* + **Explain Big O notation and how it helps in analyzing algorithms.**

Big O notation describes the upper bound of an algorithm's running time or space as the input size grows. It helps in analyzing scalability and efficiency.

* **O(1)** – Constant time
* **O(log n)** – Logarithmic time (e.g., binary search)
* **O(n)** – Linear time (e.g., linear search)
* **O(n log n)** – Log-linear time (e.g., quick sort)
* **O(n²)** – Quadratic time (e.g., bubble sort)
  + **Describe the best, average, and worst-case scenarios for search operations.**
* **Linear Search:**
  + **Best case:** O(1) [ Item is at the first index ]
  + **Average case:** O(n/2) [ On average, half the array is checked ]
  + **Worst case:** O(n) [ Item is at the end or not present ]
* **Binary Search:**
  + **Best case:** O(1) [ Item is in the middle ]
  + **Average case:** O(log n)
  + **Worst case:** O(log n)
  + **Compare the time complexity of linear and binary search algorithms.**

|  |  |  |
| --- | --- | --- |
| Time Complexity | Linear Search | Binary Search |
| Best Case | O(1) | O(1) |
| Average Case | O(n/2) | O(log n) |
| Worst Case | O(n) | O(log n) |

* + **Discuss which algorithm is more suitable for your platform and why.**

Binary Search algorithm is more suitable because:

* Faster search performance
* Scales better with larger datasets
* Predictable and consistent