayan colocando ascendentemente.	Heap sort
8	Esta técnica de ordenamiento implica dividir elementos en varios grupos o cubetas. En estas cubetas se distribuyen los elementos a  par, cada cubeta se ordena utilizando cualquier otro algoritmo. Finalmente, los elementos ordenados se colocan en el arreglo original de forma ordenada        	Bucket sort