Summary of Sorting Algorithms

- Description
- Performance/Complexity
  - run time
  - memory requirements
- Stability – when sorted by 2 keys, the first sorting is preserved within the second sorting
- Applications

		Sorts	**Description**	**Best used for:**	**Additional memory**	**Complexity**
Simple Non-recursive Small files	Stable	Bubble	Compares and swaps adjacent elements. Several passes are needed	General purpose	no	O(N^2)
		Selection	Finds subsequent maximums / minimums	Large records (very little swapping)		
		Insertion	Assumes part of the array is sorted, inserts the next element there.	Almost sorted files		
	Not stable	ShellSort	Compares and swaps elements jumping over the array	General purpose, no information about the records	A little	O($N^{3/2}$)
Advanced Not stable Large files	Recursive	QuickSort	Splits the array in two sets and a middle element: smaller to the left, bigger to the right. Then sorts recursively each set	General, commercial, usually very fast **Warning!!!** Run time may increase to O(N^2)	no	O(NlogN)
	Non-recursive	HeapSort	Uses priority heap - the smallest/largest is at the top always	Guaranteed runtime		
	Recursive	MergeSort	Splits the array into two, sorts recursively and then merges	Never used for main memory sort. The idea is applied for external sorting	yes	