Algorithm	Mnemonic	Type	Memory	Advantages	Disadvantages	Description	Time Complexity	Space Complexity
Selection Sort	"Children's sort" Choose an **element** right of "current"	Iterative	In-place	• Only n exchanges in the worst case, which is less than most other algorithms	• O(n^2) time complexity, even in the best case	• Find smallest item and put it in first position • Find next-smallest item and put it in second position, etc.	Best **O(n^2)** Avg **O(n^2)** Worst **O(n^2)**	**O(1)**
Insertion Sort	"Hand of Cards" sort Choose a **position** left of "current"	Iterative	In-place	• Linear time O(n) if input is already sorted • One of the fastest algorithms for partially sorted arrays		Iterate through array and push each item to the left as long as it is smaller than its left neighbour	Best **O(n)** Avg **O(n^2)** Worst **O(n^2)**	**O(1)**
Shell Sort	Generalisation of insertion sort (using a gap > 1)	Iterative	In-place				Best Avg Worst	**O(1)**
Mergesort	Sort left, sort right merge the two sorted sub-arrays	Recursive *Recursion levels: log n*	Requires copy of input array O(n) space	• Asymptotically optimal, i.e. worst-case time complexity is O(n log n)	• Not in-place: requires O(n) extra space to hold copy of input array	1. Sort left half of array 2. Sort right half of array 3. Merge the two sorted sub-arrays	Best **O(n log n)** Avg **O(n log n)** Worst **O(n log n)**	**O(n)**
Quicksort	Pivot left ≤ pivot ≤ right	Recursive *Recursion levels: log n (best) to n (worst) (with opt. log n worst)*	In-place But recursive, therefore O(log n) space	• Optimal O(n log n) time in average case • In practice, most of the time faster than any other algorithm	• Bad worst-case time performance of O(n^2) (if pivot is always min or max item) • Prevent this by shuffling input before sorting	1. Partition array around pivot 2. Sort sub-array left of pivot 3. Sort sub-array right of pivot	Best **O(n log n)** Avg **O(n log n)** Worst **O(n^2)**	**O(log n)**
Heapsort				• a	• W	1. P	Best **O(n log n)** Avg **O(n log n)** Worst **O(n log n)**	**O(1)**
Bubble Sort				• a	• W	1. P	Best **O(n)** Avg **O(n^2)** Worst **O(n^2)**	**O(1)**