Algorithm		Complexity			Notes
		Time		Aux-Space	
		Best-case	**Worst-case**		
Bubble Sort		O(N)	O(N^2)	O(1)	N = number of elements in the list
Selection Sort		O(N^2)	O(N^2)	O(1)	N = number of elements in the list
Insertion Sort		O(N)	O(N^2)	O(1)	N = number of elements in the list
Merge Sort		O(N logN)	O(N logN)	O(N)	N = number of elements in the list
Heap Sort		O(N logN)	O(N logN)	O(1)	N = number of elements in the list
Heap Sort	Insert	O(logN)	O(logN)	O(1)	N = number of elements in the list
	Delete	O(logN)	O(logN)	O(1)	N = number of elements in the list
	Rise	O(logN)	O(logN)	O(1)	N = number of elements in the list
	Fall	O(logN)	O(logN)	O(1)	N = number of elements in the list
Quick Sort		O(N logN)	O(N^2)	O(logN)	N = number of elements in the array
Counting Sort		O(N+k)	O(N+k)	O(k)	N = number of elements in the input array, k = range of the input elements
Radix Sort		O(Nk)	O(Nk)	O(N+k)	N = number of elements in the array, k = number of digits in each element
QuickSelect (Pivot selection)	First Element	O(N)	O(N^2)	O(logN)	N = size of the input list
	Last Element	O(N)	O(N^2)	O(logN)	N = size of the input list
	Minimum Element	O(N)	O(N^2)	O(N)	N = size of the input list
	Random Element	O(N)	O(N^2)	O(logN)	N = size of the input list
	10% Greater than all	O(N)	O(N logN)	O(N)	N = size of the input list
	Median Element	O(N)	O(N logN)	O(logN)	N = size of the input list
Dutch National Flag		O(N)	O(N^2)	O(logN)	N = number of elements in the sub-array being partitioned
Hoare Partitioning		O(N)	O(N^2)	O(1)	N = number of elements in the sub-array being partitioned
Lomuto Partitioning		O(N)	O(N^2)	O(1)	N = number of elements in the sub-array being partitioned
Out-Of-Place Partitioning		O(N)	O(N^2)	O(N)	N = number of elements in the sub-array being partitioned
Median of Medians		O(N)	O(N)	O(logN)	N = number of elements in the sub-array being partitioned
Depth-First Search		O(V+E)	O(V+E)	O(V+E)	V = number of vertices, E = number of edges
Breadth-First Search		O(V+E)	O(V+E)	O(V+E)	V = number of vertices, E = number of edges
Dijkstra's		O(E logV)	O(E logV)	O(V+E)	V = number of vertices, E = number of edges
Topological Sort		O(V+E)	O(V+E)	O(V)	V = number of vertices, E = number of edges
Kahn's		O(V+E)	O(V+E)	O(V)	V = number of vertices, E = number of edges
Prim's		O(E logV)	O(E logV)	O(V)	V = number of vertices, E = number of edges
Kruskal's		O(E logV)	O(E logV)	O(V+E)	V = number of vertices, E = number of edges
Bellman-Ford		O(VE)	O(VE)	O(E)	V = number of vertices, E = number of edges
Floyd-Warshall		O(V^3)	O(V^3)	O(V^2)	V = number of vertices, E = number of edges
Hashing	Sorted Array	O(logN)	O(N)	O(N)	N = size of the array
	Unsorted Array	O(1)	O(N)	O(N)	N = size of the array
	BST	O(logN)	O(N)	O(N)	N = number of elements in the tree
	AVL Tree	O(logN)	O(logN)	O(N)	N = number of elements in the tree
Ford-Fulkerson	Adjacency Matrix	O(EF)	O(EF)	O(V^2)	E = number of edges in the graph, F = maximum flow value, V = number of vertices in the graph
	Adjacency List	O(EF)	O(EF)	O(V+E)	E = number of edges in the graph, F = maximum flow value, V = number of vertices in the graph, E = number of edges
Min-cut Max-flow		O(V E^2)	O(V E^2) O(V E logVC)	O(V+E)	V = number of vertices, E = number of edges in the network, C = maximum capacity in the network
Hash Function	Integers	O(1)	O(N)	O(N)	N = number of integers being hashed
	Strings	O(1)	O(N)	O(N)	N = number of strings being hashed
BST & AVL	Insertion	O(logN)	O(N)	O(1)	N = number of nodes in the tree
	Deletion	O(logN)	O(logN)	O(1)	N = number of nodes in the tree
	Search	O(1)	O(logN)	O(1)	N = number of nodes in the tree
AVL rebalancing	Insertion	O(1)	O(logN)	O(N)	N = number of nodes
	Deletion	O(1)	O(logN)	O(N)	N = number of nodes
Prefix Trie	Insertion	O(M)	O(M)	O(M)	M = length of the string being inserted
	Searching	O(k)	O(k)	O(1)	k = length of the search string
Suffix Tree		O(N)	O(N^2)	O(N)	N = length of the input string
Suffix Array		O(N logN)	O(N^2 logN)	O(N)	N = length of the input string
Constructing Suffix Array using Prefix Doubling		O(N logN)	O(N log^2 N)	O(N logN)	N = length of the input string
Minimum Spanning Tree		O(E logV)	O(E logV)	O(V+E)	V = number of vertices, E = number of edges
Maximum Spanning Tree		O(E logV)	O(E logV)	O(V+E)	V = number of vertices, E = number of edges
Adjacency List	Edge Existance	O(X)	O(X)	O(V+E)	X = number of adjacent vertices of a vertex, V = number of vertices, E = number of edges
	Vertex Neighbour	O(X)	O(X)	O(V+E)	X = number of adjacent vertices of a vertex, V = number of vertices, E = number of edges
Adjacency Matrix	Edge Existance	O(1)	O(1)	O(V^2)	V = number of vertices
	Vertex Neighbour	O(V)	O(V)	O(V^2)	V = number of vertices