Explain the different types of linked lists (Singly Linked List, Doubly Linked List).

A Singly Linked List is a data structure that consists of elements (nodes) with each node pointing to the next node with the help of a single pointer. Since there is only one link, it is a simple task to make insertions or deletions, and it will be the only one that need to be updated or deleted the rest symbolic links. Traversal in a linear manner which starts from the head node and ends at the tail node. It is well-suited for scenarios with frequent insertions and deletions where backward traversal isn’t needed.

A Doubly Linked List contains nodes which are arranged in such a way that there are two pointers, one to the next node and one to the previous node. They can point to any direction allowing a full motion of data. But it has an overhead of memory which would be used as an additional pointer to the previous node. It is often seen as a better alternative to BSTs and even more useful in navigation systems, robotics, and power networks. An example of one such system is the global sensor network. Analyze the time complexity of each operation.

Time Complexity:

* Add Task: O(n) in the worst case if we need to traverse the list to find the end.
* Search Task: O(n) since we might have to traverse the whole list.
* Traverse Tasks: O(n) since we need to visit every node.
* Delete Task: O(n) in the worst case if we need to traverse the list to find the task.

Discuss the advantages of linked lists over arrays for dynamic data.

* Dynamic Size: A linked list can change its size at will so to speak, it can grow and shrink by changing pointers without changing the structure. Arrays must have a fixed size or resizing operations when the number of elements changes. This linked list property provides it with strong capabilities to manipulate varying data sizes with greater efficiency.
* Efficient Insertions and Deletions: Insertions and deletions in linked lists, however, are generally more efficient, especially for big datasets. Adding or removing elements from linked lists does not require shifting the elements to occupy the empty places in memory since they are not contiguous as opposed to arrays. Therefore, there will be fewer resources and faster data modifications. This leads to better overall performance.