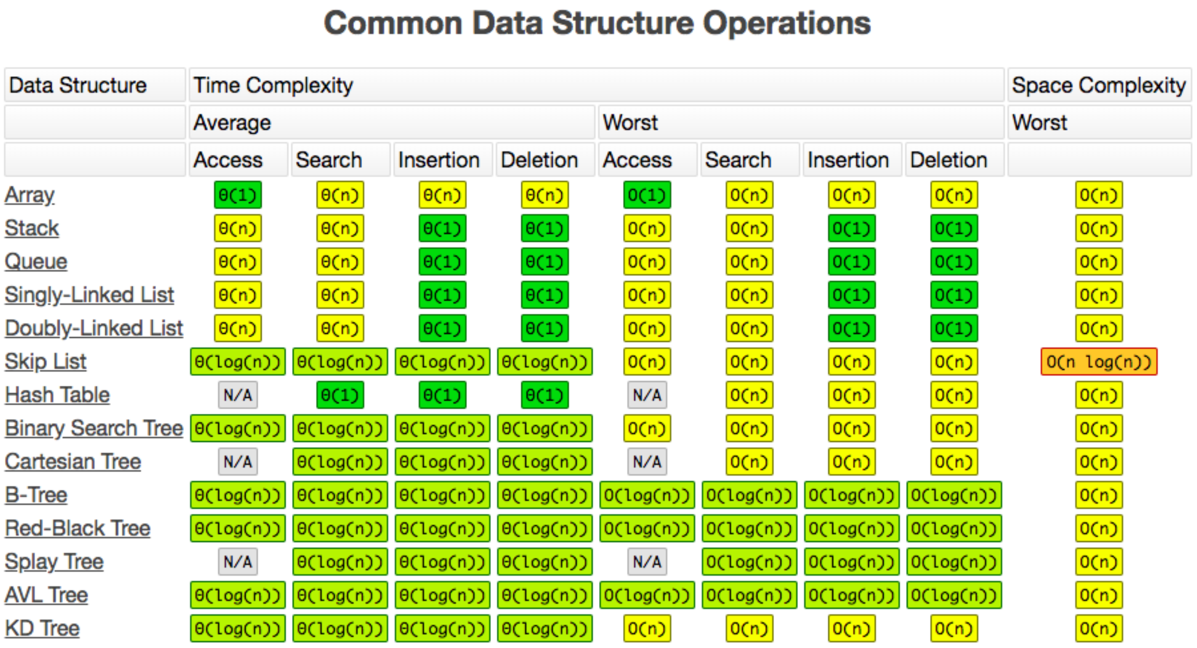

Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	θ(1)	θ(n)	θ(n)	θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	θ(n)	θ(n)	θ(1)	θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	θ(n)	θ(n)	θ(1)	θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	θ(n)	θ(n)	θ(1)	θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	θ(n)	θ(n)	θ(1)	θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	θ(log(n))	θ(log(n))	θ(log(n))	θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	θ(1)	θ(1)	θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	θ(log(n))	θ(log(n))	θ(log(n))	θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	θ(log(n))	θ(log(n))	θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	θ(log(n))	θ(log(n))	θ(log(n))	θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	θ(log(n))	θ(log(n))	θ(log(n))	θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	θ(log(n))	θ(log(n))	θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	θ(log(n))	θ(log(n))	θ(log(n))	θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	θ(log(n))	θ(log(n))	θ(log(n))	θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

Linked List

search	O(n)
space	O(n)
insert	O(n)
delete	O(n)
prepend	O(1)
append	O(1)

Advantage

- Good for append and prepending nodes.
- Does not create memory overflow
- Dynamically memory data structure
- Implementation of dynamic queue and stack to avoid stack overflow issue

Disadvantage

- Slow search because can't use binary search
- Requires more memory than array

Use case

- Image viewer
- Browser back and next history
- Music player

Stack(Last in First Out)

Push	O(1)
Pop	O(1)
Peek	O(1)
IsEmpty	O(1)
Size	O(1)

Space complexity O(n)

Advantage

- Removing and Inserting is just O(1)

Disadvantage

- Implementing a stack as an array can result in stack overflow so use linked list if necessary

Use cases

- The call stack
- Depth first Search
- String parsing
- Undo/Redo

Queue(First In First Out)

Equeue	O(1)
Dequeue	O(1)
IsEmpty	O(1)
Size	O(1)

Space complexity O(n)

Advantage and Disadvantages are same as Stack fast lookup depends on use case

Use Case:

- BFS
- Printer
- Job Scheduling

Hash Map

	Average	Worst Case
Space	O(n)	O(n)
Insert	O(1)	O(n)
Lookup	O(1)	O(n)
Delete	O(1)	O(n)

Advantage

- Fast lookup: Lookup takes O(1) time on average
- Flexible keys: Most data types can be keys, as long as they are hashtable

Disadvantage

- Slow worst case lookup: Lookup takes O(n) time in the worst case
- Unordered: Keys aren't stored in a special order, if you're looking for the smallest key, the largest key in a range, you'll need to look through every key to find it
- Single directional lookup
- Not cache friendly

Use Case:

- Dictionary
- Book shelf

BFS Tree Traversal

Time complexity	O(|V|)
Space complexity	O(|E|)

Advantage

- A BFS will find the shortest path between starting point and any other reachable nodes

Disadvantage

- A BFS on a binary tree generally require more memory than DFS