Sorting Algorithms

- **in-place** sorting algorithms do not require additional memory except for a few units of memory
- **stable** sorting algorithms preserve the relative order of elements with identical keys

	space	in-place	stable	best	average	worst	comment
SELECTION	$\mathcal{O}(1)$	✓	✗	n^2	n^2	n^2	
INSERTION	$\mathcal{O}(1)$	✓	✓	$\boxed{n}$	n^2	n^2	small arrays
SHELL	$\mathcal{O}(1)$	✓	✗				medium-sized arrays
MERGE	$\mathcal{O}(n)$	✗	✓	$n \log n$	$n \log n$	$n \log n$	linked lists / very large collections
QUICK	$\mathcal{O}(\log n)$	✓	✗	$n \log n$	$n \log n$	$\boxed{n^2}$	$\mathcal{O}(n^2)$ given sorted arrays
HEAP	$\mathcal{O}(1)$	✓	✗	$n \log n$	$n \log n$	$n \log n$	$\Omega(n)$ given identical keys
COUNTING	$\mathcal{O}(k+n)$	✗	✓	$n+k$	$n+k$	$n+k$	k: the maximum key value

slow HEAP < MERGE < QUICK fast (for random data)

Search Algorithms

	best	average	worst
SEQUENTIAL	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
BINARY	$\mathcal{O}(1)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$
INTERPOLATION	$\mathcal{O}(1)$	$\mathcal{O}(\log(\log n))$	$\mathcal{O}(n)$

Data Structures

	space	access	search	insertion	deletion	comment
`Array`	$\Theta(n)$	$\Theta(1)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	
`LinkedList`	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$	
`Stack`	$\Theta(n)$	-	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$	
`Queue`	$\Theta(n)$	-	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$	
`BST`	$\Theta(n)$	-	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$	$\mathcal{O}(n)$ given poor balance
`AVL`	$\Theta(n)$	-	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$	
`2-3`	$\Theta(n)$	-	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$	worst: all 2-nodes / best: all 3-nodes
`HashMap`	$\Theta(n)$	-	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\mathcal{O}(n)$ given uneven distribution

- **full** tree: each node has 0 or 2 non-empty children.
- **complete** tree: each level is filled left to right. (All levels except the last are completely filled.)
- **heap**: complete binary tree; each child $\leq$ its parent