Algoritmos de ordenación en memoria principal

	¿Qué hace?		Coste Mejor Caso	Coste Peor Caso	Coste Medio
Inserción Directa	Considera el primer elemento ordenado y a partir de esa premisa va insertando los elementos en el sitio y desplazando el resto hacia la derecha	Comp.	$C_{min} = n-1$	$C_{max} = \sum_{i=1}^{n} i - 1 = \frac{n^2+n-2}{2}$	$C_{prom} = \frac{1}{2} C_{max} = \frac{n^2+n-2}{4}$
		Mov.	$M_{min} = 2(n-$	$M_{max} = \frac{n^2+3n-4}{2}$	$M_{prom} = \frac{n^2+9n-10}{4}$
Inserción Binaria	Como inserción directa, pero la búsqueda del sitio la hace por dicotomía.	Comp.	$C_{min} = n-1$	$\Omega(I.B.) = \Theta(n^2)$	$C = n(\log_2 n - \log_2 e) + \log_2$
		Mov.	Como la directa	Como la directa	Como la directa
Selección Directa	Selecciona el elemento menor del vector sin ordenar y lo coloca al final de la parte ordenada.	Comp.	$C_{min} = 3(n-$	$C_{min} = 3(n-1)$	$C_{min} = 3(n-1)$
		Mov.		$M_{max} = 3(n-1) + \frac{n^2}{4}$	$M_{prom} \cong n(\ln n + \gamma)$ *donde* $\gamma = 0.577216$ ...
Intercambio Directo	Compara un par de elementos contiguos, si no está ordenado lo ordena.	Comp.	$C = \frac{n^2-n}{2}$	$C = \frac{n^2-n}{2}$	$C = \frac{n^2-n}{2}$
		Mov.	$M_{min} = 0$	$M_{max} = 3C = 3\frac{n^2-n}{2}$	$M_{prom} = \frac{1}{2} M_{max} = 3\frac{n^2-n}{4}$
Sacudida	Como el de la burbuja pero en ambos sentidos: ascendente y descendente	Comp.	$C_{min} = n-1$	$C_{max} = n - k_1\sqrt{n}$	$C_{prom} = \frac{1}{2}(n^2 - n(k_2 + \ln n))$
		Mov.	Como Intecambio directo	Como Intecambio directo	Como Intecambio directo
Shel	Se basa en la ordenación por inserción de los elementos que difieren h_1 posiciones, luego h_2, asi hasta h_t=1. Es dificil decidir la mejor secuencia: $h_k = 2^k - 1 = 1,3,5,7,...$ $h_i = 9 \times 4^i - 9 \times 2^i + 1$	Observaciones	El análisis detallado del algoritmo es supercomplicado y el libro no lo explica.		
		Coste asintótico (siendo n potencia de 2)	$O\left(\sum_{i=1}^{n/2} i - 1\right) \equiv \Theta(n^2)$	$O\left(\sum_{i=1}^{t} n^2/h_i\right) \equiv \Theta(n^2)$	
Montón	Este algoritmo usa las propiedades de montón para ordenar un vector en dos veces: construyendo un montón en la primera mitad primero y kuego invirtiendo el vector para aplicar el mismo método.	Comp.	$C_{min} = 2n(\log i)$	$C_{max} = 2n\log n - O(n)$	$C_{prom} = 2n\log n - O(n\log(\log n))$
		Mov.	¿??????	¿??????	¿??????
QuickSort	Este algoritmo es una combinación del intercambio directo y de divide y vencerás. Elige un "pivote" al azar y recorre el vector desde la derecha buscando elementos mayores que el pivote para intercambiarlos con los valores menores a la izquierda del pivote. Hasta encontrarse el incremento y el decremento y luego vuelve a comenzar con cada una de las particiones.	Observaciones	El algoritmo es de naturaleza recursiva, así que se usan la famosas fórmulas descritas en [Peña]		
		Coste asintótico	$T(n) = 2T(n/2) + cn = O(n \log n)$	$T(n) = T(1) + c\sum_{i=2}^{n} i = O(n^2)$	$T(n) = O(n\log n)$