TIPO DE ALGORITMO	QuickSort	Bidireccional	ShellSort	HeapSort
DESCRIPCIÓN	Es el algoritmo más rápido de ordenamiento, utiliza la técnica de divide y vencerás.	Va ordenando el vector por ambos extremos al mismo tiempo.	Va comparando elementos que se encuentran separados por un espacio de varias posiciones.	Almacena los elementos del vector en un “montículo” en una estructura de árbol binario (apilamiento).
VENTAJAS	• Es el algoritmo mas rápido. • No requiere de memoria adicional.	• Realiza el ordenamiento en mitad de tiempo que burbuja.	• Trabaja bien con arreglos pequeños o medios. • No requiere de memoria adicional.	• Funciona más efectivamente con datos desordenados. • No requiere de memoria adicional.
DESVENTAJAS	• Es un poco complicado de implementar. • La recursividad utiliza muchos recursos.	• Aun siendo más rápido que burbuja realiza muchas iteraciones.	• No es muy eficiente con arreglos grandes.	• Método complejo de programar.
MEJOR CASO	El pivote termina en el centro de la lista, dividiéndola en dos sublistas de igual tamaño O(n·log n)	O(n)	O(n log n)	O(n log n)
CASO PROMEDIO	O(n log n)	O(n^2)	Depende de los incrementos.	O(n log n)
PEOR CASO	El peor caso de QuickSort se produce cuando el pivote resulta ser siempre el mínimo o el máximo del conjunto. O(n^2)	O(n^2)	Depende de los incrementos	O(n log n)
ORDEN DE COMPLEJIDAD	O(n·log n)	$O(n^2)$	$O(n^2)$	O(n·log n)

TIPO DE ALGORITMO	Inserción	Burbuja	MergeSort	Selección
DESCRIPCIÓN	Consiste en ir comparando los elementos del vector desde el comienzo, a medida que avanza se ordenan los elementos del vector.	Revisa cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado.	Consiste en dividir la lista en dos mitades, ordena cada una de ellas y luego las mezcla en una nueva lista ordenada.	Busca el menor de todos los elementos del vector y lo cambia a la primera posición y así sucesivamente hasta ordenar el vector.
VENTAJAS	• Fácil implementación • Requerimientos mínimos de memoria.	• Es bastante sencillo de implementar	• Es rápido • No requiere de memoria adicional	• Es fácil de implementar • No requiere de memoria adicional.
DESVENTAJAS	• Realiza muchas comparaciones	• Es el método mas ineficiente • Requiere de muchas iteraciones	• Utiliza muchos recursos • Es complejo de implementar	• Es lento • Poco eficiente con listas grandes.
MEJOR CASO	El mejor caso ocurre cuando el arreglo esta ordenado, ya que no debe desplazar ningún elemento. O (n)	O(n)	O(n log n)	$O(n^2)$
CASO PROMEDIO	$O(n^2)$	O(n^2)	O(n log n)	$O(n^2)$
PEOR CASO	Insertar siempre en la primera posición ; entrada en orden inverso.	O(n^2)	O(n log n)	$O(n^2)$
ORDEN DE COMPLEJIDAD	$O(n^2)$	$O(n^2)$	O(n log n)	$O(n^2)$