Data Structures

O(1) < O(log(n)) < O(n) < O(n log(n)) < O (n2) < O (2n) < O (n!)

Having same average and worst case:

	Access	Search	Insertion	Deletion
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))
Splay Tree	-	Θ(log(n))	Θ(log(n))	Θ(log(n))
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))

Note: It takes O(1) only when the pointer is given to where the insertion or deletion is to be made in the linked list otherwise first the location where insertion or deletion has to be done is to be found out which might take O(n) time.

Having different average and worst case:

KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O (n)	O (n)	O (n)	O (n)
Hash Table	-	Θ(1)	Θ(1)	Θ(1)	-	O (n)	O (n)	O (n)

Heap Data Structure:

S – Sorted, US – Unsorted , Binary Heap Heapify – O (n)

	Find Max	Extract Max	Increase Key	Insert	Delete	Merge
Linked List (S)	O (1)	O (1)	O (n)	O (n)	O (1)	O (m+n)
Linked List (US)	O (n)	O (n)	O (1)	O (1)	O (1)	O (1)
Binary Heap	O (1)	O (log (n))	O (log (n))	O (log (n))	O (log (n))	O (m+n)
Pairing Heap	O (1)	O (log (n))	O (log (n))	O (1)	O (log (n))	O (1)
Binomial Heap	O (1)	O (log (n))	O (log (n))	O (1)	O (log (n))	O (log (n))
Fibonacci Heap	O (1)	O (log (n))	O (1)	O (1)	O (log (n))	O (1)

Graph Data Structure:

	Storage	Add Vertex	Add Edge	Remove Vertex	Remove Vertex	Query
Adjacency list	O (|V| + |E|)	O (1)	O (1)	O (|V| + |E|)	O (|E|)	O (|V|)
Incidence list	O (|V| + |E|)	O (1)	O (1)	O (|E|)	O (|E|)	O (|E|)
Adjacency matrix	O ($\|V\|^2$)	O ($\|V\|^2$)	O (1)	O ($\|V\|^2$)	O (1)	O (1)
Incidence matrix	O (|V| . |E|)	O (|V| . |E|)	O (|V| . |E|)	O (|V| . |E|)	O (|V| . |E|)	O (|E|)