Happy coding! 💻

🔹 Medium Difficulty (25)

1. **Count unique characters in a string.**
   Answer: Use a set or collections.Counter to count unique chars.
   Test: `"hello"` → `4`

2. **Check if two strings are anagrams.**
   Answer: Sort both or count frequencies.
   Test: `"listen"`, `"silent"` → `True`

3. **Return the nth Fibonacci number (recursive with memo).**
   Answer: Use recursion + lru_cache or dict caching.
   Test: `n=10` → `55`

4. **Find duplicates in a list.**
   Answer: Use set to track seen and duplicates list.
   Test: `[1,2,3,2,4,3]` → `[2,3]`

5. **Flatten a nested list.**
   Answer: Use recursion or iterative stack.
   Test: `[1,[2,[3,4]],5]` → `[1,2,3,4,5]`

6. **Merge two sorted lists.**
   Answer: Two-pointer merge in O(n+m).
   Test: `[1,3,5]`, `[2,4]` → `[1,2,3,4,5]`

7. **Check if parentheses string is valid.**
   Answer: Use stack and matching pairs.
   Test: `"([])"` → `True`, `"(]"` → `False`

8. **Find middle node of a linked list.**
   Answer: Floyd's tortoise and hare.
   Test: `[1,2,3,4,5]` → `3`

9. **Remove duplicates from a linked list.**
   Answer: Use hash set.

Test: `[1,2,2,3]` → `[1,2,3]`

10. **Check if binary tree is balanced.**
    Answer: Recursively check heights ±1.
    Test: Balanced tree → `True`, skewed → `False`

11. **Inorder traversal of a binary tree.**
    Answer: Recursion or iterative with stack.
    Test: Tree `[2,1,3]` → `[1,2,3]`

12. **Level-order traversal (BFS) of binary tree.**
    Answer: Use queue.
    Test: Tree `[1,2,3,4,5]` → `[[1],[2,3],[4,5]]`

13. **Sort characters by frequency.**
    Answer: Use Counter + sort by value.
    Test: `"tree"` → `"eert"` or `"rtee"`

14. **Check if number is power of two.**
    Answer: `n>0 and n&(n-1)==0`.
    Test: `16`→`True`, `18`→`False`

15. **Reverse words in a sentence.**
    Answer: `s.split()[::-1]` and join.
    Test: `"hello world"`→`"world hello"`

16. **Rotate a matrix 90°.**
    Answer: Transpose + reverse each row.
    Test: `[[1,2],[3,4]]`→`[[3,1],[4,2]]`

17. **Spiral order of a matrix.**
    Answer: Traverse layers with four loops.
    Test: `[[1,2,3],[4,5,6],[7,8,9]]`→`[1,2,3,6,9,8,7,4,5]`

18. **Find all subsets (power set).**
    Answer: Use recursion/backtracking.
    Test: `[1,2]`→`[[],[1],[2],[1,2]]`

19. **Generate permutations.**
    Answer: Use recursion, or itertools.

Test: `[1,2]`→`[[1,2],[2,1]]`

20. **Find longest common prefix.**
    Answer: Sort strings and compare first vs last.
    Test: `["flower","flow","flight"]`→`"fl"`

21. **Implement binary search.**
    Answer: Classic mid-pointer search.
    Test: `[1,3,5,7]`, `5`→ index `2`

22. **Count islands in grid.**
    Answer: Use DFS/BFS to mark visited.
    Test: `[['1','0'],['0','1']]`→`2`

23. **Convert Roman numeral to integer.**
    Answer: Iterate, subtract when next is larger.
    Test: `"MCMIV"` → `1904`

24. **LRU cache class.**
    Answer: Use OrderedDict or linked hash map.
    Test: Chain of puts/gets evict older.

25. **Implement stack using two queues.**
    Answer: Push costly or pop costly variant.
    Test: Push 1,2; pop→`2`

🔸 Hard Difficulty (25)

26. **Word ladder length.**
    Answer: BFS over word graph.
    Test: `"hit"`, `"cog"`, `["hot","dot","dog","lot","log","cog"]`→`5`

27. **Median of two sorted arrays.**
    Answer: Binary search kth element in O(log(min(m,n))).
    Test: `[1,3]`, `[2]`→`2.0`

28. **Find smallest window containing all patterns.**
    Answer: Sliding window + counts.

Test: `s="ADOBECODEBANC"`, `t="ABC"`→`"BANC"`

29. **Word break problem.**
    Answer: DP boolean array.
    Test: `"leetcode"`, `["leet","code"]`→`True`

30. **Min path sum in grid.**
    Answer: DP accumulate along paths.
    Test: `[[1,3,1],[1,5,1],[4,2,1]]`→`7`

31. **Clone a graph.**
    Answer: DFS with hash map clones.
    Test: small adjacency test.

32. **Minimum window subsequence.**
    Answer: Two-pointer with backward trace.
    Test: `S="abcdebdde"`, `T="bde"`→`"bcde"`

33. **Find max flow (Edmonds-Karp).**
    Answer: BFS augmenting paths.
    Test: Simple capacity graph.

34. **Trapping rain water.**
    Answer: Two-pointer method.
    Test: `[0,1,0,2,1,0,1,3,2,1,2,1]`→`6`

35. **Serialize and deserialize binary tree.**
    Answer: Preorder + null markers.
    Test: Roundtrip.

36. **Maximum rectangle in histogram.**
    Answer: Monotonic stack.
    Test: `[2,1,5,6,2,3]`→`10`

37. **Longest palindromic substring.**
    Answer: Expand-around-center or Manacher’s.
    Test: `"babad"`→`"bab"` or `"aba"`

38. **Basic calculator with +,-,*,/, parentheses.**
    Answer: Two stacks or recursive descent.
    Test: `"3+(2*2)"`→`7`

39. **Alien dictionary ordering.**
    Answer: Build graph and topologically sort.
    Test: `["wrt","wrf","er","ett","rftt"]`→ valid `"wertf"`

40. **N-Queens count solutions.**
    Answer: Backtracking with columns and diagonals bitmasks.
    Test: `n=4→2`

41. **Word search II.**
    Answer: Trie + DFS in board.
    Test: board, words list.

42. **Sudoku solver.**
    Answer: Backtracking + row/col/block checks.
    Test: Provide one sample board.

43. **LFU Cache.**
    Answer: Double linked list + freq map.
    Test: Eviction pattern.

44. **Minimum edit distance.**
    Answer: DP matrix.
    Test: `"kitten"`, `"sitting"→3`

45. **Largest divisible subset.**
    Answer: DP sorting divisors.
    Test: `[1,2,4,8]→[1,2,4,8]`

46. **Concat words to form a string.**
    Answer: DP or trie.
    Test: `["cat","cats","dog","catsdog"]→True`

47. **Design hit counter in sliding window.**
    Answer: Circular buffer of timestamps.
    Test: Sequence of hits/time.

48. **Find k-th largest in stream.**
    Answer: Use min-heap of size k.
    Test: Stream `[4,5,8,2]`, `k=3`, queries.

49. **Serialize arbitrary nested list.**
    Answer: JSON-like parser.

Test: `"[123,[456,[789]]]"`.

50. **Regex matching with '.' and '*'.**
    Answer: DP with state i,j.
    Test: `s = "aab"`, `p = "c*a*b"` → `True`

✅ Example Test Case Format

Q3. Fibonacci with memo

- Input: `n = 10`
- Expected: `55`

Q15. Reverse words

- Input: `"the sky is blue"`
- Expected: `"blue is sky the"`

Happy coding! 💻