o Simple but inefficient for large input. o Example: Trying all passwords. 30 problem statement in one line

ChatGPT said:

Here are **30 one-line problem statements** that can be solved using the **Brute Force algorithm** approach:

1. Find all prime numbers up to N.
2. Check if any subset of an array sums to a given value.
3. Generate all permutations of a string.
4. Find the pair of numbers with a given sum in an array.
5. Check if a number can be expressed as a sum of two squares.
6. Count all palindromic substrings in a string.
7. Find the longest substring without repeating characters.
8. Search for a pattern in a text using naive pattern matching.
9. Find all triplets in an array that sum to zero.
10. Check if a graph contains a Hamiltonian path.
11. Find the minimum difference between any two elements in an array.
12. Determine if a Sudoku puzzle is solvable.
13. Solve the N-Queens problem.
14. Determine the shortest path by checking all paths in a graph.
15. Find the maximum subarray sum by checking all subarrays.
16. Find the closest pair of points in a 2D plane.
17. Generate all valid combinations of balanced parentheses.
18. Test all binary strings of length N for a given condition.
19. Count the number of anagrams of a word in a sentence.
20. Rotate an array and check all configurations for a condition.
21. Solve a cryptarithm puzzle by testing digit assignments.
22. Find all common substrings between two strings.
23. Check if one string is a rotation of another.
24. Find all divisors of a number and check a property.
25. Generate all combinations of k elements from a set.
26. Try all seating arrangements to minimize conflict.
27. Find all ways to climb stairs with 1 or 2 steps.
28. Try all permutations to solve a traveling salesman problem.
29. Count how many times a pattern appears in a matrix.
30. Check all possible ways to partition a set into two equal-sum subsets.

Let me know if you want code or explanations for any of these.

You said:

Divide and Conquer o Breaks the problem into smaller subproblems, solves them recursively, and combines the results. o Example: Merge Sort, Quick Sort, Binary Search.

ChatGPT said:

Here are **30 one-line problem statements** that can be solved using the **Divide and Conquer** approach:

1. Sort an array using Merge Sort.
2. Sort an array using Quick Sort.
3. Search for an element using Binary Search.
4. Find the maximum and minimum elements in an array.
5. Count the number of inversions in an array.
6. Find the closest pair of points in 2D space.
7. Compute the power of a number using exponentiation by squaring.
8. Multiply two large integers using Karatsuba algorithm.
9. Perform integer multiplication using Divide and Conquer.
10. Find the majority element in an array.
11. Find the k-th smallest element in an unsorted array.
12. Merge k sorted arrays into a single sorted array.
13. Compute the convex hull of a set of points.
14. Find the peak element in a 1D array.
15. Find the maximum subarray sum (using divide and conquer version of Kadane's).
16. Perform polynomial multiplication using FFT.
17. Search in a rotated sorted array.
18. Solve the Skyline problem using divide and conquer.
19. Compute large Fibonacci numbers efficiently.
20. Count the number of occurrences of a number in a sorted array.
21. Find the median of two sorted arrays.
22. Evaluate an arithmetic expression with parentheses and operators.
23. Determine if a word exists in a grid (recursive backtracking with D&C).
24. Solve the Tower of Hanoi problem.
25. Merge overlapping intervals using sorting and merging.
26. Count number of pairs with a given difference.
27. Solve recurrence relations using the Master Theorem.
28. Implement Strassen's Matrix Multiplication.
29. Find a local minimum in a 2D matrix.
30. Compute the determinant of a matrix recursively.

Let me know if you'd like Java/C++/Python code or diagrams for any of these problems.

You said:

Greedy Algorithm o Makes the locally optimal choice at each step. o Doesn't always give the best overall result. o Example: Kruskal's MST, Dijkstra's (without priority queue), Huffman

ChatGPT said:

Here are **30 one-line problem statements** that can be solved using the **Greedy Algorithm** approach:

1. Find the Minimum Spanning Tree using Kruskal's Algorithm.
2. Find the shortest path in a graph using Dijkstra's Algorithm (without priority queue).
3. Construct an optimal prefix code using Huffman Coding.
4. Select the maximum number of non-overlapping activities.
5. Find the minimum number of coins to make a given amount.
6. Choose the least number of platforms needed for train schedules.
7. Schedule jobs to maximize profit with deadlines.
8. Cover all points on a number line using the minimum number of segments.

9. Select minimum number of jumps to reach the end of an array.
10. Minimize the sum of absolute differences of pairs.
11. Find the minimum number of arrows to burst all balloons.
12. Distribute candies to children based on ratings.
13. Buy and sell stocks to maximize profit with at most one transaction.
14. Find the smallest number of intervals to remove to eliminate overlaps.
15. Assign minimum number of meeting rooms for scheduled meetings.
16. Form the largest possible number from a list of integers.
17. Minimize the number of taps to water a garden.
18. Allocate pages to students to minimize the maximum pages assigned.
19. Perform Egyptian Fraction representation of a given fraction.
20. Connect ropes with minimum total cost.
21. Select projects to maximize capital given k project limit.
22. Divide gold bars into equal parts with minimum cuts.
23. Select k elements such that the maximum difference is minimized.
24. Minimize cash flow among a group of friends.
25. Find the optimal way to cut a rod to maximize profit.
26. Construct a palindrome with minimum insertions.
27. Minimize the cost of cutting a chocolate bar.
28. Maximize the number of items that can be put in a knapsack (fractional knapsack).
29. Solve the activity selection problem from different time ranges.
30. Maximize the sum of array elements by flipping signs of K elements.

Let me know if you want solutions, diagrams, or coding help for any specific problem.

You said:

Dynamic Programming (DP) o Solves problems by breaking them into overlapping subproblems and storing the results (memoization or tabulation). o Example: Fibonacci, Knapsack, Longest Common Subsequence (LCS).

ChatGPT said:

Here are **30 one-line problem statements** that can be solved using the **Dynamic Programming (DP)** approach:

1. Compute the Nth Fibonacci number using memoization or tabulation.
2. Solve the 0/1 Knapsack problem to maximize profit.
3. Find the Longest Common Subsequence (LCS) of two strings.
4. Compute the Longest Increasing Subsequence (LIS) in an array.
5. Find the minimum number of coins to make a given amount.
6. Count the number of unique paths in a grid.
7. Solve the Matrix Chain Multiplication problem.
8. Find the minimum number of insertions to make a string palindrome.
9. Compute the Edit Distance between two strings.
10. Count the number of ways to make change for a given amount.
11. Solve the Rod Cutting problem to maximize profit.
12. Determine if a subset with a given sum exists in an array.
13. Find the number of subsets with a given sum.
14. Calculate the maximum profit from stock trading with k transactions.
15. Find the number of ways to decode a digit string.

16. Compute the maximum sum of a non-adjacent subsequence.
17. Solve the Coin Change problem for total number of ways.
18. Find the length of the longest palindromic subsequence.
19. Count the number of Binary Search Trees with n nodes (Catalan numbers).
20. Determine if a string can be segmented into dictionary words (Word Break).
21. Find the minimum cost path in a 2D matrix.
22. Solve the Egg Dropping Puzzle with minimum attempts.
23. Count the number of palindromic substrings in a string.
24. Find the maximum profit from a job scheduling problem.
25. Calculate the number of distinct subsequences of a string.
26. Count the ways to reach the Nth stair with 1 or 2 steps.
27. Solve the Boolean Parenthesization problem.
28. Find the minimum number of jumps to reach the end of the array.
29. Solve the Painter's Partition problem to minimize time.
30. Compute the optimal score in the Matrix Game problem.

Let me know if you want **Java/C++/Python code**, **explanations**, or **DP table structures** for any of these.

You said:

Backtracking o Tries all solutions and backtracks if one doesn't lead to a solution. o Example: N-Queens, Sudoku Solver.

ChatGPT said:

Here are **30 one-line problem statements** that can be solved using the **Backtracking** approach:

1. Solve the N-Queens problem.
2. Solve a Sudoku puzzle.
3. Generate all permutations of a string or array.
4. Generate all combinations of k elements from n numbers.
5. Find all subsets (power set) of a given set.
6. Generate all valid combinations of balanced parentheses.
7. Solve the Rat in a Maze problem.
8. Solve the Knight's Tour problem on a chessboard.
9. Generate all possible binary strings of length N.
10. Solve the Word Search problem in a 2D grid.
11. Find all unique combinations that sum up to a target (Combination Sum).
12. Partition a string into all possible palindromic partitions.
13. Find all Hamiltonian paths in a graph.
14. Color a graph using m colors without adjacent nodes having the same color.
15. Solve the M-Coloring Problem for graphs.
16. Generate all magic squares of size N.
17. Find all valid arrangements of placing K knights on a chessboard.
18. Generate all lexicographically ordered permutations.
19. Solve the Crossword Puzzle Fill-In problem.
20. Find all possible placements of Tetris blocks on a grid.
21. Solve the Subset Sum problem.
22. Generate all possible expressions to evaluate to a target number.

23. Build all valid IP addresses from a string of digits.
24. Place L-shaped tiles on a grid to cover it completely.
25. Solve the 8-puzzle sliding game.
26. Generate all topological sorts of a DAG.
27. Solve the Boggle word game using a dictionary.
28. Rearrange an array to meet a specific constraint (e.g., no two adjacent elements equal).
29. Generate all valid Sudoku boards (not just solve).
30. Find all ways to assign signs (+/-) to make a target sum.

Let me know if you'd like **code samples**, **flowcharts**, or **step-by-step trace** for any of these problems!

You said:

Recursion o Function calls itself with a smaller input. o Used in divide and conquer, backtracking, and dynamic programming. o Example: Factorial, Tower of Hanoi.

ChatGPT said:

Here are **30 one-line problem statements** that can be solved using **Recursion**:

1. Compute the factorial of a number.
2. Solve the Tower of Hanoi problem.
3. Find the nth Fibonacci number.
4. Print numbers from 1 to N recursively.
5. Calculate the sum of digits of a number.
6. Reverse a string using recursion.
7. Check if a string is a palindrome using recursion.
8. Count the number of elements in a list.
9. Find the greatest common divisor (GCD) of two numbers.
10. Convert a decimal number to binary recursively.
11. Generate all subsets (power set) of a set.
12. Generate all permutations of a string.
13. Compute the sum of the first N natural numbers.
14. Find the maximum element in an array recursively.
15. Print all paths from top-left to bottom-right in a matrix.
16. Count the number of ways to climb stairs (1 or 2 steps).
17. Multiply two numbers using recursive addition.
18. Solve the N-Queens problem using recursion.
19. Find the length of a string recursively.
20. Merge two sorted arrays recursively.
21. Compute exponentiation (a^b) using recursion.
22. Search for an element in a list recursively.
23. Count ways to tile a 2×N board with 2×1 tiles.
24. Find the sum of elements in an array recursively.
25. Solve the subset sum problem using recursion.
26. Print all permutations of an array recursively.
27. Remove duplicates from a string using recursion.
28. Check if an array is sorted recursively.
29. Flatten a nested list structure using recursion.

30. Generate nth Catalan number using recursion.

Let me know if you want **code**, **recursion trees**, or **dry run examples** for any of these!

You said:

Graph Algorithms o BFS, DFS o Dijkstra's, Bellman-Ford o Floyd-Warshall, A* o Kruskal's and Prim's for MST

ChatGPT said:

Here are **30 one-line problem statements** that can be solved using **Graph Algorithms**:

🔁 Traversal Algorithms (BFS, DFS):

1. Traverse all nodes in a graph using Breadth-First Search (BFS).
2. Traverse all nodes in a graph using Depth-First Search (DFS).
3. Check if a path exists between two nodes using DFS.
4. Check if a graph is bipartite using BFS or DFS.
5. Detect a cycle in an undirected graph using DFS.
6. Detect a cycle in a directed graph using DFS.
7. Perform topological sorting of a directed acyclic graph.
8. Count the number of connected components in a graph.
9. Find all nodes reachable from a given node.
10. Solve a maze using BFS or DFS.

□ Shortest Path Algorithms:

11. Find the shortest path from a source to all nodes using Dijkstra's algorithm.
12. Find the shortest path with negative weights using Bellman-Ford algorithm.
13. Detect negative weight cycle using Bellman-Ford.
14. Find all-pairs shortest paths using Floyd-Warshall algorithm.
15. Find the shortest path in a weighted grid using Dijkstra's.
16. Solve the Single Source Shortest Path problem in a DAG.
17. Find the shortest path using A* algorithm with heuristics.
18. Find the shortest path in a binary matrix using BFS.

🌉 Minimum Spanning Tree (MST) Algorithms:

19. Find a Minimum Spanning Tree using Kruskal's algorithm.
20. Find a Minimum Spanning Tree using Prim's algorithm.
21. Check if an MST is unique.
22. Construct MST of a weighted undirected graph.
23. Find the second-best Minimum Spanning Tree.

🔄 Advanced Graph Problems:

24. Count the number of strongly connected components (Kosaraju's algorithm).
25. Find articulation points and bridges in a graph.
26. Detect critical connections in a network.
27. Solve the Union-Find problem for dynamic connectivity.
28. Perform bipartite matching in a graph.
29. Find the maximum flow using Ford-Fulkerson or Edmonds-Karp.
30. Determine the minimum number of edges to make a graph connected.

Let me know if you want **C++/Java/Python implementations**, **graph visuals**, or **step-by-step traces** for any of these problems!

You said:

Sorting Algorithms o Bubble, Insertion, Selection o Merge Sort, Quick Sort, Heap Sort o Counting, Radix, and Bucket Sort (non-comparison sorts)

ChatGPT said:

Here are **30 one-line problem statements** that can be solved using **Sorting Algorithms** (Comparison and Non-Comparison based):

🔁 Basic Comparison Sorts: Bubble, Insertion, Selection

1. Sort an array using Bubble Sort.
2. Sort an array using Insertion Sort.
3. Sort an array using Selection Sort.
4. Sort student marks using Bubble Sort.
5. Sort a list of names using Insertion Sort.
6. Sort array and count swaps made by Bubble Sort.
7. Sort a nearly sorted array using Insertion Sort.
8. Sort an array in descending order using Selection Sort.
9. Implement adaptive Insertion Sort for partially sorted input.
10. Sort characters of a string alphabetically using Bubble Sort.

⚡ Efficient Divide & Conquer Sorts: Merge, Quick, Heap

11. Sort an array using Merge Sort.
12. Sort an array using Quick Sort (Lomuto or Hoare partition).
13. Sort an array using Heap Sort.
14. Sort and count inversions using Merge Sort.

15. Find the Kth smallest element using Quick Sort partitioning.
16. Use Quick Sort to sort an array of strings.
17. Implement in-place Merge Sort.
18. Visualize sorting steps using Heap Sort.
19. Sort an array of structs/objects using Merge Sort.
20. Sort a nearly sorted (K-sorted) array using Heap Sort.

🔢 Non-Comparison Sorts: Counting, Radix, Bucket

21. Sort non-negative integers using Counting Sort.
22. Sort ages of people (0–150) using Counting Sort.
23. Sort large integers using Radix Sort.
24. Sort strings of equal length using Radix Sort.
25. Sort floating-point numbers using Bucket Sort.
26. Sort an array with only 0s, 1s, and 2s (Dutch National Flag problem).
27. Sort an array with repeated elements efficiently using Counting Sort.
28. Group elements by frequency and sort using Counting technique.
29. Sort ZIP codes using Radix Sort.
30. Sort decimal scores (0.0–1.0) using Bucket Sort.

Let me know if you'd like **code**, **performance comparisons**, or **visual step-by-step sorting animations**!

You said:

9. Searching Algorithms o Linear Search o Binary Search o Ternary Search (less common)

ChatGPT said:

Here are **30 one-line problem statements** that can be solved using **Searching Algorithms**:

🔍 Linear Search:

1. Search for a key in an unsorted array using Linear Search.
2. Find the first occurrence of a character in a string.
3. Search for a student's name in an unsorted list.
4. Count occurrences of a given number in an array.
5. Find all positions of a target element in an array.
6. Check if an element exists in a linked list.
7. Find the minimum element in an unsorted array.
8. Check if any element is greater than a given threshold.
9. Find the last occurrence of a number using Linear Search.
10. Determine if a string contains all vowels using Linear Search.

⚡ Binary Search (on sorted data):

11. Search for an element in a sorted array using Binary Search.
12. Find the first occurrence of a number in a sorted array.
13. Find the last occurrence of a number in a sorted array.
14. Count occurrences of an element in a sorted array using Binary Search.
15. Find the square root of a number using Binary Search.
16. Search in a rotated sorted array.
17. Find the peak element in a mountain array.
18. Find the smallest element greater than a target.
19. Determine the insert position of an element in a sorted array.
20. Find the element that appears only once when all others appear twice.

🔀 Ternary Search (on unimodal functions or arrays):

21. Find the maximum value in a unimodal function using Ternary Search.
22. Find the minimum of a convex function (e.g., time to complete a task).
23. Find the point where a function reaches minimum latency.
24. Optimize the search for peak in a bitonic array.
25. Find the closest value to a target in a strictly decreasing function.
26. Find the minimum time required to paint boards by painters.
27. Find optimal cutting height for a tree-cutting machine.
28. Optimize cost function in logistics using Ternary Search.
29. Find best angle/position to shoot a projectile in a physics simulation.
30. Solve mathematical optimization problems with unimodal properties.

Let me know if you'd like **code examples**, **use-case explanations**, or **complexity analysis** for any of these!

You said:

10. Bit Manipulation Algorithms o XOR tricks, Bit masking o Used in problems involving subsets or toggling bits. 🔲 🔲 Algorithm Design Techniques

ChatGPT said:

Here are **30 one-line problem statements** that can be solved using **Bit Manipulation Algorithms**:

❌ XOR Tricks:

1. Find the number that appears only once when others appear twice.
2. Find the two non-repeating elements in an array where others repeat twice.
3. Swap two numbers without using a temporary variable using XOR.
4. Determine if two numbers are equal using XOR.
5. Check if a number is a power of two using bitwise operations.
6. Count the number of bits to flip to convert A to B.
7. Find the XOR of all elements in an array.
8. Detect if there's a duplicate using XOR and range sum property.
9. Find the missing number from 0 to n using XOR.
10. Compute XOR of numbers from 1 to N efficiently.

□ Bit Masking and Subsets:

11. Generate all subsets of a set using bit masking.
12. Count the number of set bits (1s) in an integer.
13. Check if the ith bit is set in a number.
14. Turn off the rightmost set bit.
15. Get the rightmost set bit.
16. Set the ith bit of a number.
17. Clear the ith bit of a number.
18. Toggle the ith bit of a number.
19. Generate power set using bits.
20. Find subset with given sum using bitmasking.

⚙ Binary Manipulation Tricks:

21. Check if a number has alternating bits (like 101010...).
22. Divide a number by 2 using bit shift.
23. Multiply a number by 2 using bit shift.
24. Check if a number is even or odd using bitwise AND.
25. Find the position of the only set bit in a power of two.
26. Turn off all bits after the rightmost set bit.
27. Find the next higher number with the same number of set bits.
28. Reverse bits of an integer.
29. Find the binary representation of a number.
30. Calculate parity (even or odd number of 1s) of a binary number.

Would you like **code (C++/Java)**, **bitwise truth tables**, or **visual explanation** for any of these problems?

You said:

ChatGPT said:

Here are **30 one-line problem statements** that use **Recursion and Memoization**—two key **Algorithm Design Techniques**:

🔁 Recursion + 🎁 Memoization = Efficient Solutions

1. Compute the Nth Fibonacci number using recursion with memoization.
2. Solve the 0/1 Knapsack problem using recursion and memoization.
3. Count the number of ways to climb stairs (1 or 2 steps at a time).
4. Compute Edit Distance between two strings.
5. Solve the Longest Common Subsequence (LCS) problem.
6. Find the minimum number of coins needed for a given amount.
7. Count the number of binary search trees with N nodes (Catalan number).
8. Solve the Matrix Chain Multiplication problem.
9. Find the length of the Longest Increasing Subsequence.
10. Count the number of palindromic substrings in a string.
11. Partition an array into two subsets with equal sum.
12. Solve the Word Break problem using recursion and memoization.
13. Count the number of unique paths in a grid with obstacles.
14. Compute the minimum cost to reach the top of stairs.
15. Solve the Rod Cutting problem with maximum profit.
16. Determine if a string is a scramble of another.
17. Compute the nth Tribonacci number.
18. Count the number of distinct subsequences of a string.
19. Find the number of ways to decode a digit string (like "123").
20. Solve the Boolean Parenthesization problem.
21. Compute the optimal strategy for a game (e.g., pick-from-ends).
22. Count the number of ways to tile a 2×N board.
23. Calculate the total number of ways to partition a set.
24. Solve the Egg Dropping puzzle with minimum attempts.
25. Count the ways to form a target string from a dictionary.
26. Find the minimum insertions to make a string a palindrome.
27. Solve the Wildcard Pattern Matching problem.
28. Find the maximum path sum in a binary tree.
29. Solve the Painting Fence problem.
30. Compute the number of ways to assign + and – to reach a target sum.

Let me know if you want these grouped into **DP categories**, or need **code** in **Java/C++/Python** with **memo tables explained**!

You said:

Greedy Choice Property

ChatGPT said:

Here are **30 one-line problem statements** that apply the **Greedy Choice Property** — a core concept in **Greedy Algorithm Design**:

✅ Greedy Choice Property

*At each step, choose the **best immediate (local) option**, hoping it leads to a global optimum.*

💡 Classic Greedy Problems:

1. Select the maximum number of non-overlapping intervals (Activity Selection).
2. Minimize the number of coins for a given amount (Coin Change - greedy version).
3. Build an optimal prefix-free code (Huffman Coding).
4. Find the Minimum Spanning Tree using Kruskal's algorithm.
5. Find the Minimum Spanning Tree using Prim's algorithm.
6. Schedule jobs to maximize total profit with deadlines.
7. Cover all points on a number line using the minimum number of segments.
8. Distribute candies to children based on ratings (Candy Problem).
9. Buy and sell stock for maximum profit with one transaction.
10. Find the minimum number of platforms required for trains.

🔁 Greedy in Arrays & Strings:

11. Choose minimum jumps to reach end of array.
12. Remove K digits to get the smallest number.
13. Construct the largest possible number from a list of integers.
14. Find the minimum number of arrows to burst all balloons.
15. Assign minimum number of meeting rooms.
16. Select minimum number of intervals to remove to avoid overlap.
17. Partition labels in a string so that no letter appears in more than one part.
18. Schedule the maximum number of tasks without overlap.
19. Choose meetings based on earliest end time.
20. Pick up maximum weight items without exceeding capacity (Fractional Knapsack).

🎯 Advanced Greedy Choices:

21. Allocate books to minimize the maximum pages read.
22. Minimize the total cost of connecting ropes.
23. Maximize value of items in knapsack with fractional parts allowed.
24. Optimize painting of houses with minimum cost.
25. Greedily assign projects to maximize capital.
26. Solve Egyptian Fraction representation.

27. Maximize number of events attended in given time slots.
28. Divide people into groups minimizing the maximum group size.
29. Choose tasks to maximize utilization of a single processor.
30. Optimize gas refill stops during a long journey.

Would you like to see **proof of the greedy-choice property**, **exchange argument**, or **Java/C++ implementations** for any of these problems?

You said:

Optimal Substructure

ChatGPT said:

Here are **30 one-line problem statements** that exhibit the **Optimal Substructure** property — a key trait in **Dynamic Programming** and **Greedy Algorithms**:

✅ Optimal Substructure

*A problem has **optimal substructure** if an optimal solution can be built from **optimal solutions of its subproblems**.*

🎁 Classic DP Problems with Optimal Substructure:

1. Find the nth Fibonacci number.
2. Compute the maximum profit in 0/1 Knapsack problem.
3. Find the Longest Common Subsequence (LCS) of two strings.
4. Calculate the minimum number of coins to make change.
5. Find the Edit Distance (Levenshtein distance) between two strings.
6. Count the number of ways to climb stairs with 1 or 2 steps.
7. Find the maximum sum of a non-adjacent subsequence.
8. Compute the minimum cost path in a matrix.
9. Solve the Matrix Chain Multiplication problem.
10. Determine the minimum number of jumps to reach the end of an array.

🏙 Advanced DP Examples:

11. Count the number of ways to decode a digit string.
12. Find the length of the Longest Increasing Subsequence (LIS).
13. Find the length of the Longest Palindromic Subsequence.
14. Count the number of unique paths in a grid.

15. Solve the Boolean Parenthesization problem.
16. Solve the Egg Dropping puzzle.
17. Solve the Rod Cutting problem for maximum profit.
18. Compute the total number of distinct subsequences.
19. Find the maximum score in a game with pick-from-ends strategy.
20. Partition an array into two subsets with equal sum.

🔀 Greedy Problems with Optimal Substructure:

21. Select the maximum number of non-overlapping activities.
22. Construct an optimal Huffman tree.
23. Build a Minimum Spanning Tree using Kruskal's or Prim's algorithm.
24. Solve job scheduling to maximize profit.
25. Solve the Fractional Knapsack problem.
26. Distribute tasks to minimize the number of machines needed.
27. Optimize number of meeting rooms required.
28. Find the minimum number of coins (when greedy works).
29. Solve the minimum number of platforms needed for trains.
30. Find optimal merge patterns for files or ropes.

Would you like explanations for **how optimal substructure applies** to any specific problem above — or how to **differentiate it from greedy vs. DP use-cases**?

You said:

State Space Tree (used in Backtracking)

ChatGPT said:

Here are **30 one-line problem statements** that involve the concept of a **State Space Tree**, commonly used in **Backtracking** problems:

🌳 State Space Tree (Backtracking)

A **State Space Tree** represents all possible states of a problem where each node corresponds to a partial solution, and the tree is explored using **DFS-style backtracking** to build full solutions.

✅ Classic Backtracking Problems Using State Space Tree

1. Solve the N-Queens problem for a given N.

2. Generate all permutations of a string.
3. Generate all subsets (power set) of a set.
4. Find all valid combinations of k numbers from 1 to n.
5. Generate all valid parenthesis combinations for n pairs.
6. Solve a Sudoku puzzle.
7. Generate all permutations of a list of numbers with duplicates (handling duplicates).
8. Find all combinations that sum to a target value (Combination Sum I).
9. Find all unique combinations that sum to a target (Combination Sum II).
10. Find all palindromic partitions of a string.

🔁 Backtracking in Constraint Problems

11. Solve the Knight's Tour problem on an N×N board.
12. Color a graph using m colors such that no adjacent nodes share the same color.
13. Solve the Hamiltonian Path problem.
14. Solve the Rat in a Maze problem.
15. Place k knights on an N×N chessboard such that no two attack each other.
16. Fill a crossword puzzle with given words.
17. Partition a set into K subsets with equal sums.
18. Find all valid assignments for a cryptarithmetic puzzle (e.g., SEND + MORE = MONEY).
19. Solve the Word Search problem in a 2D board.
20. Generate all Latin Squares of size N×N.

📐 Number and Digit-Based Backtracking

21. Generate all binary strings of length N with no two consecutive 1s.
22. Find all expressions by adding +, -, * between digits to reach a target value.
23. Solve the Subset Sum problem.
24. Generate all magic squares of size N×N.
25. Generate numbers with increasing digits only.
26. Construct all n-digit numbers with given sum of digits.
27. Count all N-digit numbers formed using a specific set of digits.
28. Find all prime number combinations of a given number.
29. Generate all valid IP addresses from a given string.
30. Find all possible letter combinations of a phone number.

Would you like to see a **visual example** of a state space tree (e.g., for N-Queens or subsets)?

You said:

Sliding Window

ChatGPT said:

Here are **30 one-line problem statements** based on the **Sliding Window** technique — a powerful approach for solving problems on arrays or strings where you move a window of fixed or variable size across the data.

□ Sliding Window Problems

✅ Fixed-size Sliding Window

1. Find the maximum sum of any subarray of size `k`.
2. Find the minimum sum of any subarray of size `k`.
3. Count subarrays of size `k` whose average is greater than or equal to a target.
4. Check if any subarray of size `k` contains all distinct elements.
5. Find the first negative number in every window of size `k`.
6. Find the maximum in every sliding window of size `k`.
7. Find the minimum in every sliding window of size `k`.
8. Compute the median of every window of size `k`.
9. Count number of anagrams of a pattern in a text using a window of pattern size.
10. Count windows of size `k` with sum divisible by a given number `d`.

✅ Variable-size Sliding Window (with Two Pointers)

11. Find the length of the longest substring with all unique characters.
12. Find the longest substring with at most `k` distinct characters.
13. Find the length of the smallest subarray with sum ≥ target.
14. Count the number of subarrays with sum exactly `k`.
15. Count subarrays with at most `k` distinct elements.
16. Find the maximum length subarray with sum exactly `k`.
17. Find the number of subarrays with product less than `k`.
18. Find the longest substring with at most `k` zeros (binary string).
19. Find the longest substring containing same letters after k replacements.
20. Count substrings with exactly `k` distinct characters.

✅ Special & String-Based Window Problems

21. Find the smallest substring containing all characters of a pattern.
22. Find the minimum window that contains all characters of another string.
23. Find the longest substring without repeating characters.
24. Count all substrings where frequency of each character is at most 1.
25. Find all start indices of pattern anagrams in a string.
26. Longest substring with equal number of 0s and 1s.
27. Longest substring with same number of vowels and consonants.
28. Count substrings with exactly one distinct character.
29. Find the longest substring made of one repeating character with at most one swap.

30. Maximum sum of a subarray with unique elements.

Let me know if you'd like C++ or Java code examples, visual walkthroughs, or detailed explanations for any of these!

You said:

Two Pointers

ChatGPT said:

Here are **30 one-line problem statements** based on the **Two Pointers** technique — a popular approach for solving array, string, and linked list problems efficiently using two indices.

□ Two Pointers Technique Problems

✅ Basic Array & Searching

1. Find a pair in a sorted array whose sum equals a given target.
2. Find if there exists a triplet in an array that adds up to a given value.
3. Count the number of pairs with a given sum in a sorted and rotated array.
4. Remove duplicates from a sorted array in-place.
5. Merge two sorted arrays into one sorted array.
6. Check if an array is a palindrome using two pointers.
7. Move all zeros to the end while maintaining the relative order of other elements.
8. Partition an array into two subarrays with equal sum.
9. Find the maximum length of a subarray with sum <= K.
10. Count the number of pairs in a sorted array whose product is less than a target.

✅ String and Substring Problems

11. Check if two strings are anagrams using two pointers.
12. Find the longest palindromic substring.
13. Reverse vowels in a string using two pointers.
14. Check if a string is a palindrome (ignore spaces and cases).
15. Compress a string in-place (like run-length encoding).
16. Find the first occurrence of a pattern in a text using two pointers.
17. Remove all adjacent duplicates in a string (iteratively).
18. Compare two strings considering backspaces (`#`).
19. Merge two strings alternatively using two pointers.
20. Replace all spaces in a string with `%20`.

✅ Linked List Based

21. Find the middle of a linked list using slow and fast pointers.
22. Detect a cycle in a linked list.
23. Find the starting point of the cycle in a linked list.
24. Remove the nth node from the end of a linked list.
25. Check if a linked list is a palindrome.
26. Rearrange a linked list into odd-even indexed nodes.
27. Merge two sorted linked lists.
28. Find the intersection node of two linked lists.
29. Find the kth node from the end of the linked list.
30. Sort a linked list using the two-pointer merge sort approach.

Let me know if you want **C++/Java code**, visual **dry runs**, or detailed **explanations** for any of the problems.

You said:

Union-Find / Disjoint Set

ChatGPT said:

Here are **30 one-line problem statements** based on the **Union-Find / Disjoint Set** data structure — useful for solving problems related to **connected components**, **cycle detection**, and **network connectivity**.

🔗 Union-Find / Disjoint Set Problems

✅ Basic Operations and Structure

1. Implement Union-Find with path compression.
2. Count the number of disjoint sets after a series of union operations.
3. Determine if two elements belong to the same set.
4. Merge two sets if they are different.
5. Implement Union by Rank and Path Compression together.
6. Track the size of each disjoint set dynamically.
7. Return the number of elements in the same set as a given element.
8. Find the representative (parent) of a node in Union-Find.
9. Initialize Union-Find for N elements and support `find` and `union`.
10. Optimize Union-Find for near constant-time operations.

✅ Graphs: Connected Components and Cycle Detection

11. Detect a cycle in an undirected graph using Union-Find.

12. Count the number of connected components in an undirected graph.
13. Find the redundant connection (extra edge forming a cycle) in a graph.
14. Check if the given undirected graph is a valid tree.
15. Detect if adding an edge creates a cycle in a dynamic graph.
16. Check if all nodes in a graph are connected using Union-Find.
17. Number of provinces (connected groups of cities).
18. Count the number of islands in a 2D grid using Union-Find.
19. Merge all similar string groups.
20. Count the number of friend circles in a social network.

✅ Kruskal's Algorithm (Minimum Spanning Tree)

21. Use Union-Find to implement Kruskal's Algorithm.
22. Given a list of weighted edges, find the MST cost.
23. Skip edges forming a cycle while building MST.
24. Track the components merged during Kruskal's process.
25. Find the number of edges in the MST of a given graph.

✅ Advanced / Real-Life Application Problems

26. Determine if equations like `a==b` and `b!=c` are satisfiable.
27. Group emails by common users (accounts merge problem).
28. Count the number of unique groups of equivalent strings.
29. Determine the number of connected computers in a network.
30. Find the maximum number of edges that can be removed while keeping the graph fully traversable (Alice and Bob type problems).

Let me know if you'd like code, visuals, or union-find implementation with **path compression**, **rank optimization**, or even **union by size**.

You said:

Monotonic Stack/Queue

ChatGPT said:

Here are **30 one-line problem statements** focused on the **Monotonic Stack / Monotonic Queue** technique — useful for problems involving **next greater/smaller elements**, **sliding windows**, and **range queries**.

□ Monotonic Stack / Queue Problems

✅ Monotonic Stack – Basic

1. Find the next greater element for each element in an array.
2. Find the next smaller element for each element in an array.
3. Find the previous greater element for each element.
4. Find the previous smaller element for each element.
5. Count the number of days until a warmer temperature (Daily Temperatures).
6. For each bar in a histogram, find the nearest smaller to left and right.
7. Calculate the largest rectangle in a histogram.
8. Find the largest rectangle of 1's in a binary matrix.
9. Given a circular array, find the next greater element for each index.
10. Find span of stock prices using previous greater element.

✅ Monotonic Stack – Intermediate/Advanced

11. Count the number of valid mountain subarrays.
12. Evaluate the minimum number of steps to make array non-decreasing.
13. Make the array strictly increasing with minimum removals.
14. Determine the visibility count for each building in a skyline.
15. Count number of rectangles that can be formed with given heights.
16. Find the length of longest valid parentheses substring.
17. Decode an expression with operator precedence using monotonic stack.
18. Implement a custom parser using operator precedence stack.
19. Maintain a stack to validate a series of operations or expressions.
20. Simulate falling squares on a 1D line and compute heights.

✅ Monotonic Queue – Sliding Window

21. Find the maximum value in each sliding window of size k.
22. Find the minimum value in each sliding window of size k.
23. Count subarrays where the max - min $\leq$ a given limit.
24. Compute the length of the longest subarray under an absolute difference limit.
25. Maintain a deque to track the sliding window minimum efficiently.
26. Find maximum in all subarrays of length exactly K using a queue.
27. Track the maximum number of requests in a time window.
28. Find number of valid windows with monotonic property.
29. Use deque to simulate max queue or min queue behavior.
30. Implement Monotonic Queue from scratch with `push`, `pop`, and `max`.

Let me know if you want C++ or Java code templates for **Monotonic Stack**, **Sliding Window Maximum**, or problems like **Histogram Rectangle**, etc.

You said:

Array Fixed-size, index-based collection of elements.

ChatGPT said:

Here are **30 one-line problem statements** focused on **Arrays** — a foundational data structure that uses **fixed-size, index-based** storage for elements.

□ Array Problem Statements

✅ Basics

1. Find the sum of all elements in an array.
2. Find the maximum and minimum elements in an array.
3. Calculate the average of array elements.
4. Reverse the elements of an array in-place.
5. Count the number of even and odd elements in the array.
6. Find the frequency of each element in the array.
7. Check if the array is sorted in non-decreasing order.
8. Search for an element in an unsorted array.
9. Remove duplicates from a sorted array.
10. Move all zeroes to the end while maintaining the order of other elements.

✅ Intermediate

11. Rotate the array to the right by `k` positions.
12. Find the second largest element in the array.
13. Find the element that appears only once in an array where every other element appears twice.
14. Return the index of a peak element in the array.
15. Find the subarray with the maximum sum (Kadane's algorithm).
16. Find the longest subarray with sum equal to a given number `k`.
17. Count the number of subarrays with a given XOR.
18. Determine if a subarray with 0 sum exists.
19. Rearrange the array in alternating positive and negative items.
20. Find the first repeating element in the array.

✅ Advanced

21. Find the majority element (appears more than n/2 times).
22. Count the number of inversions in the array.
23. Find the smallest missing positive number.
24. Merge two sorted arrays without extra space.
25. Find all triplets with sum zero in the array.
26. Compute the maximum product of any three elements.

27. Replace every element with the greatest element on right.
28. Find equilibrium index in the array.
29. Maximum difference between two elements such that larger appears after smaller.
30. Implement a prefix sum array and solve range sum queries.

Let me know if you'd like:

- C++ or Java code snippets
- Input/output examples
- Problems grouped by difficulty or concept (e.g., prefix sums, sliding window, etc.)

You said:

Linked List Nodes connected by pointers (Singly/Doubly/ Circular)

ChatGPT said:

Here are **30 one-line Linked List problem statements** covering **Singly**, **Doubly**, and **Circular** linked lists:

➿ Linked List Problem Statements

✅ Singly Linked List (SLL)

1. Insert a node at the beginning of a singly linked list.
2. Insert a node at the end of a singly linked list.
3. Delete the first occurrence of a given value in a singly linked list.
4. Reverse a singly linked list iteratively.
5. Reverse a singly linked list recursively.
6. Find the middle node of a singly linked list.
7. Detect a cycle in a singly linked list (Floyd's algorithm).
8. Find the starting node of the cycle in a singly linked list.
9. Merge two sorted singly linked lists into one.
10. Remove duplicates from a sorted singly linked list.

✅ Doubly Linked List (DLL)

11. Insert a node at the beginning of a doubly linked list.
12. Insert a node at the end of a doubly linked list.
13. Delete a node from the beginning of a doubly linked list.
14. Delete a node from the end of a doubly linked list.
15. Reverse a doubly linked list.
16. Find the length of a doubly linked list.
17. Insert a node after a given node in a doubly linked list.

18. Delete a node with a given value from a doubly linked list.
19. Check if a doubly linked list is a palindrome.
20. Convert a doubly linked list to a balanced BST.

✅ Circular Linked List (CLL)

21. Create a circular linked list from an array.
22. Insert a node at the end of a circular linked list.
23. Insert a node at the beginning of a circular linked list.
24. Delete a node from a circular linked list.
25. Traverse a circular linked list once.
26. Split a circular linked list into two halves.
27. Check if a circular linked list contains only one node.
28. Count the number of nodes in a circular linked list.
29. Detect a loop in a circular linked list.
30. Convert a circular linked list to a singly linked list.

Let me know if you want:

- Code examples (Java/C++/Python)
- Visual diagrams
- Input/output test cases
- Interview-style questions with answers

Would you like the **Queue** problems next?

You said:

Stack LIFO (Last-In-First-Out) structure. Operations: push, pop, peek.

ChatGPT said:

Here are **30 one-line Stack problem statements** that cover core usage, advanced techniques, and interview-level challenges:

□ Stack Problem Statements (LIFO Structure)

✅ Basic Stack Operations

1. Implement a stack using arrays.
2. Implement a stack using linked list.
3. Push and pop elements from a stack.
4. Peek the top element of a stack.
5. Check if a stack is empty.

6. Reverse a string using a stack.
7. Evaluate a postfix expression using a stack.
8. Convert infix expression to postfix using stack.
9. Convert infix expression to prefix using stack.
10. Check for balanced parentheses using a stack.

🔁 Advanced Stack Applications

11. Implement a stack that supports `getMin()` in O(1) time.
12. Sort a stack using another stack.
13. Delete the middle element of a stack.
14. Implement two stacks in a single array.
15. Design a stack that supports `push()`, `pop()`, `top()`, and `getMin()` in O(1).
16. Find the next greater element for every array element using a stack.
17. Find the next smaller element for every array element using a stack.
18. Evaluate a prefix expression using a stack.
19. Remove adjacent duplicates from a string using a stack.
20. Find the length of the longest valid parentheses substring.

📶 Monotonic Stack Problems

21. Find the largest rectangle in a histogram.
22. Find the maximum area of a rectangle in a binary matrix.
23. Implement a monotonic increasing stack.
24. Implement a monotonic decreasing stack.
25. Find the number of days until a warmer temperature (Daily Temperatures).
26. Trapping Rain Water problem using a stack.
27. Find stock span for each day using a stack.
28. Remove K digits from a number to make it smallest using stack.
29. Build a stack that supports `max()` in O(1).
30. Decode a string encoded with `k[encoded_string]` using a stack.

Would you like these problems solved in **Java or C++** with input/output examples?

Let me know if you're ready for **Queue problems next**!

You said:

Queue FIFO (First-In-First-Out) structure. Variants: Queue, Deque, Priority Queue.

ChatGPT said:

Here are **30 one-line Queue problem statements** that include core operations, real-world applications, and competitive programming challenges:

📥 Queue Problem Statements (FIFO Structure)

✅ Basic Queue Operations

1. Implement a queue using an array.
2. Implement a queue using a linked list.
3. Enqueue and dequeue elements from a queue.
4. Peek the front element of the queue.
5. Check if the queue is empty.
6. Implement a circular queue.
7. Implement a queue using two stacks.
8. Implement a stack using two queues.
9. Reverse the first K elements of a queue.
10. Interleave the first half of the queue with the second half.

🔄 Deque (Double Ended Queue) Problems

11. Implement a deque using a doubly linked list.
12. Insert and delete from both ends of a deque.
13. Check if a given sequence is a palindrome using deque.
14. Design a circular deque.
15. Sliding window maximum using deque.
16. Sliding window minimum using deque.
17. Sum of minimum and maximum of all subarrays of size k using deque.
18. Design a data structure that supports `getMin()` and `getMax()` in O(1) using deque.
19. Find the first negative integer in every window of size k.
20. Maximum of all subarrays of size k.

🧠 Priority Queue (Heap) Problems

21. Implement a min heap using priority queue.
22. Implement a max heap using priority queue.
23. Merge k sorted arrays using min heap.
24. Find the kth largest element in an array.
25. Find the kth smallest element in an array.
26. Find the top K frequent elements.
27. Find the median from a stream of integers using two heaps.
28. Sort a nearly sorted (k-sorted) array.
29. Connect n ropes with minimum cost.
30. Design a scheduler to process jobs by priority and deadline using priority queue.

Would you like the **solutions in Java or C++** with test cases?

Next: Want **Tree** or **Hashing** problems?

You said:

• Tree Hierarchical structure. Types: Binary Tree, Binary Search Tree (BST), AVL Tree, Segment Tree, etc.

ChatGPT said:

Here are **50 one-line problem statements** on **Trees**, covering everything from basic Binary Trees to advanced Segment Trees and Binary Search Trees (BSTs):

🌳 Tree Problem Statements

🌗 Binary Tree – Basics (15 Problems)

1. Implement a Binary Tree using classes and pointers.
2. Perform Inorder, Preorder, and Postorder traversals.
3. Level order traversal (BFS) of a binary tree.
4. Height of a binary tree.
5. Count the number of nodes in a binary tree.
6. Count the number of leaf nodes in a binary tree.
7. Find the diameter of a binary tree.
8. Check if two binary trees are identical.
9. Mirror a binary tree.
10. Convert binary tree to its sum tree.
11. Find the lowest common ancestor (LCA) of two nodes.
12. Print all root-to-leaf paths in a binary tree.
13. Check if a binary tree is balanced (height-balanced).
14. Boundary traversal of a binary tree.
15. Vertical order traversal of a binary tree.

□ Binary Search Tree (BST) – Operations (10 Problems)

16. Insert a node into a BST.
17. Delete a node from a BST.
18. Search a node in a BST.
19. Find the minimum value in a BST.
20. Find the kth smallest element in a BST.
21. Convert a BST to a sorted Doubly Linked List.
22. Check if a binary tree is a valid BST.
23. Find the in-order successor and predecessor of a given node.
24. Find the lowest common ancestor (LCA) in BST.

25. Recover a BST in which two nodes are swapped.

🔁 Tree – Recursive + DFS Based (10 Problems)

26. Count total number of full nodes (nodes with 2 children).
27. Count number of half nodes (nodes with 1 child).
28. Check if a tree is symmetric (mirror around center).
29. Convert binary tree to its mirror using recursion.
30. Flatten a binary tree to a linked list (in-place).
31. Find sum of all nodes at a given level K.
32. Calculate maximum width of a binary tree.
33. Convert binary tree to doubly linked list (inorder-wise).
34. Serialize and deserialize a binary tree.
35. Find the longest path from root to leaf.

📊 Segment Tree / Range Queries (10 Problems)

36. Build a segment tree for range sum query.
37. Perform range sum query and point updates using segment tree.
38. Build a segment tree for range minimum query (RMQ).
39. Perform lazy propagation for range updates.
40. Count number of elements greater than x in a range.
41. Find the first index with value ≥ x using segment tree.
42. Find the maximum prefix sum in a given range.
43. Segment Tree with XOR operation for range.
44. Merge two segment trees.
45. Segment Tree for frequency count of elements.

□ Advanced Tree Topics (5 Problems)

46. Build a tree from inorder and preorder traversals.
47. Build a tree from inorder and postorder traversals.
48. Convert N-ary Tree to Binary Tree (and vice versa).
49. Find distance between two nodes in a binary tree.
50. Find maximum path sum between any two nodes.

Would you like these problems in **Java**, **C++**, or **Python**, with solutions and test cases?

Next: Want **Hash Table / Trie / Graph** problems?

You said:

Graph Collection of nodes (vertices) and edges. Types: Directed, Undirected, Weighted, Unweighted.

ChatGPT said:

Here's a concise summary of **Graph** data structures, continuing your outline:

Graph

- A **graph** is a collection of **nodes (vertices)** connected by **edges**.
- **Types of Graphs**:
  - **Directed Graph (Digraph)**: Edges have direction.
  - **Undirected Graph**: Edges do not have direction.
  - **Weighted Graph**: Each edge has a weight/cost.
  - **Unweighted Graph**: Edges have no weight.
- **Common Representations**:
  - **Adjacency Matrix**: 2D array, good for dense graphs.
  - **Adjacency List**: Array of lists, efficient for sparse graphs.
- **Common Applications**:
  - Networking (routing)
  - Social networks
  - Web crawling
  - Shortest path and connectivity problems
- **Graph Algorithms (referenced earlier)**:
  - **Traversal**: BFS, DFS
  - **Shortest Path**: Dijkstra's, Bellman-Ford, Floyd-Warshall
  - **Minimum Spanning Tree**: Prim's, Kruskal's
  - **Cycle Detection, Topological Sort, SCCs**, etc.

Would you like a chart or table version of this full DSA outline?

You said:

Graph Collection of nodes (vertices) and edges. Types: Directed, Undirected, Weighted, Unweighted. 30 problem statement in one line

ChatGPT said:

Here are **30 one-line problem statements** based on **Graph** data structures (Directed, Undirected, Weighted, Unweighted):

🔁 Graph Problem Statements (1-liners)

1. Find if a path exists between two nodes in an undirected graph.
2. Detect a cycle in an undirected graph.
3. Detect a cycle in a directed graph.

4. Count the number of connected components in an undirected graph.
5. Perform Breadth-First Search (BFS) traversal from a given node.
6. Perform Depth-First Search (DFS) traversal from a given node.
7. Find the shortest path in an unweighted graph using BFS.
8. Find the shortest path in a weighted graph using Dijkstra's Algorithm.
9. Detect negative weight cycles using Bellman-Ford Algorithm.
10. Compute all-pairs shortest paths using Floyd-Warshall Algorithm.
11. Check if a graph is bipartite (2-colorable).
12. Topologically sort a directed acyclic graph (DAG).
13. Count the number of islands in a 2D grid using DFS.
14. Implement Union-Find to check if two nodes are in the same component.
15. Find Minimum Spanning Tree (MST) using Kruskal's Algorithm.
16. Find MST using Prim's Algorithm.
17. Check if an undirected graph is a tree.
18. Reconstruct itinerary (Eulerian path) in a directed graph.
19. Find bridges (critical connections) in an undirected graph.
20. Find articulation points (cut vertices) in an undirected graph.
21. Clone an undirected graph.
22. Find all paths from source to target in a DAG.
23. Find mother vertex in a directed graph.
24. Count number of triangles in an undirected graph.
25. Detect Hamiltonian Path in a graph.
26. Detect Eulerian Cycle in a graph.
27. Solve the course schedule problem (cycle detection + topological sort).
28. Find the center of a tree (minimum height root).
29. Find the shortest path from source to all vertices in a DAG.
30. Use BFS to simulate zombie infection spread in a city grid.

Let me know if you want:

- 🔍 Explanations
- ✅ Input/Output examples
- 💡 Level-wise categorization (Easy/Medium/Hard)
- 📄 Solutions in C++ or Java

Ready to help!

You said:

Hash Table / HashMap Key-value storage with average O(1) access time.

ChatGPT said:

Here are **30 one-line problem statements** based on **Hash Table / HashMap** (Key-Value Pair, O(1) Access):

🧠 Hash Table / HashMap – 30 Problem Statements

1. Find the first non-repeating character in a string.
2. Find the frequency of each element in an array.
3. Check if two strings are anagrams.
4. Find the intersection of two arrays.
5. Find the union of two arrays.
6. Count pairs with a given sum in an array.
7. Check if a subarray with sum zero exists.
8. Find the length of the longest subarray with zero sum.
9. Group words that are anagrams.
10. Find the most frequent element in an array.
11. Two Sum problem using hashmap.
12. Find all duplicates in an array.
13. Count the number of distinct elements in a window of size k.
14. Find the longest consecutive sequence in an array.
15. Implement LRU Cache using HashMap + Doubly Linked List.
16. Check if a permutation of one string is a substring of another.
17. Count number of subarrays with a given XOR.
18. Detect and remove loop in a linked list using Hashing.
19. Check if a pair with a given difference exists in an array.
20. Subdomain visit count using string and hashmap.
21. Count characters that appear more than once.
22. Determine if a string can be rearranged to form a palindrome.
23. Count number of unique k-diff pairs in an array.
24. Find all substrings of a string with exactly k distinct characters.
25. Find the maximum length of a substring without repeating characters.
26. Count frequency of each word in a paragraph.
27. Store and retrieve employee records using ID (key).
28. Implement custom HashMap with basic `put()` and `get()`.
29. Track the first unique number in a stream.
30. Find elements occurring more than n/3 times in an array (Extended Majority Element).

Let me know if you want:

- 📥 Input/Output examples
- ✅ C++/Java implementations
- 🌟 Easy/Medium/Hard classification
- 🛠️ Practice links or explanations

Just say the word!

You said:

Trie (Prefix Tree) – For efficient string searches

ChatGPT said:

Here are **30 one-line problem statements** based on the **Trie (Prefix Tree)** data structure, ideal for efficient string operations like prefix search, autocomplete, and dictionary matching:

📚 Trie (Prefix Tree) – 30 Problem Statements

1. Insert a word into a Trie.
2. Search for a word in a Trie.
3. Check if any word in the Trie starts with a given prefix.
4. Implement autocomplete suggestions based on prefix.
5. Count the total number of words stored in a Trie.
6. Delete a word from the Trie.
7. Find the longest prefix that matches a word from the Trie.
8. Return all words in a Trie with a given prefix.
9. Implement a word dictionary with support for `.` as wildcard.
10. Count unique words in a large dictionary.
11. Find the most frequent word with a given prefix.
12. Design a contact search system using Trie.
13. Spell checker using Trie for dictionary matching.
14. Find all words that differ by at most one character from a given word.
15. Check if a given string is a concatenation of two or more dictionary words.
16. Detect if two words differ by exactly one character (Magic Dictionary).
17. Find the shortest unique prefix for every word in a list.
18. Find all palindrome pairs from a list of words using Trie.
19. Group words by shared prefix.
20. Find all suffixes of a word using a reversed Trie.
21. Replace words in a sentence with the shortest root form using Trie.
22. Find the longest word that can be built one character at a time.
23. Count how many times a word has been inserted into the Trie.
24. Build a Trie to search IP addresses with prefix matching.
25. Create a Trie that is case-insensitive.
26. Suggest product names based on typed characters.
27. Find all substrings that are valid words using Trie.
28. Implement a multi-lingual dictionary using Trie.
29. Find the longest common prefix of a list of strings.
30. Use Trie to perform prefix-based search in a large document.

Let me know if you'd like:

- Code examples (Java/C++)
- Input/output test cases
- Categorization (easy/medium/hard)
- Practice problems from LeetCode/GeeksforGeeks

Just say “expand” next to any number!

You said:

• Disjoint Set (Union-Find) – For tracking connected

ChatGPT said:

Here are **30 one-line problem statements** based on the **Disjoint Set (Union-Find)** data structure, which is useful for tracking connected components, detecting cycles, and dynamic connectivity.

⚯ Disjoint Set (Union-Find) – 30 Problem Statements

1. Check if adding an edge to a graph creates a cycle.
2. Find the number of connected components in an undirected graph.
3. Determine if two nodes are in the same connected component.
4. Merge two sets using union by rank.
5. Implement path compression in find operation.
6. Kruskal's Algorithm to find Minimum Spanning Tree.
7. Count the number of friend circles (groups of mutual friends).
8. Determine if all nodes are connected in a graph.
9. Count the number of provinces in a city connection matrix.
10. Merge accounts with overlapping emails.
11. Check if a graph is a valid tree.
12. Find the earliest time when all people become friends.
13. Determine if two nodes are connected after several union operations.
14. Number of islands using Union-Find (grid connectivity).
15. Find redundant connection in an undirected graph.
16. Track dynamic island formation in a grid as land is added.
17. Determine the smallest group of people knowing each other directly or indirectly.
18. Find if an equation system with equalities and inequalities is satisfiable.
19. Group similar strings using Union-Find.
20. Connected groups of servers in a data center (2D matrix).
21. Implement Union-Find with union by size.
22. Track connected components as edges are added to a graph.
23. Union-Find to detect cycle in a directed graph (with back edges).
24. Check if a bipartite graph exists using disjoint sets.
25. Group people by common interests using Union-Find.
26. Create an efficient friendship network using Union-Find.
27. Evaluate connectivity in a grid with dynamic wall removal.
28. Merge sets of overlapping intervals.
29. Use Union-Find to find critical connections (bridges) in a network.
30. Find the number of distinct groups in a social network.

Let me know if you'd like:

- Code snippets in Java/C++/Python
- Input-output test cases
- Categorized problem levels

- Real-world applications for projects/interviews

Just reply with the number or say **"expand"**!