

Here are **50 one-line problem statements on Arrays** — perfect for coding practice in Java, C++, or any language 📌

◆ Basic Array Problems

1. Find the largest element in an array.
 2. Find the smallest element in an array.
 3. Calculate the sum of all elements in an array.
 4. Calculate the average of array elements.
 5. Count even and odd elements in an array.
 6. Reverse the elements of an array.
 7. Copy elements from one array to another.
 8. Find the second largest element in an array.
 9. Find the second smallest element in an array.
 10. Check if a given element exists in the array.
-

◆ Searching & Sorting

Read this Blog with Leeco AI

Learn smarter with interactive doubt solving, quizzes & examples.

Stop reading and start learning.

Start Learning

11. Implement linear search in an array.
 12. Implement binary search on a sorted array.
 13. Sort an array using Bubble Sort.
 14. Sort an array using Selection Sort.
 15. Sort an array using Insertion Sort.
 16. Merge two sorted arrays into one sorted array.
 17. Find the Kth smallest element in an array.
 18. Find the Kth largest element in an array.
 19. Check if the array is sorted in ascending order.
 20. Remove duplicates from a sorted array.
-

◆ Mathematical / Logical Operations

21. Find the sum of all positive numbers in an array.
22. Count the frequency of each element in an array.
23. Find the element that appears only once (others appear twice).
24. Find the missing number from 1 to N in an array.
25. Find all pairs with a given sum.
26. Move all zeroes to the end of the array.

27. Find the intersection of two arrays.
 28. Find the union of two arrays.
 29. Rotate an array by K positions to the right.
 30. Rotate an array by K positions to the left.
-

◆ Advanced / Logical Thinking

31. Find the maximum product subarray.
 32. Find the maximum sum subarray (Kadane's algorithm).
 33. Find the majority element (appears more than $n/2$ times).
 34. Rearrange array elements in alternating positive and negative order.
 35. Separate even and odd numbers in an array.
 36. Find the first repeating element in an array.
 37. Find the first non-repeating element in an array.
 38. Replace each element with the next greater element to its right.
 39. Find the equilibrium index of an array (sum left = sum right).
 40. Find the longest increasing subsequence in an array.
-

◆ 2D Array / Matrix Based

41. Find the sum of all elements in a 2D array.
 42. Find the transpose of a matrix.
 43. Multiply two matrices.
 44. Print the diagonal elements of a square matrix.
 45. Find the sum of diagonal elements of a matrix.
 46. Check if a matrix is symmetric.
 47. Rotate a matrix by 90 degrees clockwise.
 48. Search an element in a row-wise and column-wise sorted matrix.
 49. Find the maximum element in each row of a matrix.
 50. Find the minimum element in each column of a matrix.
-

Would you like me to make **20 of these into coding problems with sample input/output** (for Java or C++) next?

You said:

matrix

ChatGPT said:

Here are **50 one-line problem statements on Matrix (2D Arrays)** — covering basics to advanced topics 📄

◆ Basic Matrix Operations

1. Read and print a matrix.
 2. Find the sum of all elements in a matrix.
 3. Find the average of all elements in a matrix.
 4. Find the maximum element in a matrix.
 5. Find the minimum element in a matrix.
 6. Count even and odd elements in a matrix.
 7. Count positive, negative, and zero elements in a matrix.
 8. Replace all negative numbers with zero in a matrix.
 9. Find the sum of each row in a matrix.
 10. Find the sum of each column in a matrix.
-

◆ Diagonal & Triangular Matrices

11. Print the main diagonal elements of a square matrix.
 12. Print the secondary diagonal (anti-diagonal) elements of a matrix.
 13. Find the sum of main diagonal elements.
 14. Find the sum of secondary diagonal elements.
 15. Check if a matrix is diagonal.
 16. Check if a matrix is upper triangular.
 17. Check if a matrix is lower triangular.
 18. Find the trace of a matrix (sum of diagonal elements).
 19. Find the sum of both diagonals of a square matrix.
 20. Replace diagonal elements with zero.
-

◆ Matrix Transformations

21. Find the transpose of a matrix.
 22. Check if a matrix is symmetric.
 23. Rotate a matrix by 90° clockwise.
 24. Rotate a matrix by 90° anticlockwise.
 25. Rotate a matrix by 180° .
 26. Reverse each row of a matrix.
 27. Reverse each column of a matrix.
 28. Flip a matrix horizontally.
 29. Flip a matrix vertically.
 30. Convert a matrix into its mirror image.
-

◆ Arithmetic Operations

31. Add two matrices.
32. Subtract one matrix from another.

33. Multiply two matrices.
 34. Multiply a matrix by a scalar value.
 35. Find element-wise product (Hadamard product) of two matrices.
 36. Check if two matrices are equal.
 37. Find the difference between diagonal sums.
 38. Find the determinant of a 2×2 or 3×3 matrix.
 39. Find the inverse of a 2×2 matrix.
 40. Compute the adjoint of a 2×2 matrix.
-

◆ Searching & Patterns

41. Search for an element in a matrix.
 42. Search for an element in a row-wise and column-wise sorted matrix.
 43. Print the boundary elements of a matrix.
 44. Print a matrix in spiral order.
 45. Print a matrix in zigzag order.
 46. Print the upper half of a matrix.
 47. Print the lower half of a matrix.
 48. Sort each row of a matrix individually.
 49. Sort each column of a matrix individually.
 50. Check if two matrices are identical after rotation.
-

Would you like me to create **20 coding problems from this list with sample input and output** (in **Java** or **C++**) next?

You said:

queue

ChatGPT said:

Here are **50 one-line problem statements on Queues** — from basics to advanced (using Array, LinkedList, or STL/Java Queue). 📄

◆ Basic Queue Operations

1. Implement a queue using an array.
2. Implement a queue using a linked list.
3. Implement a queue using two stacks.
4. Implement a stack using two queues.
5. Enqueue and dequeue elements in a queue.
6. Find the front element of a queue.
7. Find the rear element of a queue.
8. Check if a queue is empty.

9. Check if a queue is full (for array-based queue).
 10. Display all elements of a queue.
-

◆ Circular Queue

11. Implement a circular queue using an array.
 12. Implement a circular queue using a linked list.
 13. Perform enqueue and dequeue operations in a circular queue.
 14. Check if a circular queue is full or empty.
 15. Count the total elements in a circular queue.
 16. Print all elements of a circular queue.
 17. Reset a circular queue.
 18. Find the size of a circular queue dynamically.
 19. Reverse a circular queue.
 20. Merge two circular queues into one.
-

◆ Deque (Double Ended Queue)

21. Implement a deque using an array.
 22. Implement a deque using a linked list.
 23. Insert and delete elements from both ends of a deque.
 24. Check if a deque is palindrome.
 25. Find the maximum element in a deque.
 26. Reverse a deque.
 27. Implement input-restricted deque.
 28. Implement output-restricted deque.
 29. Delete all elements from a deque.
 30. Rotate a deque by 'k' positions.
-

◆ Queue Applications

31. Generate binary numbers from 1 to N using a queue.
 32. Implement a queue to perform BFS traversal of a graph.
 33. Implement a queue to perform level-order traversal of a binary tree.
 34. Check if all parentheses in an expression are balanced using a queue.
 35. Implement a printer job queue simulation.
 36. Implement a task scheduling system using a queue.
 37. Find the first non-repeating character in a stream using a queue.
 38. Reverse the first K elements of a queue.
 39. Interleave the first and second halves of a queue.
 40. Sort a queue using another queue.
-

◆ Advanced / Problem Solving

41. Find the maximum of all subarrays of size K (sliding window maximum).
 42. Find the minimum of all subarrays of size K.
 43. Check if a given sequence of operations is valid for a queue.
 44. Count the number of elements between two specific elements in a queue.
 45. Implement a circular tour problem (petrol pump problem) using a queue.
 46. Implement a queue that supports getMin() in O(1).
 47. Implement multiple queues using a single array.
 48. Reverse the entire queue using recursion.
 49. Check if two queues are identical.
 50. Design a queue that supports dynamic resizing.
-

Would you like me to make **20 coding problems with sample input and output** from this queue list (in **Java** or **C++**) next?

You said:

stack (using stl)

ChatGPT said:

Here are **50 one-line problem statements on Stack (using STL in C++)** — from basic to advanced applications ☺

◆ Basic Stack Operations (STL `stack`)

1. Push and pop elements using STL stack.
 2. Display all elements in a stack.
 3. Find the top element of a stack.
 4. Check if a stack is empty.
 5. Find the size of a stack.
 6. Reverse a stack using another stack.
 7. Reverse a stack using recursion.
 8. Copy one stack into another.
 9. Delete all elements from a stack.
 10. Check if two stacks are identical.
-

◆ Mathematical / Expression Problems

11. Check if a string has balanced parentheses.
12. Check if a string has balanced brackets {}, [], and ().
13. Convert an infix expression to postfix using stack.

14. Convert an infix expression to prefix using stack.
 15. Evaluate a postfix expression.
 16. Evaluate a prefix expression.
 17. Check if an infix expression is valid.
 18. Find the next greater element for every element in an array using stack.
 19. Find the next smaller element for every element in an array using stack.
 20. Implement a function to check if a mathematical expression is correctly parenthesized.
-

◆ Array / Sequence Based Stack Problems

21. Find the span of stock prices for each day.
 22. Find the nearest greater element to the left for each element.
 23. Find the nearest smaller element to the left for each element.
 24. Find the nearest greater element to the right for each element.
 25. Find the nearest smaller element to the right for each element.
 26. Find the largest rectangular area in a histogram.
 27. Find the maximum area of a rectangle in a binary matrix.
 28. Find the next greater frequency element for each array element.
 29. Sort a stack using another stack.
 30. Insert an element at the bottom of a stack using recursion.
-

◆ String / Character Based Problems

31. Reverse a string using a stack.
 32. Check if a string is palindrome using a stack.
 33. Remove adjacent duplicate characters using stack.
 34. Simplify a given Unix-style file path using stack.
 35. Decode a string with nested brackets like "3[a2[c]]".
 36. Convert a binary string into its decimal equivalent using stack.
 37. Validate HTML/XML tags using stack.
 38. Check if two strings are equal after backspace operations.
 39. Remove minimum parentheses to make a string valid.
 40. Evaluate arithmetic expression with parentheses using stack.
-

◆ Advanced / Logical Stack Applications

41. Implement two stacks in a single array.
42. Implement a min-stack that returns the minimum element in $O(1)$.
43. Implement a max-stack that returns the maximum element in $O(1)$.
44. Implement a stack that supports `getMiddle()` in $O(1)$.
45. Implement a stack using one queue.
46. Implement a stack using two queues.

47. Check if a given sequence of push/pop operations is valid for a stack.
48. Implement a special stack where even numbers are pushed at the bottom.
49. Delete the middle element of a stack using recursion.
50. Sort stack elements in ascending order using recursion.

Would you like me to turn **20 of these stack problems into full coding problems with sample input/output** (in C++ using STL) next?

You said:

Tree

ChatGPT said:

Here are **50 one-line problem statements on Trees** — including Binary Tree, Binary Search Tree (BST), AVL Tree, and General Tree topics 🌳

🌳 Basic Tree Operations

1. Create a binary tree and perform preorder traversal.
2. Create a binary tree and perform inorder traversal.
3. Create a binary tree and perform postorder traversal.
4. Perform level-order (BFS) traversal of a binary tree.
5. Count the total number of nodes in a binary tree.
6. Count the number of leaf nodes in a binary tree.
7. Find the height (or depth) of a binary tree.
8. Find the diameter of a binary tree.
9. Find the sum of all nodes in a binary tree.
10. Mirror a binary tree.

🌳 Binary Search Tree (BST)

11. Insert a node in a Binary Search Tree.
 12. Delete a node from a Binary Search Tree.
 13. Search for a node in a Binary Search Tree.
 14. Find the minimum value in a BST.
 15. Find the maximum value in a BST.
 16. Find the inorder predecessor and successor in a BST.
 17. Check if a binary tree is a valid BST.
 18. Find the Lowest Common Ancestor (LCA) in a BST.
 19. Convert a BST into a sorted doubly linked list.
 20. Find the Kth smallest element in a BST.
-

🌲 Tree Traversal Variations

21. Print all nodes at a given level.
 22. Print the left view of a binary tree.
 23. Print the right view of a binary tree.
 24. Print the top view of a binary tree.
 25. Print the bottom view of a binary tree.
 26. Print all leaf nodes from left to right.
 27. Print the boundary traversal of a binary tree.
 28. Perform zigzag (spiral) level order traversal.
 29. Print nodes in vertical order traversal.
 30. Print nodes in diagonal order traversal.
-

🔧 Advanced Binary Tree Problems

31. Check if two binary trees are identical.
 32. Check if one tree is a subtree of another.
 33. Check if a binary tree is symmetric (mirror of itself).
 34. Find the distance between two nodes in a binary tree.
 35. Convert a binary tree into its mirror.
 36. Find the maximum path sum in a binary tree.
 37. Check if a binary tree is balanced.
 38. Check if a binary tree is complete.
 39. Count the number of full nodes in a binary tree.
 40. Find the deepest node in a binary tree.
-

🌿 Generic / N-ary / AVL / Special Trees

41. Create a general (N-ary) tree and perform DFS traversal.
 42. Create a general (N-ary) tree and perform BFS traversal.
 43. Find the height of a general (N-ary) tree.
 44. Insert a node in an AVL tree and maintain balance.
 45. Delete a node from an AVL tree and maintain balance.
 46. Check if an AVL tree is balanced after insertion.
 47. Convert a binary tree into a sum tree.
 48. Construct a binary tree from inorder and preorder traversals.
 49. Construct a binary tree from inorder and postorder traversals.
 50. Serialize and deserialize a binary tree.
-

Would you like me to make **20 coding problems with input/output examples** from these **Tree** questions (in **C++ or Java**)?

If yes — please specify the language.

You said:

Linked list

ChatGPT said:

Here are **50 one-line problem statements on Linked Lists** — covering **Singly, Doubly, and Circular Linked Lists**, from beginner to advanced 📝

◆ Basic Singly Linked List Operations

1. Create a singly linked list and display its elements.
 2. Insert a node at the beginning of a linked list.
 3. Insert a node at the end of a linked list.
 4. Insert a node at a given position in a linked list.
 5. Delete a node from the beginning of a linked list.
 6. Delete a node from the end of a linked list.
 7. Delete a node from a specific position in a linked list.
 8. Search for an element in a linked list.
 9. Count the number of nodes in a linked list.
 10. Reverse a singly linked list iteratively.
-

◆ Intermediate Singly Linked List Problems

11. Reverse a singly linked list recursively.
 12. Find the middle element of a linked list.
 13. Find the Nth node from the end of a linked list.
 14. Detect a loop in a linked list using Floyd's cycle detection algorithm.
 15. Remove a loop from a linked list.
 16. Merge two sorted linked lists.
 17. Sort a linked list using merge sort.
 18. Check if a linked list is palindrome.
 19. Remove duplicate elements from a sorted linked list.
 20. Remove duplicate elements from an unsorted linked list.
-

◆ Advanced Singly Linked List Problems

21. Rotate a linked list by K nodes.
22. Swap two nodes in a linked list (without swapping data).
23. Pairwise swap nodes in a linked list.
24. Find the intersection point of two linked lists.
25. Find the length of a loop in a linked list.
26. Add two numbers represented by linked lists.

27. Delete nodes that have greater value on the right.
 28. Segregate even and odd nodes in a linked list.
 29. Move last node to the front of a linked list.
 30. Clone a linked list with random pointers.
-

◆ Doubly Linked List Problems

31. Create a doubly linked list and display its elements.
 32. Insert a node at the beginning of a doubly linked list.
 33. Insert a node at the end of a doubly linked list.
 34. Delete a node from a doubly linked list.
 35. Reverse a doubly linked list.
 36. Find the length of a doubly linked list.
 37. Convert a singly linked list into a doubly linked list.
 38. Find pairs with a given sum in a sorted doubly linked list.
 39. Delete all occurrences of a given key in a doubly linked list.
 40. Check if a doubly linked list is palindrome.
-

◆ Circular Linked List Problems

41. Create a circular linked list and display it.
 42. Insert a node at the beginning of a circular linked list.
 43. Insert a node at the end of a circular linked list.
 44. Delete a node from a circular linked list.
 45. Split a circular linked list into two halves.
 46. Check if a linked list is circular or not.
 47. Convert a singly linked list into a circular linked list.
 48. Solve the Josephus problem using a circular linked list.
 49. Find the last remaining node in a circular elimination game.
 50. Merge two circular linked lists.
-

Would you like me to make **20 coding problems with input/output examples** from these **Linked List** questions (in **C++ or Java**)?

If yes — please specify the **language**.

You said:

HashMap

ChatGPT said:

Here are **50 one-line problem statements on HashMap (or unordered_map in C++)** — covering **basic to advanced** topics 📄

◆ Basic HashMap Operations

1. Create a HashMap and insert key-value pairs.
 2. Retrieve a value from a HashMap using a key.
 3. Check if a key exists in a HashMap.
 4. Check if a value exists in a HashMap.
 5. Remove a key-value pair from a HashMap.
 6. Get the size of a HashMap.
 7. Iterate over all key-value pairs in a HashMap.
 8. Clear all elements from a HashMap.
 9. Check if a HashMap is empty.
 10. Copy one HashMap to another.
-

◆ Counting & Frequency Problems

11. Count the frequency of each element in an array using a HashMap.
 12. Count the frequency of characters in a string using a HashMap.
 13. Find the first non-repeating character in a string.
 14. Find the first repeating element in an array.
 15. Find elements that appear more than once in an array.
 16. Find the element with the maximum frequency.
 17. Find the element with the minimum frequency.
 18. Find common elements between two arrays using a HashMap.
 19. Find the intersection of two arrays using HashMap.
 20. Find the union of two arrays using HashMap.
-

◆ Pair & Sum Related Problems

21. Find all pairs with a given sum in an array using HashMap.
 22. Check if there exists a subarray with a given sum.
 23. Find the length of the longest subarray with a given sum.
 24. Find the length of the longest subarray with sum zero.
 25. Count the number of subarrays with sum equal to K.
 26. Find the subarray with the maximum sum using prefix sums and HashMap.
 27. Check if two arrays are equal (contain same elements) using HashMap.
 28. Find the number of pairs with a given difference.
 29. Find all pairs with XOR equal to K using HashMap.
 30. Find the count of distinct elements in every window of size K.
-

◆ String & Character Problems

31. Check if two strings are anagrams using HashMap.
 32. Find all anagram pairs in a list of strings.
 33. Find the longest substring without repeating characters.
 34. Find the longest substring with at most K distinct characters.
 35. Find the smallest window in a string containing all characters of another string.
 36. Group words that are anagrams using HashMap.
 37. Check if two strings are isomorphic using HashMap.
 38. Count word frequencies in a sentence using HashMap.
 39. Find duplicate words in a paragraph using HashMap.
 40. Replace words in a sentence using a given HashMap dictionary.
-

◆ Advanced / Logical HashMap Applications

41. Check if an array can be divided into pairs with equal sum using HashMap.
 42. Find the longest consecutive sequence in an unsorted array.
 43. Find the smallest subarray with all unique elements.
 44. Find two numbers that sum up to a target (Two Sum problem).
 45. Find four elements that sum up to a target (Four Sum using HashMap).
 46. Count the number of pairs with equal sum using HashMap.
 47. Implement a LRU (Least Recently Used) cache using HashMap + list.
 48. Implement a frequency-based cache using HashMap.
 49. Design a data structure that supports insert, delete, and getRandom in O(1) using HashMap.
 50. Check if two arrays have the same set of elements (order doesn't matter) using HashMap.
-

Would you like me to create **20 coding problems with sample input and output** from these **HashMap** questions (in **Java** or **C++**)?

If yes — please specify the **language**.

You said:

BST

ChatGPT said:

Here are **50 one-line problem statements on Binary Search Trees (BSTs)** — covering **basic operations, traversal, and advanced problems** 📄

♣ Basic BST Operations

1. Insert a node in a BST.
2. Delete a node from a BST.
3. Search for a value in a BST.

4. Find the minimum value in a BST.
 5. Find the maximum value in a BST.
 6. Find the height of a BST.
 7. Count the total number of nodes in a BST.
 8. Count the number of leaf nodes in a BST.
 9. Check if a BST is empty.
 10. Find the parent of a given node in a BST.
-

✿ BST Traversal Problems

11. Perform inorder traversal of a BST.
 12. Perform preorder traversal of a BST.
 13. Perform postorder traversal of a BST.
 14. Perform level-order (BFS) traversal of a BST.
 15. Find the kth smallest element in a BST.
 16. Find the kth largest element in a BST.
 17. Find the inorder predecessor of a given node.
 18. Find the inorder successor of a given node.
 19. Find the sum of all nodes in a BST.
 20. Find the range sum of nodes between two values.
-

🌲 BST Validation & Checking

21. Check if a binary tree is a valid BST.
 22. Check if a BST is balanced.
 23. Find the diameter of a BST.
 24. Check if a BST contains duplicate values.
 25. Find the level of a node in a BST.
 26. Find the distance between two nodes in a BST.
 27. Find the lowest common ancestor (LCA) of two nodes.
 28. Find the maximum width of a BST.
 29. Check if two BSTs are identical.
 30. Check if one BST is a subtree of another.
-

📌 BST Construction & Conversion

31. Construct a BST from a sorted array.
32. Construct a BST from preorder traversal.
33. Construct a BST from postorder traversal.
34. Construct a BST from level-order traversal.
35. Convert a BST to a sorted doubly linked list.
36. Convert a BST into a greater sum tree.
37. Convert a sorted linked list into a BST.

38. Serialize and deserialize a BST.
 39. Flatten a BST to a linked list.
 40. Build a BST from given inorder and preorder sequences.
-

🌿 Advanced BST Problems

41. Find the closest value to a given number in a BST.
 42. Count the number of nodes in a BST within a given range.
 43. Find the vertical order traversal of a BST.
 44. Find the top view of a BST.
 45. Find the bottom view of a BST.
 46. Find the left view of a BST.
 47. Find the right view of a BST.
 48. Check if there exists a pair with a given sum in a BST.
 49. Merge two BSTs into a balanced BST.
 50. Find all paths in a BST that sum to a given value.
-

I can also create **20 of these BST problems with coding examples and input/output in C++ or Java** if you want.

Do you want me to do that next?

You said:

heap

ChatGPT said:

Here are **50 one-line problem statements on Heaps / Priority Queues** — covering **min-heap, max-heap, and advanced heap applications** 🖱

◆ Basic Heap Operations

1. Build a max-heap from an array.
2. Build a min-heap from an array.
3. Insert an element into a max-heap.
4. Insert an element into a min-heap.
5. Delete the maximum element from a max-heap.
6. Delete the minimum element from a min-heap.
7. Get the maximum element from a max-heap.
8. Get the minimum element from a min-heap.
9. Heapify a subtree in a max-heap.
10. Heapify a subtree in a min-heap.

◆ Heap Sorting & Selection Problems

11. Perform heap sort on an array in ascending order.
12. Perform heap sort on an array in descending order.
13. Find the kth largest element in an array using a heap.
14. Find the kth smallest element in an array using a heap.
15. Find the median of a stream of numbers using a heap.
16. Merge k sorted arrays using a min-heap.
17. Find the top k frequent elements in an array using a heap.
18. Sort a nearly sorted (k-sorted) array using a heap.
19. Find the running median of a stream using a heap.
20. Find the smallest range that includes elements from k lists.

◆ Heap Applications in Priority Scheduling

21. Implement a priority queue using a heap.
22. Implement a min-priority queue using a heap.
23. Implement a max-priority queue using a heap.
24. Simulate a task scheduling system using a heap.
25. Merge multiple priority queues into one.
26. Find the CPU task execution order using a max-heap.
27. Simulate an event-driven system using a heap.
28. Find the minimum time to finish jobs using a heap.
29. Process tasks with varying priorities using a heap.
30. Solve the “k closest points to origin” problem using a heap.

◆ Advanced / Problem Solving with Heap

31. Find the kth largest sum of contiguous subarrays.
32. Find the kth smallest sum of contiguous subarrays.
33. Find the maximum sum of k elements from n arrays.
34. Connect n ropes with minimum cost using a min-heap.
35. Find the running maximum in a sliding window using a heap.
36. Find the smallest number with given prime factors using a heap.
37. Implement a double-ended priority queue using two heaps.
38. Find k pairs with smallest sums from two arrays.
39. Find the largest sum of non-overlapping subarrays using a heap.
40. Sort elements by frequency using a heap.

◆ Graph / Network / Miscellaneous Heap Problems

41. Implement Dijkstra's shortest path algorithm using a min-heap.
42. Implement Prim's Minimum Spanning Tree algorithm using a heap.
43. Implement Huffman encoding using a min-heap.
44. Find the minimum cost to connect sticks using a heap.
45. Find k-th largest element in a stream of data using a heap.
46. Implement a median heap using two heaps (min-heap & max-heap).
47. Maintain top k scores in a game leaderboard using a heap.
48. Schedule jobs with deadlines and profits using a max-heap.
49. Find the k largest elements in a max-heap without extracting all elements.
50. Implement a merge-k-sorted-lists function using a heap.

I can also create **20 coding problems from these Heap statements with sample input/output in C++ or Java** if you want.

Do you want me to do that next?

You said:

trie

ChatGPT said:

Here are **50 one-line problem statements on Trie (Prefix Tree)** — covering **basic operations, strings, and advanced applications** 📄

◆ Basic Trie Operations

1. Insert a word into a Trie.
2. Search for a word in a Trie.
3. Delete a word from a Trie.
4. Check if a Trie is empty.
5. Count the total words stored in a Trie.
6. Count the total nodes in a Trie.
7. Display all words in a Trie in lexicographical order.
8. Count words with a given prefix.
9. Check if a given prefix exists in the Trie.
10. Find the longest common prefix of all words in a Trie.

◆ Word / String Problems using Trie

11. Auto-complete suggestions for a given prefix.
12. Find all words in the Trie starting with a given prefix.
13. Find the longest word in the Trie.
14. Find the shortest word in the Trie.

15. Count the frequency of a word in the Trie.
 16. Insert words with duplicate counts into a Trie.
 17. Find all anagrams from a list using Trie.
 18. Search for words with one character mismatch.
 19. Find all words matching a pattern with wildcards (like * or ?).
 20. Implement a case-insensitive Trie.
-

◆ Advanced Trie Applications

21. Find the longest common prefix among multiple strings.
 22. Implement a spell checker using a Trie.
 23. Find the maximum XOR of two numbers using a binary Trie.
 24. Find all palindrome pairs in a list of words.
 25. Find all words that can be formed by concatenating two words from a list.
 26. Count the number of words that can be formed from a given set of characters.
 27. Find the word with the maximum length in the Trie.
 28. Implement prefix search with ranking based on frequency.
 29. Solve the “replace words” problem using Trie.
 30. Find the shortest unique prefix for every word in a list.
-

◆ Binary / Bitwise Trie Problems

31. Insert integers into a binary Trie.
 32. Find maximum XOR of a given number with Trie elements.
 33. Find minimum XOR of a given number with Trie elements.
 34. Count numbers in Trie less than a given number.
 35. Count numbers in Trie greater than a given number.
 36. Count numbers with a given prefix in a binary Trie.
 37. Find the K-th smallest/largest number in a binary Trie.
 38. Delete a number from a binary Trie.
 39. Merge two binary Tries.
 40. Find the number of pairs with XOR in a given range using Trie.
-

◆ Miscellaneous / Logical Trie Problems

41. Check if a word can be formed by concatenating other words in Trie.
42. Find all words with at most K edits (Levenshtein distance) using Trie.
43. Find all words that start and end with a given character.
44. Implement a dictionary with auto-suggestion and frequency ranking.
45. Store and search words with multiple languages (Unicode Trie).
46. Find duplicate words in a large dataset using Trie.
47. Solve “word break” problem using Trie.
48. Solve “concatenated words” problem using Trie.

49. Count words that are prefixes of other words.
50. Optimize memory usage of Trie using compressed Trie / Radix Tree.

I can also make **20 coding problems from these Trie statements with input/output examples in C++ or Java.**

Do you want me to do that next?

You said:

graph

ChatGPT said:

Here are **50 one-line problem statements on Graphs** — covering **basic, traversal, shortest paths, MST, and advanced topics** 📝

◆ Basic Graph Operations

1. Represent a graph using adjacency matrix.
2. Represent a graph using adjacency list.
3. Add a vertex to a graph.
4. Add an edge to a graph.
5. Remove a vertex from a graph.
6. Remove an edge from a graph.
7. Check if a graph is connected.
8. Count the number of vertices in a graph.
9. Count the number of edges in a graph.
10. Display the adjacency list or matrix of a graph.

◆ Graph Traversal

11. Perform Depth-First Search (DFS) of a graph.
 12. Perform Breadth-First Search (BFS) of a graph.
 13. Detect a cycle in an undirected graph using DFS.
 14. Detect a cycle in a directed graph using DFS.
 15. Check if a graph is bipartite using BFS or DFS.
 16. Count connected components in an undirected graph.
 17. Find all paths between two vertices.
 18. Print a topological sort of a DAG.
 19. Find strongly connected components using Kosaraju's algorithm.
 20. Find strongly connected components using Tarjan's algorithm.
-

◆ Shortest Path Problems

21. Find shortest path in an unweighted graph using BFS.
 22. Find shortest path in a weighted graph using Dijkstra's algorithm.
 23. Find shortest paths from a single source using Bellman-Ford algorithm.
 24. Detect negative weight cycles using Bellman-Ford algorithm.
 25. Find all pairs shortest path using Floyd-Warshall algorithm.
 26. Find shortest path in a DAG using topological sort.
 27. Find the longest path in a DAG.
 28. Find shortest path in a grid with obstacles.
 29. Find minimum steps to reach from source to destination in a maze.
 30. Find shortest distance between two nodes in an unweighted graph.
-

◆ Minimum Spanning Tree (MST)

31. Find MST using Kruskal's algorithm.
 32. Find MST using Prim's algorithm.
 33. Count total weight of MST.
 34. Find all edges in MST.
 35. Check if an edge is part of MST.
 36. Find second-best MST.
 37. Find MST for a graph represented using adjacency list.
 38. Find MST for a graph represented using adjacency matrix.
 39. Find maximum edge in MST.
 40. Replace an edge in MST and update MST weight.
-

◆ Advanced Graph Problems

41. Detect cycle in a directed graph using Kahn's algorithm.
 42. Find articulation points in a graph.
 43. Find bridges (critical edges) in a graph.
 44. Count number of cycles in an undirected graph.
 45. Solve shortest path with exactly K edges.
 46. Solve Traveling Salesman Problem (TSP) using DP + Bitmask.
 47. Find all Hamiltonian paths in a graph.
 48. Find all Hamiltonian cycles in a graph.
 49. Find the number of islands in a 2D grid using graph traversal.
 50. Find the shortest path visiting all nodes (graph with all-pairs distances).
-

I can also create **20 coding problems from these Graph statements with input/output examples** in **C++ or Java** if you want.

Do you want me to do that next?

You said:

Unionfind

ChatGPT said:

Here are **50 one-line problem statements on Union-Find / Disjoint Set Union (DSU)** — covering **basic operations, connectivity, and advanced applications** 📄

◆ Basic Union-Find Operations

1. Implement a basic Union-Find data structure.
 2. Implement path compression in Union-Find.
 3. Implement union by rank in Union-Find.
 4. Implement union by size in Union-Find.
 5. Find the representative (parent) of an element.
 6. Check if two elements belong to the same set.
 7. Count the total number of disjoint sets.
 8. Merge two sets.
 9. Initialize Union-Find for N elements.
 10. Display all sets in the Union-Find structure.
-

◆ Connectivity / Graph Problems

11. Check if a graph is fully connected using Union-Find.
 12. Count connected components in an undirected graph.
 13. Detect cycles in an undirected graph.
 14. Detect cycles in a graph with weighted edges using Union-Find.
 15. Implement Kruskal's algorithm to find MST.
 16. Find redundant connection in a graph.
 17. Find the size of the largest connected component.
 18. Find the number of connected components after each union.
 19. Count the number of islands in a 2D grid using Union-Find.
 20. Check if adding an edge creates a cycle in a graph.
-

◆ Dynamic Connectivity / Online Queries

21. Perform online connectivity queries (are u and v connected?).
22. Perform online union operations dynamically.
23. Implement rollback union-find for undo operations.
24. Implement persistent union-find for historical queries.
25. Find the number of distinct sets after a series of unions.
26. Merge sets with a given element into one set.

27. Find the largest set dynamically after union operations.
 28. Maintain the sum of elements in each disjoint set.
 29. Maintain the maximum element in each disjoint set.
 30. Maintain the minimum element in each disjoint set.
-

◆ Advanced Union-Find Applications

31. Solve the accounts merge problem using Union-Find.
 32. Solve the friend circles / number of provinces problem.
 33. Solve the connectivity problem in a social network.
 34. Solve the redundant connections II problem (graph with directed edges).
 35. Count the number of edges required to make a graph fully connected.
 36. Find connected components in a dynamic graph.
 37. Find connected components in a bipartite graph.
 38. Solve dynamic connectivity in a tree.
 39. Find the earliest time when all nodes are connected (union of edges over time).
 40. Solve the smallest equivalent string problem using Union-Find.
-

◆ Logical / Miscellaneous Applications

41. Detect whether a given set of equations is satisfiable using Union-Find.
 42. Solve the latest day to cross a river problem using Union-Find.
 43. Solve the maximum area of island problem using Union-Find.
 44. Count connected groups in a grid with obstacles.
 45. Solve the regions cut by slashes problem using Union-Find.
 46. Find cycles in a directed graph with DSU + backtracking.
 47. Solve the connecting cities problem using Union-Find.
 48. Implement a weighted Union-Find with sum queries.
 49. Implement a Union-Find with parity check for bipartiteness.
 50. Solve the number of islands II problem (adding land incrementally in a grid).
-

I can also create **20 coding problems from these Union-Find statements with sample input/output in C++ or Java** if you want.

Do you want me to do that next?

You said:

Number theory and engineering mathematical

ChatGPT said:

Here's a comprehensive list of **50 one-line problem statements in Number Theory and Engineering Mathematics**, covering **basic to advanced topics** used in competitive programming, engineering, and applied math:

◆ Basic Number Theory

1. Find the GCD of two numbers using Euclidean algorithm.
 2. Find the LCM of two numbers.
 3. Check if a number is prime.
 4. Generate all prime numbers up to N using Sieve of Eratosthenes.
 5. Find all divisors of a number.
 6. Count the number of divisors of a number.
 7. Find the sum of divisors of a number.
 8. Find the product of divisors of a number.
 9. Check if two numbers are coprime.
 10. Find the Euler's Totient function $\phi(n)$.
-

◆ Modular Arithmetic

11. Compute $(a^b) \% m$ using modular exponentiation.
 12. Compute modular inverse of a number.
 13. Solve linear congruences of the form $ax \equiv b \pmod{m}$.
 14. Compute $(a * b) \% m$ safely for large integers.
 15. Find factorial modulo m .
 16. Compute binomial coefficients modulo m .
 17. Find modular division using Fermat's little theorem.
 18. Solve Chinese Remainder Theorem problems.
 19. Find $(a^b + c^d) \% m$.
 20. Check if a number is a quadratic residue modulo p .
-

◆ Advanced Number Theory

21. Factorize a number into prime factors.
22. Count the number of prime factors of a number.
23. Find the largest prime factor of a number.
24. Compute GCD and LCM of multiple numbers.
25. Solve linear Diophantine equations.
26. Solve Pell's equation for small integers.
27. Find sum of totients of all numbers up to N .
28. Compute Möbius function $\mu(n)$.
29. Compute Legendre symbol (a/p) .
30. Check if a number is a perfect square using modular arithmetic.

◆ Sequences & Series

31. Find the Nth Fibonacci number using matrix exponentiation.
32. Find sum of first N natural numbers.
33. Find sum of squares of first N numbers.
34. Find sum of cubes of first N numbers.
35. Generate arithmetic progression and its sum.
36. Generate geometric progression and its sum.
37. Find sum of reciprocals of first N numbers (harmonic sum).
38. Solve linear recurrence relations using matrix exponentiation.
39. Find Nth term of a custom recurrence sequence modulo m.
40. Check if a number belongs to Fibonacci sequence.

◆ Combinatorics / Probability

41. Compute factorial of N.
42. Compute nCr and nPr .
43. Find number of ways to partition N objects.
44. Solve problems using Inclusion-Exclusion Principle.
45. Compute Catalan numbers.
46. Compute Stirling numbers of the second kind.
47. Count number of derangements for N objects.
48. Count permutations with constraints.
49. Solve problems involving binomial identities.
50. Compute probability of events using combinatorics.

If you want, I can create **20 fully worked-out coding problems from these Number Theory and Engineering Math topics with input/output examples in C++ or Java**, which are perfect for competitive programming and engineering exams.

Do you want me to do that next?

You said:

dynamic programing

ChatGPT said:

Here's a list of **50 one-line problem statements on Dynamic Programming (DP)** — covering **basic, intermediate, and advanced topics** used in competitive programming and engineering problems:

◆ Basic DP Problems

1. Compute the Nth Fibonacci number using DP.
 2. Find the number of ways to climb N stairs (1 or 2 steps).
 3. Compute factorial of N using DP.
 4. Find minimum cost to reach the end of a grid.
 5. Find maximum sum path in a 1D array.
 6. Compute the Nth Tribonacci number using DP.
 7. Count the number of binary strings of length N without consecutive 1s.
 8. Count ways to partition N using integers 1..N.
 9. Solve the coin change problem (number of ways).
 10. Solve the coin change problem (minimum coins).
-

◆ Intermediate DP Problems

11. Longest Common Subsequence (LCS) of two strings.
 12. Longest Common Substring of two strings.
 13. Longest Increasing Subsequence (LIS) in an array.
 14. Maximum sum increasing subsequence.
 15. Minimum number of deletions to make a string palindrome.
 16. Minimum number of insertions to make a string palindrome.
 17. Edit Distance between two strings.
 18. Matrix Chain Multiplication problem.
 19. Egg Dropping Puzzle.
 20. Rod Cutting problem.
-

◆ Advanced 1D/2D DP Problems

21. Maximum sum subarray (Kadane's DP variant).
 22. Maximum sum rectangle in a 2D matrix.
 23. Minimum path sum in a 2D grid with obstacles.
 24. Count number of unique paths in a grid.
 25. Count number of unique paths with obstacles.
 26. Minimum cost path in a grid with cell costs.
 27. Decode ways of a numeric string (like "123" → "ABC").
 28. Count ways to tile a $2 \times N$ board with 2×1 tiles.
 29. Count ways to tile a $3 \times N$ board.
 30. Maximum profit in a job scheduling problem.
-

◆ Knapsack & Subset Problems

31. 0/1 Knapsack problem.
32. Fractional Knapsack (DP variant, for practice).

33. Subset sum problem.
 34. Count subsets with given sum.
 35. Minimum subset sum difference.
 36. Partition array into two subsets with equal sum.
 37. Target sum problem (assign + or – signs).
 38. Coin change with limited coins.
 39. Number of ways to reach a target using given numbers.
 40. Maximum product subset problem.
-

◆ String / Sequence DP Problems

41. Count palindromic subsequences in a string.
 42. Longest palindromic subsequence.
 43. Longest palindromic substring.
 44. Count distinct subsequences of a string.
 45. Maximum length of repeating subsequence.
 46. Word break problem (dictionary based).
 47. Count number of distinct ways to form a word.
 48. Maximum ASCII sum of subsequence.
 49. Edit distance with operation costs.
 50. Wildcard pattern matching with ? and *.
-

I can also create **20 coding problems from these DP statements with sample input/output in C++ or Java**, covering **classic beginner to advanced DP problems**.

Do you want me to do that next?

You said:

greddy

ChatGPT said:

Here's a list of **50 one-line problem statements on Greedy Algorithms** — covering **classic, intermediate, and advanced greedy problems**:

◆ Basic Greedy Problems

1. Activity selection problem (maximum non-overlapping activities).
2. Fractional Knapsack problem.
3. Minimum number of coins to make change.
4. Minimum number of platforms required for a railway station.
5. Job sequencing with deadlines (maximum profit).
6. Select intervals to cover a line segment with minimum intervals.

7. Task scheduling to minimize total completion time.
 8. Maximum number of meetings in a room.
 9. Minimum number of arrows to burst balloons.
 10. Select maximum number of non-overlapping intervals.
-

◆ Intermediate Greedy Problems

11. Huffman coding (build optimal prefix codes).
 12. Assign tasks to workers to minimize maximum workload.
 13. Minimize total waiting time for jobs in a queue.
 14. Select numbers to maximize sum with constraints.
 15. Minimize sum of absolute differences between arrays.
 16. Minimum cost to connect ropes.
 17. Minimize total cost of cutting a board.
 18. Maximum sum of non-adjacent numbers in an array.
 19. Maximum profit from stock buy/sell with one transaction.
 20. Maximum profit from stock buy/sell with multiple transactions.
-

◆ Graph / Connectivity Greedy Problems

21. Prim's algorithm for Minimum Spanning Tree (MST).
 22. Kruskal's algorithm for Minimum Spanning Tree (MST).
 23. Find shortest path in a weighted graph using greedy relaxation.
 24. Minimum cost to connect all cities (like MST).
 25. Find maximum spanning tree in a graph.
 26. Minimum cost to supply water to all villages.
 27. Minimize penalty in connecting network nodes.
 28. Find the cheapest flights with at most K stops (variant greedy).
 29. Greedy solution for Dijkstra's algorithm.
 30. Assign resources optimally to minimize total cost.
-

◆ Interval / Scheduling / Selection Problems

31. Maximum number of events that can be attended.
32. Minimum number of intervals to remove to eliminate overlaps.
33. Minimize total penalty for late jobs.
34. Maximum length chain of pairs.
35. Minimum number of arrows to burst balloons in 1D.
36. Select tasks with earliest finish time.
37. Assign meeting rooms to minimize room usage.
38. Maximum length of non-overlapping activities.
39. Minimum number of refills to reach destination.
40. Cover points with minimum number of intervals.

◆ Miscellaneous / Logical Greedy Problems

41. Minimize the cost to hire K workers.
42. Maximum number of gifts to carry within weight limit.
43. Minimum number of stickers to spell a word.
44. Maximum profit by choosing non-conflicting projects.
45. Arrange books on shelves to minimize height.
46. Minimize maximum difference of pairings.
47. Distribute candies to children to satisfy conditions.
48. Maximize sum by selecting numbers from multiple arrays.
49. Greedy algorithm for Egyptian fractions.
50. Minimize average waiting time for customers in a queue.

I can also create **20 coding problems from these Greedy statements with sample input/output in C++ or Java** if you want.

Do you want me to do that next?

You said:

recursions

ChatGPT said:

Here's a list of **50 one-line problem statements on Recursion** — covering **basic, intermediate, and advanced recursive problems**:

◆ Basic Recursion Problems

1. Compute factorial of a number recursively.
2. Compute the Nth Fibonacci number recursively.
3. Compute the sum of first N natural numbers.
4. Reverse a string using recursion.
5. Check if a string is a palindrome recursively.
6. Compute the power of a number (a^b) recursively.
7. Print numbers from 1 to N recursively.
8. Print numbers from N to 1 recursively.
9. Sum of digits of a number recursively.
10. Count the number of digits in a number recursively.

◆ Intermediate Recursion Problems

11. Find the GCD of two numbers recursively.
 12. Find the LCM of two numbers recursively.
 13. Solve Tower of Hanoi problem.
 14. Generate all subsets (power set) of a set recursively.
 15. Generate all permutations of a string recursively.
 16. Solve N-Queens problem recursively.
 17. Solve Rat in a Maze problem recursively.
 18. Count paths in a grid from top-left to bottom-right.
 19. Find the maximum element in an array recursively.
 20. Find the minimum element in an array recursively.
-

◆ Advanced Recursion / Backtracking Problems

21. Solve Sudoku recursively.
 22. Solve Knight's Tour problem recursively.
 23. Generate balanced parentheses for N pairs.
 24. Word search in a grid recursively.
 25. Count the number of ways to climb stairs with 1, 2, or 3 steps.
 26. Solve subset sum problem recursively.
 27. Solve combination sum problem recursively.
 28. Print all paths in a binary tree recursively.
 29. Find the height of a binary tree recursively.
 30. Count leaf nodes in a binary tree recursively.
-

◆ Recursion on Arrays / Strings

31. Check if an array is sorted recursively.
 32. Reverse an array recursively.
 33. Find the first occurrence of an element recursively.
 34. Find the last occurrence of an element recursively.
 35. Count occurrences of an element in an array recursively.
 36. Check if a string is a subsequence of another recursively.
 37. Remove consecutive duplicates in a string recursively.
 38. Replace all occurrences of a character in a string recursively.
 39. Print all subsequences of a string recursively.
 40. Generate all palindromic partitions of a string recursively.
-

◆ Miscellaneous / Logical Recursion Problems

41. Compute the sum of elements in an array recursively.
42. Find the product of elements in an array recursively.
43. Solve the maximum sum path in a triangle recursively.
44. Count binary strings of length N without consecutive 1s.

45. Solve the “coin change” problem recursively.
 46. Solve the “egg dropping” problem recursively.
 47. Generate Gray codes of N bits recursively.
 48. Print numbers in a given range with recursion.
 49. Solve the “Josephus problem” recursively.
 50. Compute the sum of digits raised to powers recursively.
-

I can also create **20 coding problems from these Recursion statements with sample input/output in C++ or Java**, covering **classic beginner to advanced recursion problems**.

Do you want me to do that next?

You said:

brut force

ChatGPT said:

Here’s a list of **50 one-line problem statements on Brute Force / Exhaustive Search** — covering **arrays, strings, combinatorics, and other logical problems**:

◆ **Basic Brute Force Problems**

1. Find the maximum element in an array using brute force.
 2. Find the minimum element in an array using brute force.
 3. Compute the sum of all elements in an array.
 4. Count occurrences of an element in an array.
 5. Search for an element in an array linearly.
 6. Reverse an array using brute force swapping.
 7. Check if a string is a palindrome by comparing characters.
 8. Count vowels in a string using brute force.
 9. Find all divisors of a number.
 10. Check if a number is prime using trial division.
-

◆ **Intermediate Brute Force Problems**

11. Generate all subsets (power set) of a set.
12. Generate all permutations of a string or array.
13. Generate all combinations of size K from N elements.
14. Solve 0/1 Knapsack by trying all subsets.
15. Solve coin change by trying all combinations.
16. Solve subset sum problem by checking all subsets.
17. Find all pairs in an array that sum to a target.
18. Find all triplets in an array that sum to a target.

19. Find all quadruplets in an array that sum to a target.
 20. Count all increasing subsequences in an array.
-

◆ String / Pattern Brute Force Problems

21. Check if two strings are anagrams by comparing all permutations.
 22. Count all substrings of a string.
 23. Find the longest palindromic substring by checking all substrings.
 24. Count all palindromic substrings in a string.
 25. Generate all valid parentheses sequences of N pairs.
 26. Find all occurrences of a pattern in a string by checking each position.
 27. Generate all rotations of a string.
 28. Solve the word break problem by checking all splits.
 29. Generate all subsequences of a string.
 30. Count distinct substrings in a string.
-

◆ Array / Matrix Brute Force Problems

31. Find maximum sum subarray by checking all subarrays.
 32. Find maximum product subarray by checking all subarrays.
 33. Find all pairs with minimum difference.
 34. Find all triplets with minimum difference.
 35. Count number of inversions in an array using brute force.
 36. Find maximum rectangle of 1s in a binary matrix by checking all submatrices.
 37. Find all paths from top-left to bottom-right in a grid using brute force.
 38. Solve N-Queens by trying all placements.
 39. Solve Knight's Tour by trying all moves recursively.
 40. Solve Sudoku by trying all possible numbers.
-

◆ Logical / Miscellaneous Brute Force Problems

41. Solve the Traveling Salesman Problem by trying all permutations.
 42. Solve the Assignment Problem by trying all assignments.
 43. Find all subsets with sum divisible by K.
 44. Find all partitions of a number.
 45. Generate all sequences of length N from a set of numbers.
 46. Solve the Josephus problem by simulating each step.
 47. Solve the "minimum coins" problem by checking all combinations.
 48. Solve subset product problems by checking all subsets.
 49. Find all magic squares of size $N \times N$.
 50. Solve combinatorial optimization problems by checking all possibilities.
-

I can also create **20 coding problems from these Brute Force statements with sample input/output in C++ or Java**, covering classic beginner to advanced brute force problems.

Do you want me to do that next?

You said:

string

ChatGPT said:

Here's a list of **50 one-line problem statements on Strings** — covering **basic, intermediate, and advanced string problems**:

◆ Basic String Problems

1. Reverse a string.
 2. Check if a string is a palindrome.
 3. Count vowels and consonants in a string.
 4. Count occurrences of each character in a string.
 5. Convert a string to uppercase.
 6. Convert a string to lowercase.
 7. Remove duplicate characters from a string.
 8. Find the length of the longest substring without repeating characters.
 9. Check if two strings are anagrams.
 10. Concatenate two strings.
-

◆ Intermediate String Problems

11. Find the first non-repeating character in a string.
 12. Find the last non-repeating character in a string.
 13. Count the number of substrings in a string.
 14. Count the number of palindromic substrings.
 15. Find the longest palindromic substring.
 16. Check if a string is a rotation of another string.
 17. Implement string compression (like "aaabb → a3b2").
 18. Find all permutations of a string.
 19. Check if a string is a subsequence of another string.
 20. Remove all spaces from a string.
-

◆ Advanced String Problems

21. Longest common prefix of a set of strings.

22. Longest common substring between two strings.
 23. Longest common subsequence between two strings.
 24. Minimum number of insertions to make a string palindrome.
 25. Minimum number of deletions to make a string palindrome.
 26. Edit distance between two strings.
 27. Count the number of distinct substrings in a string.
 28. Pattern matching using naive method.
 29. Pattern matching using KMP algorithm.
 30. Pattern matching using Rabin-Karp algorithm.
-

◆ String Transformations / Manipulations

31. Reverse words in a sentence.
 32. Convert a string to an integer (atoi).
 33. Convert an integer to a string (itoa).
 34. Remove all vowels from a string.
 35. Replace all occurrences of a character in a string.
 36. Remove consecutive duplicates in a string.
 37. Check if two strings are anagrams ignoring spaces and cases.
 38. Count occurrences of a substring in a string.
 39. Check if a string is numeric.
 40. Check if a string is alphabetic.
-

◆ Logical / Miscellaneous String Problems

41. Group words that are anagrams.
 42. Check if a string can be rearranged to form a palindrome.
 43. Find all palindromic partitions of a string.
 44. Find the smallest window in a string containing all characters of another string.
 45. Implement run-length encoding of a string.
 46. Generate all possible substrings of a string.
 47. Check if a string matches a pattern with wildcards ? and *.
 48. Count the number of distinct words in a paragraph.
 49. Convert a string from camelCase to snake_case.
 50. Convert a string from snake_case to camelCase.
-

I can also create **20 coding problems from these String statements with sample input/output in C++ or Java**, covering classic beginner to advanced string problems.

Do you want me to do that next?

You said:

advance on string

ChatGPT said:

Here's a list of **50 advanced string problem statements** — covering **pattern matching, suffix structures, string algorithms, and combinatorial string problems**:

◆ Pattern Matching / Searching

1. Implement KMP (Knuth-Morris-Pratt) algorithm for pattern search.
 2. Implement Rabin-Karp algorithm for multiple pattern search.
 3. Implement Z-algorithm for pattern matching.
 4. Find all occurrences of a substring using naive method.
 5. Count the number of occurrences of a pattern in a text.
 6. Find the longest prefix which is also a suffix (LPS array).
 7. Find all anagrams of a pattern in a string.
 8. Find the smallest window in a string containing all characters of another string.
 9. Check if a string contains a permutation of another string.
 10. Pattern matching with wildcards ? and *.
-

◆ Palindrome / Subsequence / Substring Problems

11. Find the longest palindromic substring using DP.
 12. Find the longest palindromic substring using Manacher's algorithm.
 13. Count all palindromic substrings in a string.
 14. Find the longest palindromic subsequence.
 15. Count distinct palindromic subsequences.
 16. Minimum insertions to make a string palindrome.
 17. Minimum deletions to make a string palindrome.
 18. Find all palindromic partitions of a string.
 19. Check if a string can be segmented into palindromes.
 20. Longest repeating substring in a string.
-

◆ Suffix Structures / Trie / Automaton

21. Build a suffix array for a string.
22. Build the LCP (Longest Common Prefix) array for a string.
23. Find the lexicographically smallest rotation of a string.
24. Find the number of distinct substrings using suffix array.
25. Build a suffix tree for a string.
26. Find longest repeated substring using suffix tree.
27. Implement a trie for a set of strings.
28. Count words in a trie with a given prefix.

29. Find the longest common substring between two strings using suffix array/tree.
 30. Solve the “number of occurrences of a substring” problem using suffix automaton.
-

◆ String Compression / Transformation / Encoding

31. Run-length encoding of a string.
 32. Burrows-Wheeler Transform of a string.
 33. Huffman encoding for string compression.
 34. Convert a string to its lexicographically next permutation.
 35. Convert a string to its lexicographically previous permutation.
 36. Encode a string using base62 encoding.
 37. Decode a string from base62 encoding.
 38. String encryption using Caesar cipher.
 39. String decryption using Caesar cipher.
 40. Check if a string can be transformed into another with minimum swaps.
-

◆ Logical / Combinatorial / Advanced Problems

41. Count the number of distinct subsequences of a string.
42. Count the number of subsequences equal to a target string.
43. Count subsequences with sum of characters' values divisible by K.
44. Check if a string matches a regular expression (without libraries).
45. Solve the word break problem using DP and Trie.
46. Generate all distinct subsequences of a string.
47. Count all substrings that occur at least twice.
48. Find the shortest unique substring for each string in a set.
49. Solve the “smallest rotation to match another string” problem.
50. Compute the edit distance between two strings with costs for insertion, deletion, and substitution.