Analysis of Important Data Structures

1 Singly and Doubly Linked List

1.1 Operators Allowed

- Insert at Front
- Insert at Back
- Insert at Position
- Search
- Delete with Node Value Given
- Delete with Position
- Display

1.2 C++ Implementation

- Singly Linked List Git-Location : DataStructures/source/SinglyLinkedList.cpp
- Double Linked List Git-Location : DataStructures/source/DoublyLinkedList.cpp

1.3 Time and Space Complexity Analysis

Method	Time(Avg)	Time(Best)	Time(Worst)	Space(Avg)	Space(Best)	Space(Worst)
Insert	O(1)	O(n)	O(n)	constant	constant	constant
Delete	O(n)	O(1)	O(n)	constant	constant	constant
Search	O(n)	O(1)	O(n)	constant	constant	constant
Form-n-List	O(n)	O(n)	O(n)	O(n)	O(n)	O(n)

2 Binary Tree Types

- **Full/Proper/Plane/Strictly BT** : A full binary tree (sometimes referred to as a proper or plane binary tree) is a tree in which every node in the tree has either 0 or 2 children.
- **Complete BT** : A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.

3 Binary Search Tree(BST)

3.1 C++ Implementation

- Binary Search Tree Git-Location : DataStructures/source/BinarySearchTree.cpp

3.2 Time and Space Complexity Analysis

Method	Time(Avg)	Time(Best)	Time(Worst)	Space(Avg)	Space(Best)	Space(Worst)
Insert	O(log_2n)	O(1)	O(n)[1]	constant	constant	constant
Delete	O(log_2n)	O(1)	O(n)[1]	constant	constant	constant
Search	O(log_2n)	O(1)	O(n)[1]	constant	constant	constant
Form-n-BST	O(nlog_2n)	O(nlog_2n)	O(n)[1]	O(n)	O(n)	O(n)

- note-1 : If the BST is formed in the worst way such that all the elements are either on the right/left of every node (8-¿9-¿10).