**Session: Binary Search - Concepts, Examples, and Interview Questions**

**Topic:** Binary Search

What is Binary Search?

* A search algorithm used on **sorted arrays/lists**.
* Works by **dividing the search space in half**.
* Time Complexity: **O(log n)**
* Space Complexity: **O(1)** (Iterative), **O(log n)** (Recursive due to call stack)

**Steps:**

* Find the middle element.
* Compare target with the middle.
* If equal, return index.
* If smaller, search left half.
* If larger, search right half.

When to Use Binary Search

* Array is **sorted**.
* You want **fast lookup**.
* Suitable for problems with **search space reduction**.

Examples:

* Find an element in a sorted array.
* Search insert position.
* First/last occurrence in duplicates. [1,2,2,3,3,3,3,3,3,4,5]
* Square root, rotated arrays, peak element problems.

Example 1 - Basic Binary Search

**Problem:** Find the index of 7 in sorted array [1, 3, 5, 7, 9, 11] -> low : 7, high : 11

arr = [1, 3, 5, 7, 9, 11]  
target = 7  
low=0, high = len(arr)-1  
  
**while** low <= high:  
 mid = (low + high) / 2 -> 3  
 **if** arr[mid] == target:  
 **return** mid  
 **elif** arr[mid] < target:  
 low = mid + 1  
 **else**:  
 high = mid - 1

**Output:** Index 3

Example 2 - First Occurrence in Duplicates

**Problem:** Find first occurrence of 4 in [1, 2, 4, 4, 4, 5, 6,...INT\_LONG]

arr = [1, 2, 4, 4, 4, 5, 6]  
target = 4  
low, high = 0, len(arr)-1  
result = -1  
  
**while** low <= high:  
 mid = (low + high) // 2  
 **if** arr[mid] == target:  
 result = mid  
 high = mid - 1 *# Move left*  
 **elif** arr[mid] < target:  
 low = mid + 1  
 **else**:  
 high = mid - 1

**Output:** Index 2

Common Mistakes

* Not updating low or high correctly
* Infinite loops from mid = (low + high) instead of (low + high)//2
* Not considering **overflow** in some languages: prefer low + (high - low)/2
* Not checking base case properly in recursive version

Medium-Level Binary Search Variations (Practice Questions)

* Find Minimum in Rotated Sorted Array : <https://leetcode.com/problems/find-minimum-in-rotated-sorted-array/description/>
* Find Peak element in array : <https://leetcode.com/problems/find-peak-element/description/>
* Find first and last occurrence of an element in sorted array : <https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/description/>
* Search in a row wise and column wise sorted matrix : <https://leetcode.com/problems/search-a-2d-matrix/description/>
* Rotation Count in a Rotated Sorted array : <https://www.geeksforgeeks.org/dsa/find-rotation-count-rotated-sorted-array/>
* Find position of an element in an Infinite Sorted Array
* Median of Two Sorted Arrays : <https://leetcode.com/problems/median-of-two-sorted-arrays/>

Tips and Tricks

* Think whether binary search is on index or on answer/value
* If sorted array is given, high chances of binary search is there.

**Session: Linked List - Concepts, Examples, and Interview Questions (Java)**

What is a Linked List?

* A linear data structure where elements (nodes) point to the next.
* Each node contains:
* **Data**
* **Pointer to the next node**

**Types:**

* Singly Linked List (points to next element) 1-> 2->3->4->5
* Doubly Linked List (points to next and prev element) 1 <-> 2 <->3 <->4
* Circular Linked List (forms a circle) 1->2->3->4->5->1

Why Linked Lists?

* Dynamic size (unlike arrays).
* Efficient insert/delete at head
* Good for memory-constrained environments.

**Downsides:**

* No random access.
* Extra memory for pointers.

Basic Operations

* Insert at head
* Insert at tail
* Delete node
* Search node
* Reverse a linked list

Example - Reverse a Linked List

**class** ListNode {  
 int val;  
 ListNode next;

ListNode prev;  
 ListNode(int val) {  
 **this**.val = val;  
 }  
}

input : head -> 1<-2<-3

output : head -> 3->2->1

prev = 1, cur = 2

nextTemp = 2,

**public** ListNode reverseList(ListNode head) {  
 ListNode prev = **null**;  
 ListNode curr = head;  
 **while** (curr != **null**) {  
 ListNode nextTemp = curr.next;  
 curr.next = prev;  
 prev = curr;  
 curr = nextTemp;  
 }

head = prev;

return head;  
}

Common Interview Questions

* Reverse a Linked List : <https://leetcode.com/problems/reverse-linked-list/description/>
* Detect Cycle : <https://leetcode.com/problems/linked-list-cycle/description/>
* Merge Two Sorted Lists : <https://leetcode.com/problems/merge-two-sorted-lists/>
* Remove Nth Node from End : <https://leetcode.com/problems/remove-nth-node-from-end-of-list/description/>
* Middle of the Linked List : <https://leetcode.com/problems/swap-nodes-in-pairs/description/>