Gnome Sort: Divide los datos en dos sublistas, una ordenada y la otra desordenada,

Merge Sort: usa recursión. Se verifica si el número de elementos es 1. Si lo es, se regresa el vector. De lo contrario, se dividen los datos del vector en dos mitades. A cada mitad se le aplica el mismo proceso. Al momento de hacer la regresión, ambas mitades comparan sus elementos para hacer un vector ordenado. Se verifica si el primer elemento del vector1 es mayor que el primer elemento del vector2, se añade el menor número al nuevo vector, y se comparan los siguientes 2 números menores. Se hace esto en cada regresión hasta tener la lista ordenada.

Quick Sort: usa recursión. Se verifica si el número de elementos es 1. Si lo es, se regresa el vector. De lo contrario, se toma un elemento del vector como pivote (generalmente el centro) y se compara cada número con él. Dependiendo de si es menor o mayor se asigna a un vector u otro. Al final, se vuelve a hacer el proceso con cada uno de los vectores. Finalmente, se devuelve el resultado de Quicksort(menores) + pivote + Quicksort(mayores).

Radix Sort: usa un sort auxiliar llamado counting sort. Lo que se hace es tomar el valor de las unidades de cada número, y se usa el sort de counting. Esto se repite con decenas, centenas, etc. Hasta completar todas las unidades del mayor número.

Shell Sort: Mide la longitud de la lista, (cuantos datos hay en la lista), se divide dentro de 2 para comenzar con el primer GAP.
Por ejemplo
(1,6,8,2,4,5,7) Hay 7 datos, 7 / 2 = a 3, cuenta del 0 al 3 los datos y revisa si el dato 0 y el 3 están ordenados, sino, hace el cambio, lo mismo hasta terminar con todos los datos y luego para el siguiente gap 3 / 2 = 1, los ordena de 1 en 1, en diferentes rondas hasta que quede ordenado.

Al programar, cada sort solo acepta estructuras de datos que implementen Comparable