Common Data Structures

Arrays

- Good when the data is a fixed size so no need to adjust the size of the array

Search
- O(n) Worst Case
- If sorted then O(logn)

Get
- O(1) Worst Case

Append
- O(n) Worst Case

Linked List

- Good for implementing queues or stacks because appending and removing from the head or tail is O(1)

Search
- O(n) Worst Case

Get
- O(n) Worst Case

Append
- O(1) Worst Case

Hash Map

- Key-Value store good for storing keys that need to mapped to a value and the keys need to be access quickly

Add/Lookup Key
- O(1) Average Case, in an interview you can assume all hashmaps are O(1) add and lookup time
- O(n) Worst Case depending on hash function

Binary Trees

- Binary Trees are different than Binary Search Trees, Binary Trees are the superset of Binary Search Trees
- It does not need to maintain a particular order

Search
- O(n) Worst Time

Add
- O(logn) Worst Time

Binary Search Trees

- Binary Search Trees need to maintain a particular order
- Useful for keeping sorted order

Search
- O(logn) Worst Time

Add
- O(logn) Worst Time

Graphs

Search
- O(V + E) where V is the number of nodes and E is the number of edges (Worst Case)

Heap / Priority Queue

Insertion/Deletion/Pop
- O(logn) Worst Case