Top 50 Easy LeetCode Problems

Easy Level Problems

1. **Two Sum**
   Description: Given an array of integers, return indices of two numbers such that they add up to a specific target.
   Example: Input: [2,7,11,15], target = 9
   Output: [0,1]
   Constraints: Exactly one solution exists; same element cannot be used twice.

2. **Best Time to Buy and Sell Stock**
   Description: Find the maximum profit by choosing one day to buy and another to sell.
   Example: [7,1,5,3,6,4] → 5
   Constraints: Buy before sell.

3. **Contains Duplicate**
   Description: Determine if any value appears at least twice.
   Example: [1,2,3,1] → true
   Constraints: Use efficient time complexity.

4. **Single Number**
   Description: Every element appears twice except one. Find that one.
   Example: [4,1,2,1,2] → 4
   Constraints: Linear time and constant space.

5. **Plus One**
   Description: Add one to a number represented as an array of digits.
   Example: [1,2,9] → [1,3,0]
   Constraints: Handle carry.

6. **Move Zeroes**
   Description: Move all zeros to the end while maintaining order.
   Example: [0,1,0,3,12] → [1,3,12,0,0]
   Constraints: In-place operation.

7. **Valid Anagram**
   Description: Check whether two strings are anagrams.
   Example: "anagram", "nagaram" → true
   Constraints: Same character frequency.

8. **Valid Parentheses**
   Description: Determine if parentheses are valid.
   Example: "()[]" → true
   Constraints: Use stack.

9. **Merge Two Sorted Lists**
   Description: Merge two sorted linked lists into one sorted list.
   Example: [1,2,4], [1,3,4]
   Constraints: Maintain sorted order.

10. **Invert Binary Tree**
    Description: Invert left and right subtrees.
    Example: Input tree → Inverted tree
    Constraints: Recursive or iterative.

11. **Maximum Depth of Binary Tree**
    Description: Find maximum depth of a binary tree.
    Example: [3,9,20,null,null,15,7] → 3
    Constraints: DFS or BFS.

12. **Symmetric Tree**
    Description: Check whether a tree is a mirror of itself.
    Example: [1,2,2,3,4,4,3] → true
    Constraints: Recursive comparison.

13. **Binary Tree Paths**
    Description: Return all root-to-leaf paths.
    Example: [”1-¿2-¿5”,”1-¿3”]
    Constraints: Backtracking.

14. **Path Sum**
    Description: Check if tree has a root-to-leaf path equal to target sum.
    Example: target = 22 → true
    Constraints: Tree traversal.

15. **Climbing Stairs**
    Description: Count distinct ways to climb stairs taking 1 or 2 steps.
    Example: n = 3 → 3
    Constraints: Dynamic programming.

16. **Maximum Subarray**
    Description: Find contiguous subarray with maximum sum.
    Example: [-2,1,-3,4,-1,2,1,-5,4] → 6
    Constraints: Kadane’s Algorithm.

17. **Remove Duplicates from Sorted Array**
    Description: Remove duplicates in-place.
    Example: [1,1,2] → 2
    Constraints: Two-pointer approach.

18. **Implement strStr()**
    Description: Find first occurrence of substring.
    Example: ”hello”,”ll” → 2
    Constraints: Return -1 if not found.

19. **Search Insert Position**
    Description: Find index where target should be inserted.
    Example: [1,3,5,6], 5 → 2
    Constraints: Binary search.

20. **First Bad Version**
    Description: Find first bad version using API.
    Example: First bad = 4
    Constraints: Minimize API calls.

21. **Balanced Binary Tree**
    Description: Check height difference ≤ 1.
    Example: Balanced tree → true
    Constraints: DFS.

22. **Minimum Depth of Binary Tree**
    Description: Find shortest root-to-leaf path.
    Example: [3,9,20,null,null,15,7] → 2
    Constraints: BFS preferred.

23. **Valid Palindrome**
    Description: Check palindrome ignoring non-alphanumeric characters.
    Example: ”A man, a plan, a canal: Panama” → true
    Constraints: Two-pointer.

24. **Reverse String**
    Description: Reverse array of characters in-place.
    Example: [”h”,”e”,”l”,”l”,”o”]
    Constraints: O(1) space.

25. **Reverse Integer**
    Description: Reverse digits of integer.
    Example: 123 → 321
    Constraints: Handle overflow.

26. **Fizz Buzz**
    Description: Print numbers with Fizz/Buzz rules.
    Example: n=3 → [1,2,"Fizz"]
    Constraints: Loop-based.

27. **Happy Number**
    Description: Determine if number ends at 1.
    Example: 19 → true
    Constraints: Detect cycles.

28. **Missing Number**
    Description: Find missing number from 0 to n.
    Example: [3,0,1] → 2
    Constraints: O(n).

29. **Power of Two**
    Description: Check if number is power of 2.
    Example: 16 → true
    Constraints: Bit manipulation.

30. **Excel Sheet Column Title**
    Description: Convert number to Excel column title.
    Example: 28 → AB
    Constraints: Base-26 logic.

31. **Reverse Linked List**
    Description: Reverse a singly linked list.
    Example: 1→2→3 → 3→2→1
    Constraints: Iterative or recursive.

32. **Linked List Cycle**
    Description: Detect cycle in linked list.
    Example: Cycle exists → true
    Constraints: Floyd's cycle detection.

33. **Palindrome Linked List**
    Description: Check if linked list is palindrome.
    Example: 1→2→2→1 → true
    Constraints: Reverse second half.

34. **Min Stack**
    Description: Stack supporting getMin in O(1).
    Example: push, pop, getMin
    Constraints: Extra stack.

35. **Valid Mountain Array**
    Description: Check if array strictly increases then decreases.
    Example: [0,3,2,1] → true
    Constraints: Single peak.

36. **Shuffle the Array**
    Description: Rearrange elements in given pattern.
    Example: [2,5,1,3,4,7] → [2,3,5,4,1,7]
    Constraints: Linear time.

37. **Kids With the Greatest Number of Candies**
    Description: Check if each kid can have max candies.
    Example: [2,3,5,1,3], extra=3
    Constraints: Compare with max.

38. **Defanging an IP Address**
    Description: Replace "." with "[.]".
    Example: "1.1.1.1" → "1[.]1[.]1[.]1"
    Constraints: String replace.

39. **Number of Steps to Reduce to Zero**
    Description: Count steps to reduce number to zero.
    Example: 14 → 6
    Constraints: Bitwise or loop.

40. **Toeplitz Matrix**
    Description: Check if diagonals have same elements.
    Example: Valid matrix → true
    Constraints: Matrix traversal.

41. **Transpose Matrix**
    Description: Transpose given matrix.
    Example: [[1,2,3],[4,5,6]]
    Constraints: New matrix.

42. **Reshape the Matrix**
    Description: Change matrix dimensions if possible.
    Example: reshape allowed
    Constraints: Preserve elements.

43. **Height Checker**
    Description: Count indices differing from sorted array.
    Example: [1,1,4,2,1,3] → 3
    Constraints: Sorting.

44. **Ransom Note**
    Description: Check if note can be constructed from magazine.
    Example: "a","b" → false
    Constraints: Frequency count.

45. **Hamming Distance**
    Description: Count differing bits.
    Example: 1,4 → 2
    Constraints: XOR.

46. **Design Parking System**
    Description: Parking system for big, medium, small cars.
    Example: addCar(1) → true
    Constraints: Capacity management.