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

**Types of Linked Lists**

1. **Singly Linked List**
   * **Structure**: Each node contains a data element and a reference to the next node in the sequence.
   * **Operations**:
     + **Add**: Insert a new node at the beginning, middle, or end.
     + **Search**: Traverse the list to find a node with the desired value.
     + **Delete**: Remove a node by adjusting the references of adjacent nodes.
   * **Advantages**:
     + Simple implementation.
     + Dynamic size: Can grow and shrink as needed.
2. **Doubly Linked List**
   * **Structure**: Each node contains a data element, a reference to the next node, and a reference to the previous node.
   * **Operations**:
     + **Add**: Insert a node at the beginning, middle, or end with easier access to both previous and next nodes.
     + **Search**: Traverse in both directions (forward and backward).
     + **Delete**: Remove a node by adjusting the references of both adjacent nodes.
   * **Advantages**:
     + Allows traversal in both directions.
     + Easier to delete nodes from any position without needing to traverse from the head

**Analyze the time complexity of each operation.**

**Time Complexity**

1. **Add a Task**:
   * **Average Case**: O(n) – Traversing to the end of the list to add a new task.
   * **Worst Case**: O(n) – As above, involves traversal.
2. **Search for a Task**:
   * **Average Case**: O(n) – Linear search through the list.
   * **Worst Case**: O(n) – As above, involves traversal.
3. **Traverse Tasks**:
   * **Time Complexity**: O(n) – Visiting each node once.
4. **Delete a Task**:
   * **Average Case**: O(n) – Linear search followed by adjusting references.
   * **Worst Case**: O(n) – As above, involves traversal.

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

**Advantages of Linked Lists over Arrays for Dynamic Data**

* **Dynamic Size**: Linked lists can grow and shrink dynamically, while arrays have a fixed size.
* **Efficient Insertions/Deletions**: Linked lists allow efficient insertions and deletions (O(1)) if the node reference is known, as it only requires adjusting references.
* **No Wasted Space**: Linked lists do not require pre-allocation of memory, reducing wasted space compared to arrays which may be over-allocated or require resizing.