DS and Algo Qs

can you give me algorithms questions and data structures also
give me again

Sure! As you're preparing for interviews and want **100 questions per topic** for **Algorithms and Data Structures**, I'll start listing them topic by topic. Since it's a big set, I'll organize the list in chunks. Here's a plan of what I'll cover:

📘 DATA STRUCTURES (each will have 100 questions):

1. Arrays
2. Strings
3. Linked Lists
4. Stacks
5. Queues
6. Trees
7. Graphs
8. Hash Tables (Hash Maps & Sets)
9. Heaps
10. Tries

📗 ALGORITHMS (each will have 100 questions):

11. Sorting Algorithms
12. Searching Algorithms
13. Recursion & Backtracking
14. Divide and Conquer
15. Greedy Algorithms
16. Dynamic Programming
17. Graph Algorithms (BFS, DFS, Dijkstra, etc.)
18. Bit Manipulation
19. Sliding Window & Two Pointers
20. Mathematical Algorithms (GCD, Primes, etc.)

Would you like me to start from topic 1 (Arrays), or do you want questions from a specific topic first?

topic 1 (Arrays)

Perfect! Here are **100 Data Structures Interview Questions** on **Topic 1: Arrays**, organized by difficulty — from easy to hard, covering a wide range of patterns used in interviews.

🔹 Easy (1–30) – Basic Understanding & Traversals

1. Find the maximum element in an array.
2. Find the minimum element in an array.
3. Calculate the sum of all elements in an array.
4. Calculate the average of all elements in an array.
5. Reverse an array.
6. Check if an array is a palindrome.
7. Search for an element in an array (linear search).
8. Count the frequency of a given element.
9. Remove duplicates from an array (without using set).
10. Find the second largest element in an array.
11. Rotate an array by one place.
12. Left rotate an array `k` times.
13. Right rotate an array `k` times.
14. Find the element with maximum frequency.
15. Count the number of even and odd numbers.
16. Check if the array is sorted.
17. Merge two sorted arrays.
18. Sort an array of 0s and 1s.
19. Find the difference between the largest and smallest element.
20. Replace every element with the next greatest element.
21. Move all zeros to the end of the array.
22. Count how many times a number appears in an array.
23. Find the index of the first occurrence of a number.
24. Find the index of the last occurrence of a number.
25. Check if two arrays are equal.
26. Find the missing number in an array from 1 to n.
27. Merge two arrays and sort them.
28. Count how many elements are greater than a given number.

29. Find the longest consecutive sequence in an array.
30. Find the number of positive, negative, and zero elements.

🔹 Medium (31–70) – Pattern-Based & Optimized Solutions

31. Kadane's Algorithm – Maximum subarray sum.
32. Maximum product subarray.
33. Two Sum problem.
34. Find all pairs with a given sum.
35. Triplet sum in array.
36. Count pairs with a given difference.
37. Find duplicates in an array.
38. Union of two arrays.
39. Intersection of two arrays.
40. Cyclically rotate array by one.
41. Find leader elements in array.
42. Count subarrays with sum equal to `k`.
43. Find the equilibrium index.
44. Rearrange array in alternating positive & negative items.
45. Find the smallest missing positive number.
46. Find subarray with given sum.
47. Maximum length of subarray with equal 0s and 1s.
48. Count inversions in array.
49. Count number of subarrays with XOR = k.
50. Product of array except self.
51. Find majority element (appears more than n/2 times).
52. Count frequencies of all elements in O(n) time.
53. Reorder array according to indexes.
54. Maximum sum of i*arr[i] among all rotations.
55. Maximum circular subarray sum.
56. Rearrange array in max/min form.
57. Find repeating and missing number.
58. Find duplicate in array with constant space.
59. Count number of distinct elements in every window of size k.
60. Max sum of non-adjacent elements.
61. Maximum length of increasing subarray.

62. Sort array of 0s, 1s, and 2s.
63. Find all elements that appear more than ⌊n/3⌋ times.
64. Find common elements in three sorted arrays.
65. Largest subarray with 0 sum.
66. Minimum size subarray sum.
67. Maximum subarray sum after k concatenations.
68. Smallest subarray with sum > k.
69. Length of the longest arithmetic subarray.
70. Check if array can be divided into pairs with equal sum.

🔹 Hard (71–100) – Advanced Logic & Optimization

71. Merge overlapping intervals in array of intervals.
72. Median of two sorted arrays.
73. Maximum profit from stock prices (single transaction).
74. Best time to buy and sell stock – II (multiple transactions).
75. Best time to buy and sell with cooldown.
76. Best time to buy and sell with transaction fee.
77. Longest increasing subsequence (O(n log n) solution).
78. Maximum number of non-overlapping subarrays with target sum.
79. Minimum number of swaps to sort the array.
80. Maximum distance between same elements.
81. Count number of triplets with sum less than a given value.
82. Find k closest elements to a given value.
83. Count triplets forming geometric progression.
84. Median in a stream of integers.
85. Find the minimum number of operations to make all elements equal.
86. Find the minimum swaps required to bring all elements less than or equal to k together.
87. Count all subarrays having sum divisible by k.
88. Find smallest window to be sorted to make array sorted.
89. Longest subarray with at most two distinct elements.
90. Count number of nice subarrays (odd count subarrays).
91. Maximum subarray sum with at most k elements.
92. Find minimum jumps to reach the end of array.
93. Count binary subarrays with same number of 0s and 1s.
94. Longest mountain in array.

95. Trapping Rain Water problem.
96. Maximum area of histogram.
97. Find next greater element for each element.
98. Find maximum of minimums for every window size.
99. Merge k sorted arrays.
100. Find longest bitonic subarray.

Let me know if you want **solutions**, **explanations**, or want to move on to **Topic 2: Strings** next!

Topic 2: Strings

Awesome! Here's **Topic 2: Strings** with **100 interview questions**, organized from **Easy** → **Medium** → **Hard**. This list covers patterns like string manipulation, hashing, sliding window, dynamic programming, and more — ideal for coding interviews.

🔹 Easy (1–30) – Fundamentals & Basic Manipulations

1. Reverse a string.
2. Check if a string is a palindrome.
3. Convert string to uppercase/lowercase.
4. Find length of a string without using `len()`.
5. Count vowels and consonants.
6. Remove all spaces from a string.
7. Remove duplicates from a string.
8. Check if two strings are anagrams.
9. Find the first non-repeating character.
10. Find the frequency of characters.
11. Count words in a sentence.
12. Toggle each character's case.
13. Replace spaces with `%20`.
14. Find the longest word in a sentence.
15. Find the ASCII value of each character.
16. Find all substrings of a string.
17. Check if two strings are rotations of each other.
18. Remove punctuation from a string.
19. Check if string contains only digits.

20. Sort characters in a string.
21. Capitalize the first letter of each word.
22. Find common characters in two strings.
23. Find duplicate characters in a string.
24. Swap two strings without a temporary variable.
25. Count occurrences of a given word in a string.
26. Reverse each word in a sentence.
27. Check if one string is a subsequence of another.
28. Count number of palindromic substrings.
29. Convert a string to an integer.
30. Remove all numeric characters from a string.

🔹 Medium (31–70) – Pattern-Based & Optimized String Problems

31. Implement strstr() function (find substring).
32. Implement `**atoi()**` function (string to int).
33. Longest common prefix among a list of strings.
34. Longest Palindromic Substring (expand around center).
35. Length of longest substring without repeating characters.
36. Group anagrams from an array of strings.
37. Find smallest window in a string containing all characters of another string.
38. Check if string is valid shuffle of two strings.
39. Print all permutations of a string.
40. Rabin-Karp Algorithm for substring search.
41. KMP Algorithm (Pattern Searching).
42. Manacher's Algorithm (for longest palindrome in linear time).
43. Encode and decode strings (like Leetcode-style format).
44. Minimum number of flips to make binary string alternate.
45. Longest repeating character replacement.
46. Compare version numbers given as strings.
47. Implement basic string compression (e.g., "aaabb" → "a3b2").
48. Find all repeated substrings of length `**k**`.
49. Find first recurring character in a string.
50. Check if string can be segmented into dictionary words.
51. Longest word that can be built one character at a time by other words.

52. Check if a string follows a pattern (like word pattern match).
53. Minimum characters to add to make string palindrome.
54. Lexicographically smallest substring.
55. Implement run-length encoding.
56. Multiply two numeric strings.
57. Validate IP address (IPv4 & IPv6).
58. Zigzag conversion of string.
59. Roman to Integer conversion.
60. Integer to Roman conversion.
61. Decode ways (like "111" → 3 ways to decode).
62. Count and say sequence generator.
63. Add binary strings.
64. Remove adjacent duplicates in string.
65. Check for balanced parentheses.
66. Check if string is valid palindrome ignoring non-alphanumerics.
67. Generate all combinations of balanced parentheses.
68. Check if two strings are meta strings (can be made equal with one swap).
69. Find longest prefix which is also a suffix.
70. Replace every character with its next character in alphabet.

🔹 Hard (71–100) – Advanced Concepts & Optimizations

71. Longest palindromic subsequence.
72. Longest common subsequence.
73. Edit Distance (min operations to convert one string to another).
74. Wildcard pattern matching with `*` and `?`.
75. Regular Expression Matching (with `.` and `*`).
76. Count number of distinct substrings using Trie.
77. Count all palindromic substrings using DP.
78. Shortest common supersequence.
79. Find the kth non-repeating character.
80. Minimum insertions to make a string palindrome.
81. Find shortest string that contains both strings as subsequences.
82. Number of ways to decode a binary message.
83. Longest duplicated substring.

84. Minimum window to make two strings anagrams.
85. Count number of distinct subsequences.
86. Binary string to decimal conversion and vice versa.
87. Lexicographically next greater permutation of a string.
88. Minimum number of swaps to make binary string alternating.
89. Number of matching subsequences.
90. Find string after backspace (like typing in an editor).
91. Remove all adjacent k duplicates in string.
92. Multiply large numbers represented as strings.
93. Implement strstr() using rolling hash.
94. Find the smallest substring containing all unique characters.
95. Word Ladder – shortest transformation sequence.
96. Palindrome partitioning (all possible ways).
97. Minimum cuts to make a string palindrome.
98. Longest substring with at most k distinct characters.
99. Compare strings considering backspaces.
100. Find minimum edit distance with insert/delete/replace.

Let me know if you want:

- Solutions or code for any of these,
- The **next topic (Linked Lists)**,
- Or a random mix of practice questions.

Ready when you are!

> topic (Linked Lists),

Great! Here's **Topic 3: Linked Lists** with **100 interview questions**, organized by **Easy** → **Medium** → **Hard**. This covers all the essential operations, two-pointer techniques, recursion, and pointer manipulation questions that are frequently asked in top coding interviews.

🔹 Easy (1–30) – Basics, Traversal, and Construction

1. What is a linked list?
2. Create a singly linked list.
3. Traverse a linked list.

4. Insert a node at the beginning.
5. Insert a node at the end.
6. Insert a node at a given position.
7. Delete a node from the beginning.
8. Delete a node from the end.
9. Delete a node at a given position.
10. Find the length of the linked list (iteratively).
11. Find the length of the linked list (recursively).
12. Search for an element in a linked list.
13. Find the middle of a linked list.
14. Reverse a linked list (iteratively).
15. Reverse a linked list (recursively).
16. Count occurrences of an element.
17. Check if a linked list is circular.
18. Check if a linked list is a palindrome.
19. Compare two linked lists for equality.
20. Remove duplicates from an unsorted list.
21. Remove duplicates from a sorted list.
22. Print the nth node from the beginning.
23. Print the nth node from the end.
24. Find the intersection point of two lists.
25. Merge two sorted linked lists.
26. Detect a loop in a linked list (Floyd's cycle detection).
27. Remove a loop from a linked list.
28. Clone a linked list with next and random pointer.
29. Swap two nodes (not data).
30. Delete a node without head pointer (given only the node).

🔹 Medium (31–70) – Two-pointer, Recursion, Optimization

31. Sort a linked list using merge sort.
32. Detect and remove cycle in a list.
33. Check if the linked list is palindrome (using stack).
34. Rotate a linked list by `k` nodes.
35. Flatten a multilevel doubly linked list.

36. Add two numbers represented by linked lists.
37. Merge k sorted linked lists.
38. Find intersection node of two lists (efficient method).
39. Segregate even and odd nodes.
40. Rearrange linked list in zigzag fashion.
41. Rearrange list: place odd index nodes after even ones.
42. Split circular linked list into two halves.
43. Clone a linked list with arbitrary/random pointers.
44. Find the starting point of loop in linked list.
45. Move last element to front of a linked list.
46. Pairwise swap elements of linked list.
47. Reverse a linked list in groups of `k`.
48. Remove all nodes with duplicate numbers (Leetcode 82).
49. Add 1 to a number represented by linked list.
50. Convert a binary number in a linked list to integer.
51. Check if two lists are identical.
52. Find modular node (k-th node from end if length divisible by k).
53. Reverse alternate `k` nodes in a linked list.
54. Multiply two numbers represented by linked lists.
55. Remove loop using hashing.
56. Delete nodes which have a greater value on right.
57. Clone linked list using hashing.
58. Swap kth node from beginning with kth node from end.
59. Find the first non-repeating character in a stream (using LL).
60. Flatten a linked list with child pointers.
61. Convert sorted linked list to balanced BST.
62. Find next greater node for each node.
63. Sort 0s, 1s, and 2s in a linked list.
64. Find length of loop if present.
65. Check if linked list is length even or odd.
66. Find triplet in three linked lists with given sum.
67. Check if linked list is palindrome using recursion.
68. Insert node in a sorted linked list.
69. Detect intersection without extra space.
70. Remove every kth node from linked list.

🔹 Hard (71–100) – Advanced Pointers, Complex Operations

71. Deep copy a linked list with arbitrary pointers (O(1) space).
72. Reverse nodes in pairs (recursively).
73. Copy linked list with both `next` and `random` pointers.
74. Implement LRU Cache using linked list and hashmap.
75. Merge alternate nodes of two linked lists.
76. Flatten a linked list where each node has a `next` and `down` pointer.
77. Add numbers represented by linked list (recursive).
78. Reorder list as L0 → Ln → L1 → Ln-1 ...
79. Delete N nodes after every M nodes.
80. Sort a linked list of 0s, 1s and 2s without counting.
81. Flatten sorted linked lists into one.
82. Partition linked list around a value (like quick sort pivot).
83. Convert a doubly linked list to balanced BST.
84. Clone complex linked list with `random` pointer in O(1) space.
85. Reverse sublist between position `m` and `n`.
86. Reverse the entire list recursively using tail recursion.
87. Rotate linked list to the right by `k` places.
88. Swap every pair of adjacent nodes.
89. Detect palindrome in O(1) extra space and O(n) time.
90. Convert linked list to a complete binary tree.
91. Delete all duplicates such that no element appears more than once.
92. Find sum of two linked lists stored in reverse order.
93. Split a list into k parts (as equal as possible).
94. Merge k sorted linked lists using min-heap.
95. Count pairs from two lists whose sum is equal to a target.
96. Check if loop exists and return its length.
97. Find merge point of two linked lists without extra space.
98. Detect loop using visited node count method.
99. Reverse every alternate `k` nodes in linked list.
100. Convert a linked list into a number and vice versa.

Let me know if you'd like:

- **Solutions** or **code** for any,

- To move on to **Topic 4: Stacks**,
- Or combine topics for practice sets!

You're doing awesome — keep going! 💪

Topic 4: Stacks,

Nice! Here's **Topic 4: Stacks** with **100 Interview Questions**, organized from **Easy** → **Medium** → **Hard**. These questions cover core stack operations, variations (min-stack, max-stack), expression problems, monotonic stacks, and classic interview patterns.

🔹 Easy (1–30) – Stack Basics & Standard Operations

1. Implement a stack using an array.
2. Implement a stack using a linked list.
3. Push, pop, peek, and isEmpty operations.
4. Find the size of a stack.
5. Reverse a stack using recursion.
6. Sort a stack using recursion.
7. Delete the middle element of a stack.
8. Implement two stacks in one array.
9. Check for balanced parentheses.
10. Convert infix to postfix expression.
11. Convert infix to prefix expression.
12. Convert postfix to infix expression.
13. Convert prefix to infix expression.
14. Evaluate a postfix expression.
15. Evaluate a prefix expression.
16. Check if a string is a valid parentheses string.
17. Remove adjacent duplicates from a string using a stack.
18. Design a stack that supports getMin() in O(1).
19. Design a stack that supports getMax() in O(1).
20. Find next greater element for each element in array.
21. Find next smaller element for each element in array.
22. Find previous greater element.
23. Find previous smaller element.

24. Implement a stack using queues.
25. Implement a queue using stacks.
26. Check if an expression has duplicate parentheses.
27. Find the length of the longest valid parentheses substring.
28. Insert at the bottom of the stack using recursion.
29. Check if a given sequence is a stack permutation.
30. Find the minimum element in a stack without extra space.

🔹 Medium (31–70) – Application-Based & Pattern Questions

31. Celebrity problem using stack.
32. Stock span problem.
33. Design a special stack that returns minimum in O(1) time and space.
34. Largest rectangle in a histogram.
35. Max area in binary matrix (based on histogram).
36. Implement k stacks in a single array.
37. Implement a MinStack with constant space.
38. Implement MaxStack.
39. Remove all consecutive duplicates in string k times.
40. Decode a string with nested brackets (e.g., `` `"3[a2[c]]"` ``).
41. Clone a stack without affecting the original.
42. Implement browser forward/backward functionality using stacks.
43. Evaluate expression with parentheses and operators.
44. Redundant brackets check.
45. Implement an iterator using stacks (BST iterator).
46. Reverse a string using stack.
47. Calculate span of stocks over `` `n` `` days.
48. Decode string with numbers and brackets.
49. Implement getMinStack using auxiliary stack.
50. Build expression tree from postfix expression.
51. Print stack from top to bottom without modifying it.
52. Count number of reversals to balance a bracket expression.
53. Minimum insertions to balance parentheses.
54. Calculate minimum number of brackets to be removed.
55. Flatten a nested list using stack.

56. Evaluate arithmetic expression (with +, -, *, /, parentheses).
57. Design circular stack with fixed capacity.
58. Calculate minimum cost to balance a bracket expression.
59. Find max frequency stack (like Leetcode FreqStack).
60. Sum of minimums of all subarrays.
61. Find next greater element in circular array.
62. Reverse a stack using two stacks.
63. Find maximum of minimum for every window size.
64. Longest valid parentheses using stack.
65. Find the largest rectangle under histogram using stack.
66. Implement LRU cache using Stack + Map.
67. Convert decimal to binary using stack.
68. Check if the given push and pop operations are valid for a stack.
69. Implement Undo/Redo operations using stacks.
70. Check if a string of parentheses is balanced and properly nested.

🔹 Hard (71–100) – Monotonic Stacks, Optimization, Design

71. Sliding window maximum using stack.
72. Maximum rectangle in binary matrix (Leetcode 85).
73. Trapping Rain Water using stack.
74. Daily temperatures (monotonic stack).
75. Asteroid collision problem.
76. Remove k digits to make smallest number.
77. Maximum score of a good subarray.
78. Find all next greater permutations of elements in circular array.
79. Build a calculator with operator precedence using stacks.
80. Sum of subarray minimums (monotonic stack).
81. Find largest rectangle with all 1s in binary matrix.
82. Online stock span with real-time queries.
83. Minimum number of adjacent swaps to make string palindrome.
84. Build stack that can get second minimum in O(1).
85. Stack with get middle operation in O(1).
86. Implement stack that can return frequency of element in O(1).
87. Largest rectangle in skyline histogram (variable width).

88. Remove duplicate letters to get smallest lexicographical string.
89. Generate all valid parentheses using stack.
90. Find max depth of nested parentheses.
91. Evaluate ternary expression using stack.
92. Decode nested encoded strings (like `"3[a2[bc]]"`).
93. Implement design of a text editor using stack (undo, redo).
94. Track longest valid parentheses substring with O(1) space.
95. Convert N-ary tree to expression using stack traversal.
96. Make string valid by removing minimal brackets.
97. Decode compressed strings using stack.
98. Min stack using single stack and integer math.
99. Stack-based pattern matching (similar to parsing).
100. Calculate exclusive time of functions using stack logs.

Let me know if you'd like:

- Any **solutions**, **code**, or **Python-based stack problems**,
- To move on to **Topic 5: Queues**,
- Or need mixed sets for practice!

You're crushing it! 🔥

> Topic 5: Queues,

Let’s gooo! Here's **Topic 5: Queues** with **100 interview questions**, organized from **Easy** → **Medium** → **Hard**. Covers standard queues, circular queues, deque, monotonic queues, priority queues, heaps, and design-based problems — all the hot stuff for interviews.

🔹 Easy (1–30) – Queue Basics & Operations

1. What is a queue?
2. Implement a queue using array.
3. Implement a queue using linked list.
4. Implement enqueue and dequeue operations.
5. Implement peek and isEmpty.
6. Find the size of a queue.
7. Reverse a queue using recursion.

8. Reverse a queue using stack.
9. Generate binary numbers from 1 to N using queue.
10. Interleave the first half of the queue with the second half.
11. Check for palindrome using queue.
12. Find the first non-repeating character in a stream.
13. Implement a circular queue.
14. Print first negative integer in every window of size k.
15. Implement queue using two stacks.
16. Implement stack using two queues.
17. Sum of all elements in a queue without using loop.
18. Display queue elements without modifying the queue.
19. Rotate queue by k elements.
20. Count number of elements in a queue without using size().
21. Reverse first k elements of a queue.
22. Check if two queues are equal.
23. Merge two queues into one.
24. Implement dequeue (double-ended queue).
25. Find max in each sliding window of size k.
26. Delete middle element of a queue.
27. Find frequency of an element in queue.
28. Convert a queue into a stack.
29. Implement queue using singly linked list.
30. Implement priority queue using list.

🔹 Medium (31–70) – Applications, Sliding Window, Priority Queues

31. Design circular queue using array.
32. Design queue with getMin() operation.
33. Implement LRU Cache using queue + hashmap.
34. Implement LFU Cache using queue + hashmap.
35. Rearrange characters so that no two adjacent are same.
36. Sum of maximum and minimum elements of all subarrays of size k.
37. Find first circular tour that visits all petrol pumps.
38. Circular tour with gas and cost arrays (Leetcode Gas Station).

39. Rotten oranges problem using BFS and queue.
40. Distance of nearest cell having 1 in binary matrix.
41. Shortest path in binary matrix (BFS with queue).
42. Knight's minimum steps in chessboard (BFS).
43. Snake and Ladder problem using queue.
44. Find maximum of all subarrays of size k.
45. Sliding window maximum using deque.
46. Sum of minimums of all subarrays using deque.
47. Implement a monotonic increasing queue.
48. Implement a monotonic decreasing queue.
49. Design a Hit Counter (number of hits in past 5 minutes).
50. Serialize and deserialize binary tree using queue.
51. Binary tree level order traversal.
52. Zigzag traversal of binary tree.
53. Right side view of binary tree.
54. Left side view of binary tree.
55. Bottom view of binary tree.
56. Top view of binary tree.
57. Connect nodes at same level using queue.
58. Clone graph using BFS and queue.
59. Shortest path in unweighted graph using BFS.
60. Course Schedule (topological sort with queue).
61. Alien Dictionary (topological sort with queue).
62. Task Scheduler (Leetcode 621 – with gaps between tasks).
63. Kth largest element in stream using priority queue.
64. Design a priority queue from scratch.
65. Implement min-heap and max-heap using priority queue.
66. Merge k sorted lists using min-heap.
67. Find median from a stream using two heaps.
68. Reduce array to half using min operations (priority queue).
69. Top K frequent elements using heap.
70. Sort a k-sorted array using priority queue.

🔹 Hard (71–100) – Advanced Queue Problems, Design, Optimization

71. Minimum number of swaps for bracket balancing using queue.
72. Maximum sum of all subarrays of size k.
73. Find all nodes at distance K from a target node (BFS).
74. Find shortest bridge in a binary matrix using queue.
75. Number of islands using BFS and queue.
76. Binary tree vertical order traversal.
77. Implement K queues in a single array.
78. Maximum width of a binary tree using queue.
79. Minimum time to burn a binary tree from a leaf node.
80. Shortest distance from all buildings (BFS from multiple sources).
81. Count number of islands after each operation (dynamic BFS).
82. Multi-source BFS to spread fire in grid.
83. Find minimum height trees (using BFS from leaves).
84. Find cheapest flights with up to K stops (Priority Queue + BFS).
85. Swim in rising water (Priority Queue BFS).
86. Network Delay Time (Dijkstra with Priority Queue).
87. Build a stream of characters (autocomplete using queue/trie).
88. Water Jug Problem (classic BFS).
89. Trapping Rain Water II (BFS with min-heap).
90. Minimum knight moves on chessboard (BFS).
91. Longest subarray with at most k distinct elements using queue.
92. Serialize and deserialize N-ary tree using queue.
93. Find shortest path in maze using queue.
94. Implement bounded blocking queue (multi-threaded).
95. Flatten nested list using queue.
96. Sliding window maximum with lazy deletion.
97. Dynamic median of numbers (stream processing).
98. Move all 0s to end using queue approach.
99. Word Ladder II – return all shortest paths (multi-level BFS).
100. Reconstruct itinerary using priority queue and graph traversal.

Let me know if you'd like:

- **Python implementations** for any,
- Next up: **Topic 6: Trees** 🌳,
- Or practice questions mixed from multiple topics!

You're building a killer prep set! 💥

Topic 6: Trees

Boom! Here's **Topic 6: Trees** with **100 Interview Questions**, organized by **Easy** → **Medium** → **Hard**. This includes binary trees, binary search trees, traversals, recursion, views, and advanced problems like tree DP and serialization. Tree problems are must-haves for coding interviews. 🌳🔥

🔹 Easy (1–30) – Basics, Traversals, and Recursion

1. What is a binary tree?
2. Implement inorder traversal (recursive).
3. Implement preorder traversal (recursive).
4. Implement postorder traversal (recursive).
5. Implement inorder traversal (iterative).
6. Implement preorder traversal (iterative).
7. Implement postorder traversal (iterative).
8. Level order traversal using queue.
9. Find height/depth of a binary tree.
10. Count number of nodes in binary tree.
11. Count number of leaf nodes.
12. Find maximum element in a binary tree.
13. Search an element in a binary tree.
14. Check if two trees are identical.
15. Check if a tree is symmetric.
16. Mirror a binary tree.
17. Construct binary tree from inorder & preorder.
18. Construct binary tree from inorder & postorder.
19. Diameter of binary tree (longest path).
20. Sum of all nodes in binary tree.
21. Path sum exists in binary tree (root to leaf).
22. Count all paths with a given sum.
23. Find minimum depth of binary tree.
24. Print all root-to-leaf paths.
25. Convert binary tree to its mirror.

26. Find LCA (Lowest Common Ancestor) of two nodes.
27. Check if a tree is balanced (height-balanced).
28. Right view of a binary tree.
29. Left view of a binary tree.
30. Top view of a binary tree.

🔹 Medium (31–70) – Binary Search Trees, Views, Serialization

31. What is a binary search tree (BST)?
32. Insert a node in BST.
33. Search in BST.
34. Delete a node from BST.
35. Validate if a tree is a BST.
36. Convert sorted array to balanced BST.
37. Convert sorted linked list to balanced BST.
38. Lowest common ancestor in BST.
39. Find kth smallest element in BST.
40. Find kth largest element in BST.
41. Floor and Ceil in BST.
42. Trim BST within a given range.
43. Convert BST to Greater Sum Tree.
44. Convert BST to Doubly Linked List (DLL).
45. Find inorder successor in BST.
46. Find inorder predecessor in BST.
47. Vertical order traversal.
48. Zigzag (spiral) level order traversal.
49. Diagonal traversal of binary tree.
50. Boundary traversal of binary tree.
51. Serialize and deserialize a binary tree (level order).
52. Serialize and deserialize a binary tree (preorder).
53. Build tree from string with brackets.
54. Check if one tree is subtree of another.
55. Flatten binary tree to linked list (preorder).
56. Binary tree to DLL (inorder based).
57. Print ancestors of a given node.

58. Find distance between two nodes.
59. Count full nodes, half nodes, leaf nodes.
60. Maximum sum path between two leaves.
61. Maximum sum path from any node to any node.
62. Check if binary tree is complete.
63. Boundary traversal in anti-clockwise.
64. Binary tree from string with brackets.
65. Replace node with sum of its inorder predecessor and successor.
66. Delete leaf nodes with given value recursively.
67. All nodes distance K from target node.
68. Find duplicate subtrees.
69. Clone a binary tree with random pointer.
70. Count nodes equal to average of subtree.

🔹 Hard (71–100) – Tree DP, Optimizations, Advanced Design

71. Maximum width of a binary tree.
72. Longest ZigZag path in binary tree.
73. Find nodes at k distance from root.
74. Tree diameter using single DFS.
75. Morris inorder traversal (no stack/recursion).
76. Morris preorder traversal.
77. Recover BST where two nodes are swapped.
78. Check if binary tree is height-balanced in O(n).
79. Count number of unique BSTs with `n` nodes (Catalan number).
80. Construct BST from preorder.
81. Construct BST from postorder.
82. BST iterator (inorder traversal as iterator).
83. Largest BST in binary tree.
84. Distance between two nodes in BST.
85. All possible full binary trees with n nodes.
86. Count all nodes in complete binary tree in $O(\log^2 n)$.
87. Find smallest subtree with all deepest nodes.
88. Maximum path sum between any two nodes.
89. Longest consecutive sequence path in binary tree.

90. House Robber III (tree version).
91. Cameras in a binary tree (minimum number).
92. Tree coloring problem.
93. Delete a binary tree (deallocate memory).
94. Check whether two trees are mirror of each other.
95. Count paths with sum = K (efficient version).
96. Tree isomorphism problem.
97. Diameter of N-ary tree.
98. Construct N-ary tree from parent-child array.
99. Serialize/Deserialize N-ary tree.
100. Multi-source LCA using Binary Lifting (advanced).

Let me know if you want:

- **Python solutions** for any of these,
- Next up: **Topic 7: Graphs**,
- Or a **custom problem set** across Trees + other DS topics.

You're crushing these topics like a boss! 💻🌲🔥

> Topic 7: Graphs

Awesome! Here's **Topic 7: Graphs** with **100 Interview Questions**, ranging from **Easy** → **Medium** → **Hard**. These include all the essentials: representations, BFS/DFS, shortest paths, cycles, connected components, topological sort, MST, and advanced algorithms like Union-Find, Dijkstra, Bellman-Ford, Floyd-Warshall, etc. 🧠⚡

🔹 Easy (1–30) – Graph Representation & Traversal Basics

1. What is a graph?
2. Difference between directed and undirected graphs.
3. Represent a graph using an adjacency list.
4. Represent a graph using an adjacency matrix.
5. Implement BFS traversal (iterative).
6. Implement DFS traversal (recursive).
7. Implement DFS traversal (iterative).
8. Detect a cycle in an undirected graph using DFS.
9. Detect a cycle in an undirected graph using BFS.

10. Check if a graph is connected.
11. Find number of connected components in undirected graph.
12. Check if a path exists between two nodes (DFS).
13. Convert edge list to adjacency list.
14. Convert adjacency matrix to adjacency list.
15. Find degree of each node in undirected graph.
16. Print all paths between two vertices in a graph.
17. Check if a graph is bipartite using BFS.
18. Check if a graph is bipartite using DFS.
19. Count nodes reachable from a given node.
20. Detect cycle in directed graph using DFS.
21. Find the mother vertex in a graph.
22. Check if a graph has a valid path.
23. Implement topological sort using DFS.
24. Implement topological sort using Kahn's Algorithm (BFS).
25. Find in-degree and out-degree of nodes.
26. Check if graph is a tree.
27. Create transpose of a directed graph.
28. BFS level order traversal of graph (shortest path in unweighted).
29. DFS postorder and preorder numbering.
30. Check whether graph is Eulerian or Semi-Eulerian.

🔹 Medium (31–70) – Shortest Paths, Components, Cycles

31. Dijkstra's Algorithm (with priority queue).
32. Bellman-Ford Algorithm.
33. Floyd-Warshall Algorithm.
34. Find shortest path in an unweighted graph.
35. Find shortest path from a source to all vertices.
36. Find all nodes at distance K from a given node.
37. Word Ladder (shortest transformation sequence).
38. Course Schedule (topological sort + cycle detection).
39. Course Schedule II (return valid ordering).
40. Number of provinces / connected components.
41. Clone an undirected graph.

42. Reconstruct itinerary from tickets.
43. Alien Dictionary (topological sort).
44. Find if there is a cycle in prerequisites graph.
45. Find strongly connected components (Kosaraju's Algo).
46. Tarjan's Algorithm to find SCC.
47. Bridges in a graph (Tarjan's Algo).
48. Articulation points in a graph.
49. Number of islands in a 2D grid (graph version).
50. Number of distinct islands (with shapes).
51. Find redundant connection (cycle in undirected using Union-Find).
52. Find redundant directed connection.
53. Detect cycle using Union-Find (Disjoint Set Union).
54. Union-Find with path compression.
55. Kruskal's Algorithm for MST.
56. Prim's Algorithm for MST.
57. Minimum cost to connect all points (MST).
58. Network Delay Time (Dijkstra variation).
59. Cheapest flights with up to K stops (modified BFS/Dijkstra).
60. Find eventual safe states (topological sort in reverse).
61. Is graph bipartite (BFS coloring).
62. Bipartite graph check in disconnected graph.
63. All paths from source to target (backtracking/DFS).
64. Reorder routes to make all paths lead to city 0.
65. Longest path in a DAG.
66. Detect negative weight cycle using Bellman-Ford.
67. Snake and Ladder shortest path using BFS.
68. Find all critical connections in a network.
69. Path with maximum probability (BFS with priority queue).
70. Max area of island using DFS/BFS.

🔹 Hard (71–100) – Advanced Graph Algorithms & Design

71. Disjoint Set Union (DSU) with union by rank + path compression.
72. Kruskal's algorithm with DSU.
73. Minimum swaps to make graph connected.

74. Detect cycle in directed graph using color array.
75. Maximum number of edges that can be removed to make graph bipartite.
76. Lexicographically smallest topological ordering.
77. Hamiltonian Path (Backtracking).
78. Traveling Salesman Problem (TSP – DP + Bitmask).
79. Eulerian Path/Circuit in directed graph.
80. Minimize malware spread in network.
81. Find whether a cycle exists of even length.
82. Kth shortest path from source to target.
83. Design Twitter – use graph and heap.
84. Design LinkedIn (mutual connections, degree of separation).
85. Find minimum number of buses to destination.
86. Minimum steps to reach end of array with teleportation (graph model).
87. Find number of islands with DFS, BFS, and Union-Find.
88. Minimum cost to supply water to all houses (MST).
89. Reconstruct original array from pairwise sums.
90. Shortest path in a weighted maze (Dijkstra).
91. Minimum cost path in grid with direction.
92. Build and evaluate division equations (graph with weights).
93. Swim in rising water (priority queue BFS).
94. Jump Game IV (shortest path with value-graph BFS).
95. Count total number of triangles in an undirected graph.
96. Graph coloring problem (m-coloring).
97. Knight's shortest path on chessboard.
98. Bi-directional BFS for shortest path.
99. Design a graph class with addEdge(), removeEdge(), isConnected().
100. Kahn's Algorithm with multiple valid topological orders.

Let me know if you'd like:

- Python implementations for any of these,
- Graph **visualizations**,
- Move on to **Topic 8: Heaps & Priority Queues**,
- Or mixed sets for daily practice!

Graph mastery = interviewer magnet. You're killin' it! 🚀📊

Topic 8: Heaps & Priority Queues,

Yes! Here's **Topic 8: Heaps & Priority Queues** with **100 Interview Questions**, organized by **Easy → Medium → Hard**. This covers all types: **Min Heap**, **Max Heap**, **Custom Comparators**, **Priority Queues**, and problems involving **Top K**, **Sliding Windows**, **Stream Processing**, and **Heap + Greedy** combos.

🔹 Easy (1–30) – Heap Basics, Implementation, and Operations

1. What is a heap?
2. Difference between min-heap and max-heap.
3. How is a heap represented in an array?
4. Build a min-heap from an array.
5. Build a max-heap from an array.
6. Insert an element into a min-heap.
7. Delete the minimum element from a min-heap.
8. Heapify up vs heapify down.
9. Check if an array is a min-heap.
10. Convert max-heap to min-heap.
11. Convert min-heap to max-heap.
12. Peek the minimum element in a min-heap.
13. Peek the maximum element in a max-heap.
14. Implement a min-heap from scratch.
15. Implement a max-heap from scratch.
16. Use Python `heapq` to implement min-heap.
17. Simulate max-heap using `heapq` in Python.
18. Implement a priority queue using heap.
19. Difference between heap and priority queue.
20. Check if two heaps are equal.
21. Merge two heaps into one.
22. Convert binary tree to binary heap.
23. Convert BST to heap.
24. Implement isHeap utility to validate heap.
25. Build heap in O(n) using Floyd's method.
26. Compare performance: insert vs buildHeap.
27. Find the second smallest element in a min-heap.

28. Find the second largest element in a max-heap.
29. Implement d-ary heap (not binary).
30. Track median of stream using two heaps.

🔹 Medium (31–70) – Top K, Stream, Greedy + Heap

31. Find the K largest elements in an array.
32. Find the K smallest elements in an array.
33. Kth largest element in an array.
34. Kth smallest element in an array.
35. Kth largest element in a stream.
36. Kth smallest pair distance.
37. Find median in a data stream.
38. Top K frequent elements.
39. Top K frequent words.
40. Top K most frequent numbers in a stream.
41. Find K closest numbers to a target.
42. Find K closest points to origin.
43. Merge K sorted arrays using min-heap.
44. Merge K sorted linked lists using min-heap.
45. Sort a nearly sorted (K-sorted) array.
46. Sort characters by frequency using heap.
47. Rearrange characters so that no two adjacent are same.
48. Rearrange string k distance apart.
49. Frequency sort of array using heap.
50. Reduce array size to half using min operations.
51. Maximize score after splitting array.
52. Minimize deviation in array using max-heap.
53. Minimize cost of connecting ropes.
54. Minimum sum of absolute differences of pairs.
55. Minimize the maximum difference between heights.
56. Smallest range covering elements from K lists.
57. Task Scheduler using heap.
58. Last Stone Weight problem.
59. Find the Kth largest element in a matrix.

60. Find K pairs with smallest sums.
61. Find maximum in each sliding window of size K.
62. Sliding window median.
63. Sliding window maximum with lazy deletion.
64. Buy tickets with priority order (like hospital queue).
65. Process tasks based on earliest arrival time.
66. Simulate CPU scheduling with priority.
67. Find smallest missing positive integer using heap.
68. Furthest building you can reach using ladders and bricks.
69. Minimum operations to halve array sum.
70. Find top K frequent elements in a dynamic stream.

🔹 Hard (71–100) – Advanced Heap Problems, Design, Optimization

71. Design a data structure to support insert, delete, getMin.
72. Design a heap that supports decreaseKey.
73. Design an efficient priority queue with updateKey.
74. Dynamic median finder (insert + getMedian in O(log n)).
75. Trapping Rain Water II using min-heap.
76. Swim in rising water (priority queue BFS).
77. Network delay time (Dijkstra using min-heap).
78. Minimum cost to connect all points (Prim's MST with heap).
79. Cheapest flights within K stops (min-heap + BFS).
80. Smallest number from digits with swaps allowed (heap + BFS).
81. Construct target array with sum using max-heap.
82. Path with maximum probability (Dijkstra variant with max-heap).
83. Maximize score after n operations (state compression + heap).
84. Kth largest sum of contiguous subarrays.
85. Merge files with minimum cost (Huffman coding style).
86. Huffman encoding using priority queue.
87. Design Twitter feed using heap.
88. Design Task Scheduler with cooldown using heap.
89. Find skyline of buildings (event-based heap problem).
90. Allocate minimum number of platforms using heap.

91. Find maximum capital with K projects.
92. Seat reservation manager (min-heap based).
93. Maximize distance to closest person (heap variant).
94. Design data structure for constant time insert/delete/getMax.
95. Remove duplicates keeping highest frequency (heap+dict).
96. Fractional Knapsack using max-heap.
97. Maximum product of k integers in array.
98. Minimum cost to split gold bar (greedy + min-heap).
99. Simulate LRU cache using heap (with timestamps).
100. Minimize sum of products of pairs from two arrays.

Let me know if you want:

- ✅ **Python implementations** for any question,
- 🔁 Next: **Topic 9: Tries**,
- 💪 Or a **daily random challenge** mix from all topics.

You're absolutely crushing these interview preps! 🔥🧠

> Topic 9: Tries,
>
> 💪 Or a daily random challenge mix from all topics.

Let's go! Here's **Topic 9: Tries** with **100 Interview Questions**, progressing from **Easy** → **Medium** → **Hard**. Tries (Prefix Trees) are essential for string manipulation, autocomplete, dictionary matching, word search, and bitwise operations.

🔹 Easy (1–30) – Trie Basics & Operations

1. What is a Trie?
2. Insert a word into a Trie.
3. Search for a word in a Trie.
4. Check if any word starts with a given prefix.
5. Delete a word from a Trie.
6. Design a Trie class with insert, search, startsWith.
7. Count total words in a Trie.
8. Count total nodes in a Trie.
9. Find the longest prefix of a word in Trie.

10. List all words stored in a Trie.
11. Implement Trie using a dictionary.
12. Case-insensitive Trie implementation.
13. Support wildcard search with '.' in Trie.
14. Count how many words start with a given prefix.
15. Return all words with a given prefix.
16. Build Trie from a dictionary of words.
17. Find word in Trie with given prefix and suffix.
18. Return word list in lexicographical order.
19. Autocomplete feature using Trie.
20. Replace words with their root forms using Trie.
21. Implement word filter with prefix and suffix matching.
22. Remove prefix from all matching words in Trie.
23. Remove suffix from all matching words in Trie.
24. Find if Trie is empty.
25. Convert a list of strings into a Trie.
26. Display Trie structure visually (nested format).
27. Check if two Tries are identical.
28. Reverse Trie (for suffix matching).
29. Build a Trie from reversed words.
30. Return all palindromic words from Trie.

🔹 Medium (31–70) – Word Search, Prefix/Suffix, Compression

31. Word Search in 2D board using Trie.
32. Find all words in board using Trie and backtracking.
33. Count number of words matching a pattern.
34. Search suggestions system.
35. Magic dictionary (edit distance 1 check).
36. Add and search word with wildcard.
37. Longest word in dictionary that can be built one letter at a time.
38. Implement suffix Trie.
39. Build a compressed Trie (radix tree).
40. Shortest unique prefix for every word.
41. Group words with common prefixes.

42. Count prefix occurrences in stream.
43. Dynamic stream checker with Trie.
44. Find words that can be formed from characters.
45. Longest common prefix using Trie.
46. Minimum length encoding of words (reverse Trie).
47. Maximize XOR of two numbers in an array (bit Trie).
48. Find max XOR of prefix and suffix.
49. Count distinct substrings using Trie.
50. Find number of palindrome pairs in word list using Trie.
51. Find all words that match a CamelCase pattern.
52. Find all contact names matching prefix.
53. Detect if one string is a prefix of another.
54. Word transformation using Trie.
55. Suffix Trie vs Suffix Array.
56. Trie-based spell checker (suggestions).
57. Replace characters in word using Trie rules.
58. Minimum number of deletions to make prefix-free set.
59. Find all prefixes of a string in Trie.
60. Build Trie from string and count matching prefixes.
61. Print all pairs of anagram words using Trie.
62. Find all compound words in a list.
63. Return shortest root that replaces word in sentence.
64. Find all k-length substrings with duplicates using Trie.
65. Create Trie from all substrings of a string.
66. Count strings that differ by one character.
67. Minimum cost of deleting duplicate words (Trie + DP).
68. Identify inconsistent entries (where prefix is another entry).
69. List of contacts (phone directory Trie).
70. Validate autocomplete system with Trie.

🔹 Hard (71–100) – Bitwise, Dynamic Trie, Trie + Greedy/DP

71. Max XOR of two numbers using Bitwise Trie.
72. Max XOR between prefix and suffix using Trie.
73. Find minimum XOR pair in array using Trie.

74. Longest common suffix in a list of words.
75. Count pairs with XOR less than K.
76. Count number of unique substrings in a string.
77. Count prefix-free sets using Trie.
78. Count number of palindrome paths in Trie.
79. Minimum insertions to make a word prefix-free.
80. Find smallest string not present in Trie.
81. Lexicographically smallest word not in dictionary.
82. Find if any word is a concatenation of two dictionary words.
83. Minimum cost to merge words into Trie.
84. Check if all prefixes of a word are in the Trie.
85. Count number of common prefixes between two Tries.
86. Find longest word chain using Trie and DFS.
87. Find minimum number of invalid words (word break problem with Trie).
88. Find number of words with matching prefix and suffix.
89. Longest word in Trie with even ASCII sum.
90. Trie based data deduplication.
91. Compressed Trie with edge labels and search.
92. Implement pattern matching engine using Trie.
93. Bitwise AND of array range using Trie.
94. Optimize DNS lookups using Trie.
95. File system simulation using Trie.
96. DNS-based cache expiration using Trie.
97. String segmentation problem using Trie and DP.
98. Memory-efficient autocomplete system.
99. Longest matching suffix with multiple queries.
100. Trie + Aho-Corasick algorithm for multiple pattern search.

✅ Ready for:

- Python code examples
- Visualizations of Trie structures
- Or 🔥 your first **Daily Random Interview Challenge** from all topics?

Say the word, and I'll drop your challenge 💪📈

ALGORITHMS (each will have 100 questions):
Sorting Algorithms

Awesome! Here's **Topic 10: Sorting Algorithms** with **100 Interview Questions**, ordered from **Basic** → **Advanced**. This covers everything from classic sorting methods to hybrid/comparison-based sorts, in-place, external sorts, and application-based sorting problems.

🔹 Easy (1–30) – Basics of Sorting Algorithms

1. What is a sorting algorithm?
2. Difference between comparison-based and non-comparison sorts.
3. Implement Bubble Sort.
4. Implement Insertion Sort.
5. Implement Selection Sort.
6. Time and space complexity of Bubble Sort.
7. Time and space complexity of Insertion Sort.
8. Time and space complexity of Selection Sort.
9. Stable vs Unstable sorting – explain with examples.
10. What is in-place sorting?
11. Sort an array in ascending order using Bubble Sort.
12. Sort an array in descending order using Insertion Sort.
13. Sort a linked list using Insertion Sort.
14. Count number of swaps in Bubble Sort.
15. Sort an array of 0s, 1s, and 2s (Dutch National Flag problem).
16. Check if a given array is sorted.
17. Sort an array using Selection Sort and count comparisons.
18. Sort array and remove duplicates.
19. Sort characters in a string alphabetically.
20. Sort dates represented as strings.
21. Sort an array of strings based on length.
22. Sort an array based on frequency of elements.
23. Sort array of names by last name.
24. Sort students by grade, then name.
25. Sort points based on distance from origin.
26. Sort integers without using built-in functions.
27. Sort an array and return indices of original elements.

28. Implement Cocktail Shaker Sort.
29. Sort only the first K elements of an array.
30. Sort every 3rd element of the array separately.

🔹 Medium (31–70) – Divide & Conquer, Optimizations

31. Implement Merge Sort.
32. Implement Quick Sort (Lomuto partition).
33. Implement Quick Sort (Hoare partition).
34. Time and space complexity of Merge Sort.
35. Time and space complexity of Quick Sort.
36. When to prefer Merge Sort over Quick Sort?
37. Implement Merge Sort for linked list.
38. Implement Quick Sort for linked list.
39. Sort an array of objects using custom comparator.
40. Count inversions in an array using Merge Sort.
41. Find Kth smallest/largest element using QuickSelect.
42. Sort nearly sorted (K-sorted) array.
43. Sort a large file that doesn't fit into memory.
44. Implement hybrid sort (e.g., Timsort).
45. What is tail recursion optimization in Quick Sort?
46. Randomized Quick Sort – when and why?
47. Three-way partitioning Quick Sort.
48. External Merge Sort – when and how?
49. Sort an array using only swaps.
50. Sort by parity (even before odd).
51. Sort elements by absolute difference from a number.
52. Sort based on custom lexicographical rules.
53. Sort by frequency and value.
54. Sort array by modulo of elements.
55. Sort array of pairs based on second value.
56. Sort logs (digit vs letter logs).
57. Sort array of fractions.
58. Sort array of tuples by multiple keys.
59. Sort sparse matrix row-wise.

60. Sort characters by their occurrence count.
61. Sort employee records by date of joining.
62. Sort matrix row-wise and column-wise.
63. Sort a rotated sorted array.
64. Sort version numbers (e.g. “1.0.3” < “1.0.12”).
65. Sort a stack using only stack operations.
66. Sort without using conditional statements (ternary or tricks).
67. Find duplicate elements after sorting.
68. Check if two arrays are equal after sorting.
69. Sort array except one fixed index.
70. Find if an array can be sorted with one swap.

🔹 Hard (71–100) – Advanced & Application-Based

71. Implement Counting Sort.
72. Implement Radix Sort.
73. Implement Bucket Sort.
74. When to use Counting vs Radix vs Bucket sort?
75. Sort array with negative and positive integers using Counting Sort.
76. Sort floating point numbers using Bucket Sort.
77. Sort IP addresses.
78. Sort strings using Radix Sort.
79. Find number of swaps to sort an array (minimum swap problem).
80. Sort array using cyclic sort technique.
81. Sort array with one misplaced element.
82. Sort array into wave form.
83. Sort linked list with 0s, 1s, and 2s.
84. Sort large numbers represented as strings.
85. Sort file paths alphabetically.
86. Sort dictionary by values.
87. Sort based on the frequency of characters in a string.
88. Sort array and maintain relative order.
89. Sort an array with constraints (e.g., one element locked).
90. Sort array using XOR-based comparison.
91. Sorting with memory/mutation constraints.

92. Sort based on last digit of numbers.
93. Sort based on prime number priority.
94. Sort array by number of set bits.
95. Sort array by digit sum.
96. Sorting puzzle: maximum number of sorted subarrays.
97. Sorting + Binary Search hybrid (efficient queries).
98. Minimum steps to sort via adjacent swaps only.
99. Merge overlapping intervals then sort.
100. Minimum number of operations to sort array using specific moves.

✅ Let me know if you'd like:

- Code examples for any of these
- Visual sorting algorithm demos
- Your **daily random challenge** now (I can pull one right away!)

Want to move to **Topic 11: Searching Algorithms** next?

> Topic 11: Searching Algorithms

Let's dive into **Topic 11: Searching Algorithms** with **100 Interview Questions**, progressing from **Basics → Advanced**. This includes linear search, binary search (and its variants), exponential search, ternary search, search on rotated/sorted arrays, matrices, bitonic sequences, and more.

🔹 Easy (1–30) – Basic Search Concepts & Simple Implementations

1. What is a search algorithm?
2. Difference between linear search and binary search.
3. Implement linear search.
4. Implement binary search (iterative).
5. Implement binary search (recursive).
6. Time and space complexity of linear search.
7. Time and space complexity of binary search.
8. Search for an element in a sorted array.
9. Search for an element in an unsorted array.
10. Search for the first occurrence of an element.
11. Search for the last occurrence of an element.

12. Count occurrences of an element in sorted array.
13. Find the index of an element if present.
14. Return True/False if element is in list.
15. Search for an element in a 2D matrix (row-wise sorted).
16. Search an element in 2D matrix (fully sorted matrix).
17. Find position to insert element in sorted array.
18. Find the floor of a number in a sorted array.
19. Find the ceiling of a number in a sorted array.
20. Find square root using binary search.
21. Find cube root using binary search.
22. Find element in rotated sorted array (no duplicates).
23. Find peak element in an array.
24. Find minimum element in rotated sorted array.
25. Find maximum in a bitonic array.
26. Exponential search – explain and implement.
27. Ternary search – explain and implement.
28. Search in a mountain array.
29. Find missing element in sorted array with missing number.
30. Binary search vs interpolation search.

🔹 Medium (31–70) – Binary Search Variants & Applications

31. Find first and last position of a target in a sorted array.
32. Binary search for smallest element greater than target.
33. Binary search for largest element smaller than target.
34. Search in a rotated sorted array with duplicates.
35. Find peak index in a mountain array.
36. Find the rotation count in a rotated array.
37. Find local minima in an unsorted array.
38. Find K closest elements to a target.
39. Find Kth smallest element using binary search.
40. Search in infinite sorted array.
41. Search for square root of a non-perfect square to given precision.
42. Find element in an almost sorted array (each element misplaced by 1 position).
43. Search in sorted array of unknown length.

44. Search in linked list (sequential vs jump search).
45. Find transition point (0s to 1s in binary sorted array).
46. Find frequency of a target in sorted array.
47. Count 1s in a sorted binary array.
48. Find element in circularly sorted array.
49. Binary search on answer space – what is it?
50. Minimize max distance to gas stations (binary search on real numbers).
51. Find median of two sorted arrays.
52. Find K-th element of two sorted arrays.
53. Find max value under given condition using binary search.
54. Find max number of pages to allocate to students.
55. Allocate minimum number of painters.
56. Minimum number of days to make m bouquets.
57. Minimum capacity to ship packages within D days.
58. Aggressive cows problem.
59. Find number of elements less than or equal to X.
60. Koko eating bananas (binary search).
61. Minimum eating speed to finish bananas.
62. Find smallest divisor given a threshold.
63. Search in matrix with sorted rows and columns.
64. Binary search on bitonic array.
65. Binary search in rotated matrix.
66. Find a duplicate number in an array (Floyd's + BS).
67. Find smallest missing positive number using search.
68. Find range of elements less than or equal to X in matrix.
69. Matrix median using binary search.
70. Binary search for boundary value under constraints.

🔹 Hard (71–100) – Advanced Search Techniques

71. Find K-th smallest number in multiplication table.
72. Find K-th missing positive number.
73. Search in 3D matrix.
74. Find target in matrix using staircase search.
75. Find index where arr[i] == i.

76. Find duplicate in sorted array with values 1 to n.
77. Find element that occurs more than n/2 times (majority element via search).
78. Find element with maximum frequency in array using binary search.
79. Count number of pairs with absolute difference less than K.
80. Minimize the maximum distance between pairs using search.
81. Minimize max product under constraints.
82. Find min days to make bouquets (binary search on time).
83. Search in a dictionary of unknown size.
84. Search in rotated sorted array with multiple rotations.
85. Find common elements in three sorted arrays.
86. Find K-th smallest sum of pairs in two arrays.
87. Find peak element in matrix using binary search.
88. Find minimum speed to complete all tasks (search on speed).
89. Maximize minimum distance between balls.
90. Find smallest subarray with sum >= K using binary + prefix sums.
91. Search for perfect square in array.
92. Binary search on circular arrays.
93. Binary search to minimize time needed to paint boards.
94. Find unique element in sorted array where all others appear twice.
95. Find minimum element in rotated sorted array with duplicates.
96. Find first bad version (API simulation).
97. Find smallest number such that sum of digits ≤ k.
98. Ternary search to find minimum of unimodal function.
99. Find K-th smallest prime fraction.
100. Binary search optimization with greedy/DP hybrid (e.g., min max weight problem).

✅ Let me know if you want:

- Code examples (Python implementations)
- Visual explanation of any binary search variant
- Ready for **Topic 12: Greedy Algorithms**
- OR want your **Daily Random Interview Challenge** now?

Topic 12: Greedy Algorithms

Let's crush **Topic 12: Greedy Algorithms** with **100 Interview Questions**, organized from **Easy** → **Hard**. Greedy algorithms are used when a problem can be broken down into stages where each stage has a "locally optimal" choice that leads to a global optimum.

🔹 Easy (1–30) – Greedy Basics & Classic Problems

1. What is a greedy algorithm?
2. Greedy vs Dynamic Programming – key differences.
3. When does the greedy approach fail?
4. Activity selection problem (maximum non-overlapping intervals).
5. Fractional Knapsack problem.
6. Non-Fractional (0/1) Knapsack – why greedy fails.
7. Minimize number of coins to make change (coin change problem).
8. Minimum number of platforms required for trains.
9. Job sequencing with deadlines (maximum profit).
10. Huffman encoding (build Huffman tree).
11. Gas station problem – can you complete the circuit?
12. Distribute candies to children with ratings.
13. Assign cookies to children (greedy capacity match).
14. Maximize sum of non-adjacent elements (greedy vs DP).
15. Minimize jumps to reach end (jump game).
16. Jump game II – minimum jumps to reach end.
17. Merge overlapping intervals.
18. Insert interval into sorted list and merge.
19. Partition labels (group characters greedily).
20. Greedy string rearrangement to avoid adjacent duplicates.
21. Reorganize string by frequency.
22. Buy and sell stocks – maximize profit (one transaction).
23. Buy and sell stocks – multiple transactions allowed.
24. Schedule tasks with cooldown.
25. Minimum number of arrows to burst balloons.
26. Longest palindrome by rearranging characters.
27. Remove K digits to make smallest number.
28. Minimum number of meeting rooms required.
29. Largest number formed from array of integers.
30. Construct string with character constraints.

Medium (31–70) – Interval, Scheduling, Graph-Based Greedy

31. Minimum cost to connect sticks (merge ropes).
32. Network delay time using greedy Dijkstra.
33. Reconstruct queue by height.
34. Remove duplicates to get smallest lexicographical string.
35. Minimum swaps to make array sorted.
36. Minimize max difference after increasing/decreasing array elements.
37. Minimize absolute difference between arrays.
38. Find smallest number from digits with constraints.
39. Minimize cost of cutting a rod.
40. Distribute items to people under fairness constraint.
41. Minimize time to finish all tasks by multiple workers.
42. Assign tasks to machines – minimize completion time.
43. Connect N ropes with minimum cost.
44. Maximize sum of array after K negations.
45. Lexicographically smallest subsequence of distinct characters.
46. Maximum sum of i*arr[i] after rotations.
47. Reduce array to K elements with min total cost.
48. Construct target array with multiple operations.
49. Longest possible palindrome after k deletions.
50. Minimize penalties for task scheduling.
51. Remove intervals to make non-overlapping list.
52. Greedy coloring of graph.
53. Greedy coin problem for non-standard denominations.
54. Minimize cash flow in a group of people (debts).
55. Form string without 3 consecutive same letters.
56. Maximize score after flipping pairs.
57. Split string into balanced substrings.
58. Rearrange characters to avoid repeating pairs.
59. Find the maximum number of chunks to sort an array.
60. Find maximum number of non-overlapping substrings.
61. Choose pairs to form min/max sum under constraints.
62. Minimize deletions to make all elements same.
63. Maximize area covered by intervals.

64. Choose k intervals to maximize sum of lengths.
65. Reduce string to smallest possible form.
66. Pick minimum intervals to cover the full range.
67. Find greedy solution for water trapping problem.
68. Maximize points from cards (sliding + greedy).
69. Construct minimum string from priority constraints.
70. Efficient greedy rounding problem (floor vs ceil sum target).

🔹 Hard (71–100) – Greedy + Heap/Greedy + Sorting/Greedy Optimization

71. Minimum cost to hire k workers with quality constraints.
72. Maximize performance of a team (speed + efficiency tradeoff).
73. Minimize deviation in array.
74. Maximize average of selected subsequence.
75. Minimize unfairness in candy distribution.
76. Minimize number of operations to equalize array.
77. Greedy strategy for game theory – coin row problem.
78. Find smallest range covering elements from k lists.
79. Create largest possible number with digit constraints.
80. Maximize capital after k projects.
81. Minimize the score after splitting array into k parts.
82. Minimum number of removals to make mountain array.
83. Minimum time to finish tasks with split capability.
84. Schedule maximum events in minimum rooms with time conflict.
85. Construct tree from leaf value cost (greedy merge).
86. Maximum frequency stack (greedy with priority queue).
87. Rearrange barcodes (distance constraint).
88. Optimize string compression with k deletions.
89. Balance parentheses with minimum insertions (greedy bracket matching).
90. Count good strings that meet lexicographical constraints.
91. Greedy solution for airplane seat assignment.
92. Minimize cost to move chips to same position.
93. Find smallest lexicographical path from board.
94. Make array strictly increasing (greedy with auxiliary array).

95. Minimize the max pair difference for team selection.
96. Maximize sum of min pairs.
97. Minimum flips to convert binary matrix to zero matrix.
98. Greedy solution to string transformation with costs.
99. Assign jobs to workers to maximize profit.
100. Build the lexicographically smallest subsequence from constraints (Monotonic Stack + Greedy).

✅ Want:

- Python implementations for any?
- Explanation of where greedy fails vs DP?
- Move to **Topic 13: Divide and Conquer Algorithms**
- Or fire up a **Daily Random Interview Challenge** from all topics?

You call it! 💡💪

> Topic 13: Divide and Conquer Algorithms

Let's hit **Topic 13: Divide and Conquer Algorithms** with **100 curated interview questions**, ranging from classic to clever 💥. Divide and Conquer splits a problem into subproblems, solves them recursively, and combines their results.

🔹 Easy (1–30) – Understanding Basics & Core Concepts

1. What is divide and conquer?
2. Divide and conquer vs dynamic programming.
3. Implement merge sort.
4. Implement quick sort.
5. Time and space complexity of merge sort.
6. Time and space complexity of quick sort.
7. Why quick sort is not stable?
8. Merge two sorted arrays using divide and conquer.
9. Count inversions in an array.
10. Binary search as a divide and conquer approach.
11. Maximum subarray sum (Kadane's vs divide and conquer).
12. Find min and max in array using divide and conquer.
13. Search in rotated sorted array using divide and conquer.

14. Find majority element using divide and conquer.
15. Find kth smallest element using divide and conquer.
16. Find median of two sorted arrays.
17. Merge K sorted arrays using divide and conquer.
18. Merge K sorted linked lists using divide and conquer.
19. Karatsuba algorithm for fast multiplication.
20. Check if an array is sorted using divide and conquer.
21. Sort colors (Dutch national flag) using divide and conquer.
22. Find power of a number efficiently (exponentiation by squaring).
23. Reverse a linked list using divide and conquer.
24. Merge two BSTs into a balanced BST.
25. Convert sorted array to balanced BST.
26. Convert sorted linked list to balanced BST.
27. Closest pair of points (O(n log n)).
28. Find duplicate in array using divide and conquer.
29. Find smallest positive number missing from array.
30. Implement integer division using divide and conquer.

🔹 Medium (31–70) – Applications in Arrays, Strings, Trees

31. Count number of subarrays with sum greater than K.
32. Count number of reverse pairs in an array.
33. Find smallest/largest element in unsorted array.
34. Find peak element using binary search.
35. Find power(x, n) using recursion (even/odd split).
36. Search in a 2D matrix using divide and conquer.
37. Multiply two large numbers using divide and conquer.
38. Longest substring that is a palindrome using divide and conquer.
39. Maximum/minimum difference pair in array.
40. Matrix multiplication using divide and conquer.
41. Inorder traversal using divide and conquer.
42. Preorder traversal using divide and conquer.
43. Postorder traversal using divide and conquer.
44. Build segment tree using divide and conquer.
45. Range minimum query using segment tree.

46. Merge intervals using divide and conquer.
47. Longest common prefix using divide and conquer.
48. Maximum product subarray using divide and conquer.
49. Search in bitonic array using divide and conquer.
50. Find intersection of two arrays using divide and conquer.
51. Longest common subsequence using divide and conquer.
52. Tiling problem (2^n x 2^n grid with one tile missing).
53. Sudoku solver using divide and conquer.
54. Divide array into K subsets of equal sum.
55. Find number of range sums in array.
56. Number of pairs with sum in given range.
57. Count subarrays with equal number of 0s and 1s.
58. Find all elements occurring more than n/3 times.
59. Rotate matrix using divide and conquer.
60. Reconstruct binary tree from inorder and preorder traversal.
61. Reconstruct binary tree from inorder and postorder traversal.
62. Find all triplets with sum less than target.
63. Find inversion count using merge sort.
64. Divide and conquer to evaluate mathematical expression.
65. Balance a binary search tree using D&C.
66. Range sum queries with segment trees.
67. Implement Strassen's matrix multiplication.
68. Find GCD using divide and conquer (Euclidean method).
69. Divide string into balanced parts.
70. Find longest increasing subsequence using divide and conquer.

🔹 Hard (71–100) – Advanced Patterns & Optimization

71. Skyline problem.
72. Count smaller elements after self.
73. K-th smallest pair distance.
74. Find the smallest good base.
75. Find min time to paint N boards using painters.
76. Count range sum using prefix + D&C.
77. Reconstruct itinerary (divide path traversal).

78. Minimize number of multiplications in matrix chain multiplication.
79. Find minimum operations to convert string A to B.
80. Max rectangle area in histogram using divide and conquer.
81. Divide points into k clusters with minimum intra-cluster distance.
82. Efficient polynomial multiplication using FFT (Fast Fourier Transform).
83. Solve N-Queens using D&C.
84. Parallel merge sort using D&C.
85. D&C for balancing AVL trees.
86. Find the longest palindromic substring using center expansion + D&C.
87. Search word in matrix (backtracking + D&C).
88. Convex hull using divide and conquer.
89. Determine if array can be divided into equal sum parts.
90. Minimum cost to divide gold bar into pieces.
91. Efficient implementation of random quicksort.
92. Find peak in 2D matrix.
93. Closest pair of points in 2D/3D plane.
94. Big Integer multiplication using divide and conquer.
95. Weighted job scheduling using greedy + divide.
96. Divide and conquer for range XOR queries.
97. Median of medians algorithm (selection in linear time).
98. Solve Sudoku with bitmasks + D&C.
99. D&C on tries to split words into optimal groups.
100. Build optimal binary search tree using divide and conquer.

✅ Want:

- Code examples for merge sort, quick sort, skyline, or anything specific?
- Move to **Topic 14: Dynamic Programming**?
- Or should I drop your **Daily Random Coding Challenge** from all topics?

Topic 14: Dynamic Programming?

Let's tackle one of the most powerful topics in algorithm interviews: **Topic 14: Dynamic Programming (DP)** 💡

Here are **100 dynamic programming interview questions**, organized from **Easy** → **Medium** → **Hard**, and grouped by patterns like 1D DP, 2D DP, DP on Trees, DP with Bitmask, DP with

Memoization/Tabulation, and more.

🔹 Easy (1–30) – DP Fundamentals & Classic 1D Problems

1. What is dynamic programming?
2. Top-down vs Bottom-up DP.
3. Recursion vs Memoization vs Tabulation.
4. Fibonacci sequence using DP (memoization + tabulation).
5. Climbing stairs (ways to reach the nth step).
6. Minimum cost climbing stairs.
7. House robber (no two adjacent houses).
8. House robber II (circular street).
9. Maximum subarray sum (Kadane's algorithm).
10. Number of ways to reach the top (1, 2, 3 steps).
11. Decode ways (A=1, B=2...) – count valid decodings.
12. Count number of ways to make change (coin change I).
13. Minimum number of coins to make amount (coin change II).
14. Longest common subsequence.
15. Longest common substring.
16. Longest increasing subsequence.
17. 0/1 Knapsack.
18. Partition equal subset sum.
19. Subset sum problem.
20. Word break problem.
21. Can sum / how sum / best sum (classic recursion to DP).
22. Edit distance (Levenshtein distance).
23. Palindromic subsequence count.
24. Is string a palindrome (memoized check).
25. Rod cutting problem.
26. Jump game (can reach end).
27. Jump game II (minimum jumps to reach end).
28. Number of ways to reach point (m, n) in grid.
29. Unique paths in a grid (with obstacles).
30. Tiling problem (ways to tile 2xN board).

Medium (31–70) – 2D DP, Tabulation, Optimization

31. Matrix chain multiplication.
32. Boolean parenthesization problem.
33. Egg dropping problem.
34. Scrambled string check.
35. Wildcard matching (with `*` and `?`).
36. Regular expression matching.
37. Maximum sum rectangle in 2D matrix.
38. Paint house problem (min cost with no adjacent same color).
39. Burst balloons for max coins.
40. Minimum path sum in grid.
41. Cherry pickup problem.
42. Number of longest increasing subsequences.
43. Minimum insertions to make string palindrome.
44. Minimum deletions to make string palindrome.
45. Count palindromic substrings in a string.
46. Minimum number of operations to convert A to B.
47. Longest repeating subsequence.
48. Longest palindromic subsequence.
49. Optimal BST construction.
50. Number of ways to make target from given expressions.
51. Count number of palindromic subsequences.
52. Longest bitonic subsequence.
53. Maximum profit in job scheduling (like weighted activity selection).
54. Largest square of 1s in a binary matrix.
55. Word wrap problem (minimize extra spaces cost).
56. Interleaving strings (check if one string is a shuffle of two).
57. Minimum cuts for palindrome partitioning.
58. Minimum falling path sum.
59. Find number of arithmetic slices.
60. Count different subsequences.
61. Largest divisible subset.
62. Number of distinct subsequences of a string.
63. Decode ways II (includes `*` wildcard).

64. Maximum length of pair chain.
65. Maximum sum increasing subsequence.
66. Paint fence (no more than 2 adjacent same colors).
67. Counting tilings of grid with dominos.
68. Domino and tromino tiling.
69. Count ways to partition array into k subsets.
70. Count of dice roll combinations to reach a sum.

🔹 Hard (71–100) – Advanced DP Patterns: Trees, Bitmasking, States

71. DP on trees – diameter of tree.
72. DP on trees – maximum sum of non-adjacent nodes.
73. DP on trees – longest path with same value.
74. DP with bitmask – traveling salesman problem (TSP).
75. DP with bitmask – count arrangements (like N-queens variant).
76. DP with memoization – maximize score with picking numbers.
77. DP with sliding window optimization.
78. Maximum profit from at most k transactions (stock problem).
79. Best time to buy and sell stock with cooldown.
80. Best time to buy and sell with transaction fee.
81. Count binary strings with no consecutive 1s.
82. Minimum subset difference partition.
83. Count partitions with given difference.
84. Max product subarray (sign trick + dp).
85. Count palindromic subsequences of length k.
86. N-digit numbers formed using digits 1-9 with constraints.
87. Digit DP – count numbers with certain properties.
88. DP with Trie – word break with dictionary.
89. Number of ways to tile a board with domino and L-shaped tiles.
90. Minimum cost to split string into palindromic substrings.
91. Number of ways to color grid under constraints.
92. Stone game (Alice vs Bob) – who wins optimally.
93. Predict winner in array game.
94. Palindrome removal game.

95. Stickers to spell word (state compression).
96. String compression with deletions allowed.
97. Minimum cost to build binary tree from leaf values.
98. Shortest common supersequence.
99. Arithmetic slices II (count subsequences with same difference).
100. 2-player game with optimal moves (Minimax + DP).

🔥 Want:

- Python code for any of these?
- Explanations grouped by pattern (e.g., "DP on Trees" or "Bitmask")?
- Move to **Topic 15: Backtracking Algorithms**?
- Or get a **Daily Random Challenge** from all 14 topics so far?

Your move, legend 💪

> Topic 15: Backtracking Algorithms?

Ohhh yeah, let's hit **Topic 15: Backtracking Algorithms** – a *must-know* for solving problems with **combinatorics, constraints, permutations**, and **decision trees**! 🔁⏪
Here are **100 Backtracking Interview Questions**, escalating from classic to cunning 👇

🔹 Easy (1–30) – Classic Backtracking Patterns

1. What is backtracking?
2. How is backtracking different from recursion?
3. N-Queens problem (place N queens on a board).
4. Rat in a Maze (find all paths from start to finish).
5. Knight's Tour problem.
6. Generate all permutations of a string.
7. Generate all subsets (the power set).
8. Subsets with duplicates.
9. Generate balanced parentheses.
10. Letter combinations of a phone number.
11. Word search in a 2D board.
12. Palindrome partitioning of a string.
13. Sudoku Solver.

14. Solve a magic square of size N.
15. Permutations of an array with duplicate elements.
16. Generate all binary strings of length N.
17. Restore IP addresses from string.
18. Count of subsets with a given sum.
19. Word break II (list all segmentations).
20. Binary watch combinations.
21. Combination sum (repetition allowed).
22. Combination sum II (unique combos).
23. Gray code generator.
24. Generate well-formed brackets of 3 types.
25. Coin change combinations.
26. All valid 8-puzzle moves from initial state.
27. Minimum number of squares to sum to N.
28. Find all paths from top-left to bottom-right in grid.
29. Jump to reach last index with all possible paths.
30. Solve Tower of Hanoi with k disks.

🔹 Medium (31–70) – Decision Trees & Combinatorics

31. N-Knights problem (place knights so they don't attack).
32. Generate all valid expressions with operators to reach target.
33. Generate unique BSTs.
34. Partition array into k equal sum subsets.
35. Factor combinations of a number.
36. Add operators to reach target expression.
37. Combination generator with constraints.
38. Palindrome string formation from array.
39. Hamiltonian path detection.
40. Coloring of graph with at most k colors.
41. M Coloring Problem.
42. Tiling a board with dominoes.
43. Crossword puzzle fill with word list.
44. Generate valid time combinations.
45. Matchsticks to square (can form square?).

46. Reconstruct itinerary with backtracking.
47. Split array into k subsets with equal average.
48. Campus bike allocation.
49. Count total N-Queens solutions.
50. Word ladder II – all shortest transformation sequences.
51. Tug of war problem (divide array into 2 subsets with minimum diff).
52. 24 Game (check if 4 nums can make 24 using ops).
53. Maximum score path with k turns.
54. Convert binary string to palindrome with minimum flips.
55. Sum of digits equals target with length N.
56. Knight's tour count (how many possible tours).
57. Build all palindromes from char frequency.
58. Balanced parenthesis with min removals.
59. K permutations out of n numbers.
60. Multiply two numbers using backtracking (with constraints).
61. Replace wildcards with valid binary strings.
62. Next lexicographical permutation via backtracking.
63. Total distinct Sudoku boards.
64. Interleave two strings to form a third.
65. Number maze solver with teleporters.
66. Non-overlapping arrangement of intervals.
67. Reconstruct 2D matrix with sum constraints.
68. Unlock pattern generator (Android lock screen).
69. Arrange people into teams with skill constraints.
70. Backtrack-based calculator evaluator.

🔹 Hard (71–100) – Multi-State Search, Optimization, Advanced

71. Knight's minimum moves to reach a square (min path via backtracking).
72. Crossword fill with overlapping words.
73. Arrange words to form a valid crossword.
74. N-Queens II (with obstacles).
75. Word break with synonyms and dictionary groups.
76. Generate all possible mathematical expressions with precedence.
77. Place non-overlapping rectangles on a grid.

78. K-Sum problem (return all k-sized subsets summing to target).
79. Solve alphametic puzzle (e.g., SEND + MORE = MONEY).
80. K combinations of balanced parentheses.
81. Assign tasks to k workers with min time.
82. Tetris-like grid fill with shapes.
83. Pentomino tiling on 6x10 board.
84. Maze with portals and backdoors.
85. Robot path with limited energy & constraints.
86. Knight's probability to remain on board after k moves.
87. Generate all seating arrangements with distance rules.
88. Constraint satisfaction for scheduling exams.
89. Fill Latin Square of size N.
90. Minesweeper puzzle solver.
91. Largest square of 1s using backtracking.
92. Bitmask backtracking with constraints (like Travelling Salesman).
93. Cryptarithm puzzles.
94. Optimize seating in theater (gap, aisle constraints).
95. Rearrange characters with adjacent difference constraint.
96. Generate all possible expression trees.
97. Sliding puzzle with min swaps.
98. Sudoku variant: X Sudoku (diagonals must be 1–9).
99. Global constraint satisfaction (CSP solver).
100. Solve KenKen puzzle with constraints.

🧠 Want:

- Code or dry run walkthrough for a few problems?
- Daily random mix from *any of the 15 topics so far*?
- Should we go to **Topic 16: Recursion** next?

You're steamrolling these 🔥

> Topic 16: Recursion

Let's gooo! 🚀
Welcome to **Topic 16: Recursion** — the beating heart behind so many algorithms, especially in **divide and conquer**, **backtracking**, **DP**, and even **tree/graph traversals**.

Here are **100 Recursion Interview Questions**, covering classic problems, tree recursion, optimizations, and tricky edge cases.

🔹 Easy (1–30) – Foundational Recursion

1. What is recursion?
2. Base case vs recursive case?
3. Print numbers from 1 to N using recursion.
4. Print N to 1 using recursion.
5. Sum of first N natural numbers (recursive).
6. Factorial of a number (recursive).
7. Fibonacci series (recursive).
8. Reverse a string using recursion.
9. Check if a string is a palindrome using recursion.
10. Count digits of a number using recursion.
11. Find GCD (Euclidean algorithm with recursion).
12. Power of a number (recursive exponentiation).
13. Print array elements recursively.
14. Maximum element in array using recursion.
15. Binary search using recursion.
16. Linear search using recursion.
17. Replace all occurrences of a char in string recursively.
18. Tower of Hanoi (classic recursion).
19. Count the number of ways to climb stairs.
20. Sum of digits using recursion.
21. Product of array elements (recursive).
22. Remove all duplicates from string using recursion.
23. Find first/last index of an element recursively.
24. Print all subsets of a set.
25. Generate all permutations of a string.
26. Check if array is sorted recursively.
27. Insert an element in a sorted array recursively.
28. Count number of zeroes in a number recursively.
29. Check if two strings are anagrams recursively.
30. Find length of a linked list recursively.

Medium (31–70) – Patterns & Divide and Conquer

31. Merge sort using recursion.
32. Quick sort (recursive version).
33. Count subsets with a given sum.
34. Generate all binary strings of length N.
35. Subset sum problem.
36. Generate combinations (nCr) using recursion.
37. Josephus problem.
38. Solve maze using recursion.
39. Find all paths from source to destination in grid.
40. Print all valid parentheses of n pairs.
41. Decode ways (recursive version).
42. Longest common subsequence (recursive).
43. Longest palindromic subsequence (recursive).
44. Count total palindromic substrings recursively.
45. Multiply two numbers recursively (no `*`).
46. Find Nth node from end of linked list recursively.
47. Add two numbers recursively (no `+`).
48. Sum of elements between two nodes in BST.
49. Count binary trees with n nodes (Catalan recursion).
50. Count ways to reach Nth stair (1, 2, or 3 steps).
51. Nth Tribonacci number recursively.
52. Find power using exponentiation by squaring.
53. Replace Pi in string recursively ("3.14").
54. Convert string to integer recursively.
55. Convert integer to string recursively.
56. Add binary numbers using recursion.
57. Interleave two strings recursively.
58. Interleave multiple strings recursively.
59. Check for balanced brackets using recursion.
60. Mirror a binary tree recursively.
61. Check if a tree is symmetric recursively.
62. Count leaf nodes in binary tree recursively.
63. Sum of all nodes in binary tree.

64. Convert binary tree to DLL (recursive).
65. Serialize & deserialize binary tree recursively.
66. Lowest common ancestor using recursion.
67. Diameter of a binary tree using recursion.
68. Validate BST using recursion.
69. Max path sum in a binary tree.
70. Postorder traversal recursively.

🔹 Hard (71–100) – Complex & Multi-level Recursion

71. All permutations of numbers (with duplicates).
72. N-Queens solver (pure recursion).
73. Sudoku solver using recursion.
74. Word break II (list all valid breaks).
75. Generate all expression combinations to reach target.
76. 24 Game using recursion and floating-point ops.
77. Hamiltonian path check recursively.
78. Crossword puzzle fill recursively.
79. Remove invalid parentheses (all possibilities).
80. Tiling problem (ways to tile a 2xN board).
81. Count number of BSTs (Catalan with recursion).
82. All unique BSTs (generate structure recursively).
83. Wildcard pattern matching with `*`, `?`.
84. Regular expression matching with `.` and `*`.
85. Solve a cryptarithmetic puzzle recursively.
86. Knight's tour (can it complete the board?).
87. Build expression tree recursively from postfix.
88. Find median in stream using two heaps + recursive balance.
89. Generate all rooted trees of N nodes.
90. Number of ways to reach score N with 3, 5, 10.
91. Count dice roll combinations (sum = N, D dice, F faces).
92. Boolean parenthesization (count ways to evaluate to True).
93. Count valid number sequences with adjacent diff constraint.
94. Matchsticks to square recursively.
95. Partition array into 2 subsets with equal sum.

96. Minimum insertions to make palindrome (recursive).
97. Palindrome partitioning II (min cuts).
98. Combination sum III (k numbers add to n).
99. Build all valid android lock patterns.
100. Solve mini "interpreter" that handles nested expressions recursively.

🧠 Ready for:

- Any of these problems in Python?
- Daily mix of challenges from all **16 topics** so far?
- Or should we hit **Topic 17: Bit Manipulation** next?

You're crushing this prep! ⚡

> Topic 17: Bit Manipulation

Ohhh snap! You're entering **bit wizardry mode** now ⚡🧙
Welcome to **Topic 17: Bit Manipulation** — the land of shifts, masks, XOR tricks, and mind-blowing optimizations.

Here are **100 Bit Manipulation Interview Questions**, from beginner to brain-melter:

🔹 Easy (1–30) – Bit Basics & Fundamentals

1. What is bit manipulation?
2. Convert decimal to binary and vice versa.
3. Check if a number is even or odd using bitwise.
4. Multiply number by 2 using bit shift.
5. Divide number by 2 using bit shift.
6. Check if ith bit is set.
7. Set ith bit.
8. Clear ith bit.
9. Toggle ith bit.
10. Count number of set bits (Brian Kernighan's Algorithm).
11. Turn off rightmost set bit.
12. Isolate the rightmost set bit.
13. Isolate the rightmost unset bit.
14. Swap two numbers using XOR.

15. Check if a number is power of 2.
16. Find the only non-repeating element in array (XOR).
17. XOR of all elements in array.
18. Add two numbers without using `+` or `-`.
19. Check if two numbers have opposite signs.
20. Find sign of an integer without branching.
21. Calculate absolute value using bitwise.
22. Convert uppercase to lowercase using bits.
23. Check if number is a power of 4.
24. Find minimum of two numbers using bits.
25. Find maximum of two numbers using bits.
26. Toggle case of character using bit.
27. Determine if a number has exactly one bit set.
28. Count number of 1-bits in binary representation.
29. Find binary representation of negative numbers (two's complement).
30. Remove least significant bit that is set.

🔹 Medium (31–70) – XOR Tricks, Bitmasking, Patterns

31. Find two non-repeating elements in array.
32. Find missing number in array (1 to n).
33. Find the single number in array where every other appears thrice.
34. Find element appearing once where others appear twice and once appears thrice.
35. Detect if integer is palindrome in binary.
36. Count set bits from 1 to N.
37. Generate power set using bitmasks.
38. Subset sum problem using bitmasks.
39. Detect if N and N-1 have common set bits.
40. Flip all bits of a number.
41. Replace all 0s with 1s in binary representation.
42. Determine position of rightmost set bit.
43. Determine position of leftmost set bit.
44. Number of bits to be flipped to convert A to B.
45. Divide two integers without using `/`, `*`, `%`.
46. Multiply two numbers without using `*`.

47. Bitwise AND of numbers in a range.
48. Implement logical NOT using only bit operators.
49. Find XOR of all numbers from 1 to N.
50. Implement XOR gate logic in code.
51. Count number of bits to store a number.
52. Minimum XOR value of pairs in array.
53. Maximum XOR of any two numbers in array.
54. Find XOR of array elements except self.
55. Determine if N is sparse (no two adjacent 1s).
56. Copy bits from one number to another in a given range.
57. Rotate bits left and right.
58. Decimal equivalent of binary linked list.
59. Merge bits from two numbers.
60. Find next higher number with same number of 1s.
61. Find previous lower number with same number of 1s.
62. Count total set bits in all numbers from 1 to N.
63. Calculate XOR from L to R.
64. Is power of 2 using only one-liner bit trick?
65. Set all bits from L to R in a number.
66. Clear all bits from L to R.
67. Generate all masks of size N.
68. Convert Gray code to binary.
69. Convert binary to Gray code.
70. Number of bits required to convert integer A to B.

🔹 Hard (71–100) – Bitmask DP, Optimization, Advanced Logic

71. N-Queens using bitmasks.
72. TSP (Traveling Salesman Problem) with bitmask DP.
73. Count ways to assign jobs (bitmask DP).
74. Minimum number of flips to make binary string alternate.
75. Minimum XOR value of all subset XORs.
76. Find smallest subset with given XOR value.
77. Maximum XOR subset of an array.
78. Count subsets with given XOR value.

79. Find maximum AND value of pairs.
80. Maximum AND sum in a matrix.
81. Find max XOR path in a tree.
82. Minimum number of operations to convert number A to B using only AND/XOR.
83. Count total unique bitwise ORs of all subarrays.
84. Count numbers with exactly K set bits.
85. DP on bits: Count numbers less than N with given digit constraints.
86. Bitwise Trie: Insert & search for max XOR pair.
87. Solve cryptarithm puzzle using bit masks.
88. Implement BitSet (like Java) in Python.
89. Count valid android lock patterns (bitmasking cells).
90. Sum of XORs of all subsets of an array.
91. Minimize OR after removing one element.
92. Generate minimal Gray code sequence for n-bits.
93. Longest subarray with given XOR.
94. Longest subarray with bitwise AND > 0.
95. Max sum of XOR-split in array.
96. Find K-th bit in binary string (Kth expansion problem).
97. Bitmask-based sudoku solver.
98. Find Kth smallest XOR of subarrays.
99. Count of numbers <= N having exactly K bits set.
100. Implement a simple CPU instruction set (AND, OR, NOT, XOR) using functions.

🧠 Want:

- Code examples or breakdowns for any of these?
- Daily random challenge from **all 17 topics** so far?
- Or jump to **Topic 18: Two Pointers / Sliding Window**?

Bit by bit, you're building a beast-level brain 🧠💥

Sliding Window & Two Pointers Mathematical Algorithms

Yesss, you're on fire! 🔥
Let's double-drop **Topic 18** and **Topic 19**:

✅ Topic 18: Sliding Window & Two Pointers

These are *essential* for optimizing time complexity, especially in arrays and strings. Mastering these can instantly turn `O(n²)` brute forces into `O(n)` beauties.

🔹 Two Pointers (1–50)

1. Find a pair in sorted array that adds up to K.
2. Remove duplicates from sorted array in-place.
3. Move all zeroes to end without extra space.
4. Check if a string is a palindrome.
5. Container With Most Water.
6. Trapping Rain Water.
7. Merge two sorted arrays (no extra space).
8. Find intersection of two arrays.
9. Reverse words in a string in-place.
10. Reverse characters in a string (no built-in functions).
11. Longest substring without repeating characters.
12. Merge intervals (sorting + two pointers).
13. Partition array into 3 parts with equal sum.
14. Dutch National Flag (sort 0s, 1s, 2s).
15. Segregate even and odd numbers.
16. Find if there exists a triplet with 0 sum.
17. 3Sum (classic two pointers).
18. 4Sum using two pointers.
19. Remove element in-place.
20. Shortest subarray with sum at least K.
21. Sort array of 0s and 1s.
22. Valid palindrome II (remove at most one char).
23. Minimize max difference in heights.
24. Find closest pair from two arrays.
25. Count pairs with absolute diff <= K.
26. Maximum number of vowels in substring of length k.
27. Shortest distance between words in array.
28. Count number of pairs with product < K.
29. Longest mountain in array.
30. Reverse subarray between two indices.
31. Replace elements with greatest element on right.
32. Find maximum average subarray of length k.

33. Number of substrings with same number of 0s and 1s.
34. Maximum product of two integers in array.
35. Count number of valid triangles.
36. Check if a subarray is sorted.
37. Sort a nearly sorted (k-sorted) array.
38. Minimum swaps to bring K together.
39. Minimum window with all distinct elements.
40. Rearrange array in max-min form.
41. Merge sorted intervals (again).
42. Count pairs with given sum using two pointers.
43. Find sum of min and max in each window.
44. Longest nice subarray.
45. Longest substring with at most 2 distinct characters.
46. Find missing ranges in sorted array.
47. Check if string is rotation of another using 2 pointers.
48. Longest even length substring with equal sum halves.
49. Check for palindromic subarrays of fixed size.
50. Maximum number of 1s in window of size k.

🔹 Sliding Window (51–100)

51. Maximum sum of subarray of size K.
52. Minimum sum subarray of size K.
53. First negative number in every window of size K.
54. Maximum of all subarrays of size K.
55. Minimum window substring (classic).
56. Longest substring without repeating characters.
57. Longest subarray with at most K distinct.
58. Longest substring with same letters after K replacements.
59. Count anagrams in a string.
60. Find all anagrams of a pattern in a string.
61. Check for permutation in string.
62. Max consecutive 1s after flipping at most K zeroes.
63. Max length of substring with at most K replacements.
64. Minimum swaps to bring K elements together.
65. Smallest window containing all characters.

66. Longest subarray with sum = K.
67. Max number of unique elements in subarray of size K.
68. Max difference between max and min in subarray.
69. Maximum frequency after K increments.
70. Number of substrings divisible by 3.
71. Longest repeating character replacement.
72. Number of substrings of length K with K distinct.
73. Longest subarray with absolute diff <= limit.
74. Min length subarray with sum ≥ target.
75. Count binary substrings with equal 0s and 1s.
76. Count subarrays with exactly K different integers.
77. Max profit from stock prices with at most two transactions.
78. Count of subarrays where elements are in increasing order.
79. Max product subarray (Kadane + window).
80. Count subarrays with sum less than K.
81. Count of windows with exactly K primes.
82. Number of subarrays with sum divisible by K.
83. Longest repeating substring with at most one mismatch.
84. Sum of all subarray minimums.
85. Sum of all subarray maximums.
86. Count number of substrings containing all vowels.
87. Smallest subarray with all 1s.
88. Maximize score after splitting string.
89. Number of subarrays with bounded maximum.
90. Longest subarray with sum divisible by K.
91. Longest subarray with difference of 1.
92. Number of nice subarrays (odd count = k).
93. Length of longest subarray with at most 1 zero.
94. Longest arithmetic subarray.
95. Count distinct elements in every window of size k.
96. Number of subarrays with exactly k odd numbers.
97. Minimum operations to reduce x to zero.
98. Max number of vowels in a substring.
99. Count subarrays where max – min ≤ k.
100. Sliding window median.

✅ Topic 19: Mathematical Algorithms (100 Qs)

Focus on GCD, LCM, primes, modular math, number theory tricks.

Do you want me to drop all 100 Math Algorithm questions now as Topic 19?
Or should I give you code solutions/daily challenges/mix from 18 first?

Topic 19: Mathematical Algorithms (100 Qs)

Let's gooo! 🧮📐 Welcome to **Topic 19: Mathematical Algorithms** — the power tools of any problem solver 💪
Whether it's **primes, GCDs, modulos, or fancy number theory**, this is your mathematical battlefield.

✅ 100 Mathematical Algorithm Questions

🔹 Basics & Number Theory (1–30)

1. Compute GCD (Euclidean Algorithm).
2. Compute LCM of two numbers.
3. Find all factors of a number.
4. Count the number of divisors of a number.
5. Check if a number is prime.
6. Generate all prime numbers ≤ N (Sieve of Eratosthenes).
7. Optimized prime checking up to sqrt(N).
8. Prime factorization of a number.
9. Count distinct prime factors.
10. Find largest prime factor of a number.
11. Compute a^b (power) efficiently (binary exponentiation).
12. Compute (a^b) % m.
13. Modular inverse of a number.
14. Solve ax + by = gcd(a, b) (Extended Euclidean Algorithm).
15. Check if a number is a perfect square.
16. Count perfect squares in a range.
17. Sum of first N natural numbers.
18. Sum of first N odd/even numbers.
19. Count trailing zeroes in factorial.

20. Compute N! mod m.
21. Compute C(n, r) % m.
22. Precompute factorials modulo m.
23. Compute nCr using Pascal's Triangle.
24. Check if two numbers are co-prime.
25. Euler's Totient Function.
26. Generate Euler's totient from 1 to N.
27. Fermat's Little Theorem.
28. Wilson's Theorem for Primes.
29. Sum of divisors of a number.
30. Product of divisors of a number.

🔹 Modular Arithmetic & Tricks (31–60)

31. Modular addition, subtraction, multiplication.
32. Modular exponentiation.
33. Modular division (inverse-based).
34. Find remainder when large number (in string) is divided by M.
35. Check if number is divisible by 3, 4, 5... (digit tricks).
36. Count numbers divisible by a given number in range [L, R].
37. GCD of array.
38. LCM of array.
39. Find N-th Fibonacci number.
40. Find N-th Fibonacci modulo M.
41. Fast doubling method for Fibonacci.
42. Check if a number is a Fibonacci number.
43. Count of coprime pairs in array.
44. Calculate power of prime in N!.
45. Calculate number of trailing zeroes in N! in base B.
46. Number of digits in N!.
47. Find X such that (X^K) mod P = N.
48. Multiply large numbers represented as strings.
49. Find X^Y where X and Y are large strings.
50. Perform modular inverse with non-prime mod using Extended Euclidean.
51. Square root modulo prime.
52. Check if number can be written as sum of two squares.

53. Count solutions to a^2 + b^2 = N.
54. Calculate (A*B*C*D...) mod M efficiently.
55. Find smallest number with exactly N divisors.
56. Check if N is a power of 10.
57. Find sum of all numbers ≤ N that are divisible by A or B.
58. Solve N % (i+1) for i from 1 to N efficiently.
59. Count numbers with exactly 3 divisors.
60. Number of unordered coprime pairs from 1 to N.

🔹 Combinatorics & Counting (61–85)

61. Count number of combinations nCr ≤ K.
62. Count permutations of array with constraints.
63. Number of distinct arrangements with repetitions.
64. Count number of ways to distribute N identical items into K boxes.
65. Integer partitions of N.
66. Catalan Numbers.
67. Bell Numbers.
68. Stirling Numbers of the Second Kind.
69. Number of derangements (subfactorial).
70. Count number of subsets with even/odd sum.
71. Total ways to reach nth stair (climbing stairs problem).
72. Count binary strings without consecutive 1s.
73. Count n-digit numbers with digit sum = K.
74. Number of binary trees with N nodes.
75. Count lattice paths from (0, 0) to (N, N).
76. Number of trailing zeroes in binomial coefficient.
77. Check if binomial coefficient is divisible by a prime.
78. Count valid parenthesis expressions.
79. Distribute items into groups with constraints.
80. Count ways to make change using coins.
81. Count subset sums = K.
82. Count strings of length N that satisfy rules.
83. Count permutations with k inversions.
84. Distribute balls into boxes with at least one ball in each.
85. Count of positive integers ≤ N that are divisible by at least one in given set.

🔹 Advanced Number Theory (86–100)

86. Chinese Remainder Theorem.
87. Solve system of congruences.
88. Miller-Rabin primality test.
89. Pollard's Rho for factorization.
90. Finding primitive root modulo N.
91. Find number of primitive roots modulo prime.
92. Discrete logarithm (Baby Step Giant Step).
93. Find square root of number modulo N.
94. Compute multiplicative order modulo N.
95. Lucas Theorem for nCr % p.
96. Count of circular primes ≤ N.
97. Count of emirp (reversible) primes ≤ N.
98. Fermat factorization method.
99. Karatsuba multiplication for large numbers.
100. Find smallest primitive root of a prime.

🧠 Ready for:

- Any of these solved with code?
- Topic 20 next (like Geometry, Union-Find, Segment Trees, etc)?
- Or want a random daily challenge from all 19 topics?

You're building a legendary prep vault here 🔥