**Time Complexity of various Data Structure**

**Linked List:**

* Insertion: **Insertion at the head** requires you to add the element and update the head pointer. Time complexity = O(1)  
  **Insertion at the tail** requires you to keep a pointer to the tail element, add the element at the tail and update the tail pointer. Time complexity = O(1)
* **Deletion:** **Deleting the head** **element**requires updating the head and deleting the previously head element. Time complexity = O(1).  
  **Deleting the tail** **element** has a time complexity of O(n) as we have to transverse through the entire linked list to update next pointer for each.
* **Searching:** Time complexity is equal to O(n).

Stacks:

* Push: Time complexity is **O(1)**
* Pop: Time complexity is **O(1)**
* Searching: Time complexity is **O(n)**

**Queues:**

* **Enqueue:** O(1)
* **Dequeue:** O(1)
* **Searching:** O(n)

**Graphs:**

The Time complexity of both **BFS and DFS will be O(V + E)**, where V is the number of vertices and E is the number of Edges. If it is an adjacency matrix, it will be O(V^2). If we use an adjacency list, it will be O(V+E).

Type Add vertex Add edge Remove vertex Remove edge

Adjacency list O(1) O(1) O (|V| + |E|) O(|E|)

Adjacency matrix O(|V|2 ) O(1) O(|V|2 ) O(1)

**Binary Tree:**

**Insertion, Deletion and Searching:** Average case: **O(log n)**   
 Worst Case(Unbalanced tree): **O(n)**