Quick Revision Notes Guide (Ex: Singly Linked List)

1️⃣ **Singly Linked List** contains 2 nodes:

- **Data** (value stored in the node)
- **Pointer** (reference to the next node in the list)

2️⃣ Structure of a Node:

```
class Node {
   int data;
   Node next;
   Node(int data) {
      this.data = data;
      this.next = null;
   }
}
```

3️⃣ Basic Operations:

1. Insertion

- **At the Beginning** (O(1))
- **At the End** (O(n))
- **At a Specific Position** (O(n))

```
void insertAtBeginning(int data) {
   Node newNode = new Node(data);
   newNode.next = head;
   head = newNode;
}
```

2. Deletion

- **From the Beginning** (O(1))
- **From the End** (O(n))
- **From a Specific Position** (O(n))

```
void deleteFromBeginning() {
   if (head == null) return;
   head = head.next;
}
```

3. Searching (O(n))

```
boolean search(int key) {
    Node temp = head;
    while (temp != null) {
        if (temp.data == key) return true;
        temp = temp.next;
    }
    return false;
}
```

4. Traversal (O(n))

```
void traverse() {
    Node temp = head;
    while (temp != null) {
        System.out.print(temp.data + " -> ");
        temp = temp.next;
    }
    System.out.println("null");
}
```

4️⃣ Advantages:

✅ Dynamic Size (No need to preallocate memory) ✅ Efficient Insertion/Deletion (compared to arrays)

5️⃣ Disadvantages:

❌ More memory required (extra pointer per node) ❌ Accessing an element is slower (O(n) search time)

6️⃣ Time Complexity Summary:

Operation	**Time Complexity**
Insertion (Beginning)	O(1)
Insertion (End)	O(n)
Deletion (Beginning)	O(1)
Deletion (End)	O(n)
Searching	O(n)
Traversal	O(n)

(**NOTE**: if any DS have different patterns of questions, NOTE DOWN ALL PATTERNS in short notes)