### Time Complexities

| **Operation** | **Singly Linked List** | **Doubly Linked List** |
| --- | --- | --- |
| Add (Beginning) | O(1) | O(1) |
| Add (End) | O(n) | O(1) |
| Add (Specific Position) | O(n) | O(n) |
| Search by Value | O(n) | O(n) |
| Traverse | O(n) | O(n) |
| Delete (By Value) | O(n) | O(n) |
| Delete (By Position) | O(n) | O(n) |

### Advantages

| **Advantage** | **Linked Lists** | **Arrays** |
| --- | --- | --- |
| Dynamic Size | Can grow/shrink dynamically | Fixed size or resizing needed |
| Efficient Insertions/Deletions | O(1) for nodes | O(n) due to shifting elements |
| No Wasted Space | Memory as needed | Potential wasted space |
| Ease of Complex Operations | Simplifies operations | Requires additional steps |
| Flexibility | Supports complex structures | More rigid |

### When to Use Linked Lists Over Arrays

* **Frequent Insertions/Deletions:** Linked lists are more efficient.
* **Unknown or Variable Size:** Linked lists handle resizing better.
* **Complex Data Structures:** Linked lists provide a more suitable foundation.