**Leetcode 704 – Binary Search**

## Problem Understanding

**Leetcode 704. Binary Search**  
You are given a **sorted** array of integers nums and an integer target.  
Return the **index** of the target if it is in the array.  
If not, return **-1**.

### Key Points:

* The input array is **sorted in ascending order**.
* You must implement the algorithm using **binary search**, achieving **O(log n)** time.

## Optimized Java Solution (Iterative Binary Search)

public class Solution {

public int search(int[] nums, int target) {

int left = 0, right = nums.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2; // prevents overflow

if (nums[mid] == target) return mid;

else if (nums[mid] < target) left = mid + 1;

else right = mid - 1;

}

return -1; // not found

}

}

## Dry Run Using Table

**Input:** nums = [-10, -3, 0, 5, 9, 12], target = 9

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step | left | right | mid | nums[mid] | Action |
| 1 | 0 | 5 | 2 | 0 | target > 0 → left = 3 |
| 2 | 3 | 5 | 4 | 9 | ✅ target found → return 4 |

✔**Output:** 4

## Time / Space Complexity

|  |  |  |
| --- | --- | --- |
| Metric | Value | Explanation |
| ⏱ Time | O(log n) | Halves the search space |
| 💾 Space | O(1) | Constant extra space used |

## Alternate Approaches

|  |  |
| --- | --- |
| Approach | Notes |
| ✅ Iterative Binary Search | Most efficient and simple |
| ✅ Recursive Binary Search | Clean, but uses call stack (O(log n) space) |
| ❌ Linear Search | O(n) time — not acceptable for this problem |