Here are **30 one-line problem statements** ideal for practicing **Binary Search on Answer** (also called **parametric search**) — all structured for **Codeforces-style** training:

✓ Maximize X that satisfies a condition:

1. **Max length of a rope to cut into at least K pieces.**
2. **Max distance between cows placed in stalls.**
3. **Max side of square that fits inside a grid with at most K cuts.**
4. **Max height of building such that total construction cost ≤ budget.**
5. **Max number of candies each child can get, given total candies and N children.**
6. **Max weight that can be lifted with a given number of moves.**
7. **Max height of water in a tank that doesn't overflow after T seconds.**
8. **Max speed at which a vehicle can go without missing any traffic signal.**
9. **Max possible time interval between two train arrivals with safety constraints.**
10. **Max power level of all machines without exceeding the total power limit.**

✓ Minimize X that satisfies a condition:

11. **Min time to paint all boards using K painters.**
12. **Min max load per worker to finish N tasks.**
13. **Min length of subarray with sum ≥ K.**
14. **Min number of jumps to reach end if max jump length is limited.**
15. **Min number of pages each student must read in book allocation.**
16. **Min largest distance after placing M routers in N houses.**
17. **Min radius of lamps to light the whole street.**
18. **Min maximum elevation change to build a road.**
19. **Min maximum packet size to send data within D days.**
20. **Min max latency to connect all computers in a network.**

✓ Variation: Search over real numbers (double binary search)

21. **Max volume of water that can flow per second through pipes.**
22. **Min possible temperature to cool all rooms within T minutes.**
23. **Max probability of passing all exams under given strategy.**
24. **Min fuel needed to reach destination with refuel constraints.**
25. **Max average score after choosing K exams from N.**
26. **Min average delivery time over N packages.**
27. **Max height water can rise without breaching any barrier.**
28. **Min rope thickness to safely support swinging object.**
29. **Max area of rectangle inside matrix with height constraints.**
30. **Min cost per unit area for land purchase with area ≥ A.**

Would you like me to add links to similar real Codeforces problems for practice for each of these?

You said:

Greedy + Invariants Prove choices are optimal using logic, not trial-error.

ChatGPT said:

Here are **30 one-line problem statements** designed for **Greedy + Invariants**—where you **prove the greedy choice is optimal using logical justification, not brute-force**. These are perfect for Codeforces-style practice:

✅ Classic Greedy with Invariant Logic

1. **Select max number of non-overlapping intervals.**
2. **Give minimum number of coins to make value V (with specific denominations).**
3. **Assign jobs to workers to minimize total completion time.**
4. **Choose max number of activities one can attend with no time overlap.**
5. **Distribute candies so that difference between richest and poorest is minimized.**
6. **Place guards in minimum positions to cover a hallway with blind spots.**
7. **Remove characters from a string to make it lexicographically smallest.**
8. **Schedule lectures to minimize number of lecture halls used.**
9. **Build the longest tower using boxes of decreasing weight and increasing strength.**
10. **Form largest number by removing K digits from a string of digits.**

✅ Sorting + Greedy Decisions

11. **Maximize number of people who can cross a bridge before it collapses.**
12. **Maximize sum of absolute differences between adjacent array elements (after rearranging).**
13. **Rearrange soldiers to avoid collisions with minimal moves.**
14. **Split array into k parts such that the max of mins is maximized.**
15. **Choose k projects with minimum total deadline penalty.**
16. **Cover all intervals using minimum number of points.**
17. **Select meetings to attend to earn maximum total value.**
18. **Pick tasks with deadlines to maximize total value earned.**
19. **Find minimum number of arrows to burst all balloons represented as intervals.**
20. **Choose maximum number of tasks from two lists with time constraint.**

✅ Invariants with Greedy Logic (non-obvious proofs)

21. **Perform operations on array to make all elements equal with min moves.**
22. **Add least number of edges to make graph connected using greedy merging.**

23. **Minimize total wait time in a queue by sorting task durations.**
24. **Transform one string to another with adjacent swaps (min swaps needed).**
25. **Choose friends to invite based on trust levels and invitations given.**
26. **Buy tickets over days to minimize total cost with expiry logic.**
27. **Keep reducing array sum with operations until target is reached.**
28. **Choose subset of elements such that their sum is divisible by K.**
29. **Distribute N gifts to maximize happiness, where happiness = min(friendliness, gifts).**
30. **Minimize number of platforms needed at railway station for train timings.**

Would you like **real Codeforces problem links** for these types of Greedy + Invariants questions next?

You said:

Two Pointers / Sliding Window Used with arrays and strings under constraints.

ChatGPT said:

Here are **30 one-line problem statements** ideal for practicing **Two Pointers and Sliding Window** techniques — especially in **arrays and strings under constraints**, perfect for Codeforces-style problems:

✓ Two Pointers — Fixed/Variable Window Size

1. **Find if any two elements in sorted array sum to X.**
2. **Count number of subarrays with sum less than K.**
3. **Find longest subarray with at most K distinct elements.**
4. **Find smallest window in a string containing all characters of another string.**
5. **Count pairs with absolute difference less than K.**
6. **Find longest substring with equal number of 0s and 1s.**
7. **Find max length of subarray with sum ≤ K.**
8. **Given sorted array, find all unique triplets that sum to 0.**
9. **Count subarrays with exactly K odd numbers.**
10. **Find first window of size K with max sum.**

✓ Sliding Window — Dynamic Window Expansion/Shrinking

11. **Longest substring with all unique characters.**
12. **Longest substring with at most K repeating characters.**
13. **Minimum size subarray with sum ≥ S.**
14. **Max sum of all contiguous subarrays of size K.**
15. **Count all substrings where vowels > consonants.**
16. **Check if permutation of string B exists in string A.**

17. **Find max number of fruits collected from trees (2 types allowed).**
18. **Count subarrays where product is less than K.**
19. **Longest substring where character frequency does not exceed K.**
20. **Minimum swaps to group all 1s together.**

✓ Variations on Movement with Constraints

21. **Find number of ways to split array into 3 parts with equal sum.**
22. **Count pairs in sorted array with sum in range [L, R].**
23. **Longest segment where total cost of converting A to B ≤ maxCost.**
24. **Maximize number of students who can get scholarship based on sliding window of marks.**
25. **Count substrings with at most K distinct characters.**
26. **Find max length of balanced string with equal 0s and 1s.**
27. **Count binary substrings with equal consecutive 0s and 1s.**
28. **Maximize number of ones after flipping one subarray.**
29. **Longest continuous increasing subsequence using window.**
30. **Check if there is a subarray of length at least K with average ≥ target.**

Would you like me to categorize these by difficulty or link them to actual Codeforces problems for hands-on practice?

You said:

Prefix Sums + Hashing Subarray/substring matching, frequency checking

ChatGPT said:

Here are **30 one-line problem statements** ideal for mastering **Prefix Sums + Hashing** — perfect for **subarray/substring matching, frequency counting, and value lookups** under Codeforces-style constraints:

✓ Prefix Sum + Hashing — Subarrays and Substrings

1. **Count number of subarrays with sum equal to K.**
2. **Find length of longest subarray with sum 0.**
3. **Count number of subarrays with equal number of 0s and 1s.**
4. **Check if a string contains an anagram of another string.**
5. **Find longest substring where frequency of all characters is even.**
6. **Find count of subarrays where sum is divisible by K.**
7. **Check if subarrays with same sum exist using prefix sums.**
8. **Find length of longest prefix which is also suffix but not equal to string.**
9. **Count palindromic substrings using prefix hashes.**
10. **Check if two substrings are equal using rolling hash.**

✅ Subarray Property Checking with Hash Maps

11. **Check if any subarray sums to zero.**
12. **Count subarrays where sum of elements is a multiple of M.**
13. **Longest substring with same character counts in two strings.**
14. **Number of subarrays with at most K distinct integers.**
15. **Count subarrays where XOR is equal to K.**
16. **Find all substrings with matching hash values.**
17. **Count subarrays where number of even and odd numbers is same.**
18. **Find number of substrings where frequency of each character is ≤ K.**
19. **Minimum number of changes to make substring an anagram.**
20. **Find number of index pairs where prefix sum difference is constant.**

✅ String Hashing + Frequency Hashing

21. **Detect duplicate substrings of length L using rolling hash.**
22. **Find longest repeating substring using binary search + hash.**
23. **Check if string is rotation of another using hash matching.**
24. **Find longest substring with equal character frequencies.**
25. **Number of substrings with exactly K distinct characters.**
26. **Count pairs of substrings which are anagrams.**
27. **Find all palindromic substrings of a given string.**
28. **Count substrings where character count is a multiple of 3.**
29. **Check if two strings are equivalent after K rotations.**
30. **Find total frequency of substrings of length L using hash table.**

Would you like a **topic-wise breakdown with Codeforces problem links** for hands-on practice next?

You said:

Bitmasking Subset sums, XOR conditions, on/off states

ChatGPT said:

Here are **30 one-line problem statements** ideal for **Bitmasking** practice — focused on **subset sums, XOR conditions, on/off states**, etc., in the style of Codeforces problems:

✅ Subset Generation & Sum/XOR Conditions

1. **Count number of subsets whose sum is exactly K.**
2. **Find subset with maximum XOR value in array.**

3. **Check if any subset XOR equals a given number.**
4. **Count number of subsets with even XOR.**
5. **Count number of subsets with sum divisible by M.**
6. **Generate all subsets of an array using bitmask.**
7. **Find smallest subset of elements whose XOR is non-zero.**
8. **Check if subset exists with XOR equal to target T.**
9. **Find maximum subset XOR using trie or DP + bitmask.**
10. **Subset of strings where no two share a common letter (mask conflict).**

✓ On/Off States and Combinatorics

11. **Turn bulbs on/off to achieve target configuration with minimal switches.**
12. **Simulate N devices where each has K toggles using bitmask state.**
13. **Count valid ways to switch lights where no two adjacent are on.**
14. **Minimum number of steps to go from initial bitmask to target bitmask.**
15. **Switch toggle problem: given operations, reach all-on state.**
16. **Count number of bitmask states reachable under given operations.**
17. **Minimum toggles to turn off all lights in a grid.**
18. **Simulate N people wearing different items (bitmask wardrobe combinations).**
19. **Represent permissions (read/write/execute) using 3-bit mask per user.**
20. **Detect overlapping permissions using AND of bitmasks.**

✓ Dynamic Programming + Bitmask

21. **DP over subsets to minimize cost of assigning tasks to workers.**
22. **Solve traveling salesman problem using bitmask DP.**
23. **Count Hamiltonian paths in graph using bitmask DP.**
24. **Partition elements into two groups with minimum XOR difference.**
25. **Assign N items to K persons using bitmask subset coverage.**
26. **Maximize team performance using skills as bitmasks.**
27. **Minimize number of groups such that no pair shares common bit.**
28. **Bitmask DP to count ways to tile a board.**
29. **Choose maximal set of non-conflicting items using bitmasks.**
30. **Find minimum number of steps to cover all required features using given tools (set cover via bitmask).**

Would you like to pair these with **real Codeforces/AtCoder problems** or get **C++/Java starter templates** for solving these types?

You said:

Basic DP (1D/2D, with states) LIS, LCS, knapsack, segment sums with conditions

ChatGPT said:

Here are **30 one-line problem statements** ideal for practicing **Basic Dynamic Programming (DP)** — focusing on **1D/2D DP**, classical problems like **LIS, LCS, Knapsack**, and **segment sums with conditions**. These are tailored for **Codeforces-style constraints**:

✅ 1D/2D DP — Longest Increasing Subsequence (LIS) and Variants

1. **Find length of the Longest Increasing Subsequence in array.**
2. **Count number of Longest Increasing Subsequences.**
3. **Find LIS after removing at most one element.**
4. **Find minimum number of deletions to make array strictly increasing.**
5. **Find Longest Bitonic Subsequence.**
6. **Find length of Longest Decreasing Subsequence.**
7. **Partition array into minimal number of LIS subsequences.**
8. **Find the Longest Common Increasing Subsequence of two arrays.**
9. **Find minimum moves to make array non-decreasing using LIS logic.**
10. **Find maximum sum of increasing subsequence.**

✅ 2D DP — LCS, Edit Distance, Matching

11. **Find length of the Longest Common Subsequence between two strings.**
12. **Find minimum number of insertions and deletions to convert A to B.**
13. **Find minimum edit distance between two strings.**
14. **Find length of longest palindromic subsequence.**
15. **Count number of distinct subsequences of A equal to B.**
16. **Check if one string is a subsequence of another using DP.**
17. **Compute shortest common supersequence of two strings.**
18. **Find number of ways to convert string A to B using insert/delete/replace.**
19. **Longest alternating subsequence using DP.**
20. **Find longest repeating subsequence in a string.**

✅ 1D/2D DP — Knapsack and Subset Sums

21. **0/1 Knapsack: Max value with weight ≤ W.**
22. **Subset sum: Check if sum S is possible using array elements.**
23. **Count number of subsets that sum to target S.**
24. **Min number of coins needed to make amount X.**
25. **Unbounded Knapsack: Repeat items to maximize value under weight W.**
26. **Partition array into 2 subsets with equal sum.**
27. **Min subset difference of two subsets.**
28. **Find number of ways to assign + and - signs to reach target sum.**
29. **Bounded Knapsack with limited quantity of each item.**

30. **Count number of combinations to reach sum K using array elements (coin change).**

Would you like links to **actual Codeforces problems** for each of these, or would you like the **standard DP table setup/states** for these classic problems?

You said:

Graphs (DFS, BFS, Toposort) Grids, dependencies, components, simple trees.

ChatGPT said:

Here are **30 one-line problem statements** ideal for mastering **Graph Algorithms** like **DFS, BFS, Topological Sort** — including use cases in **grids, components, and trees**. These are framed for **Codeforces-style contests**:

✅ DFS / BFS — Basic Graph Traversal

1. **Count number of connected components in an undirected graph.**
2. **Find if a cycle exists in an undirected graph using DFS.**
3. **Find shortest path from node A to B in unweighted graph using BFS.**
4. **Check if a graph is bipartite using BFS/DFS coloring.**
5. **Count number of islands in a 2D grid (connected 1s).**
6. **Find largest island (connected area of 1s) in grid.**
7. **Flood fill a region in grid with a new color.**
8. **Check if there's a path from top-left to bottom-right in a binary matrix.**
9. **Perform DFS on tree to calculate subtree sizes.**
10. **Find farthest node from a given source in tree using BFS.**

✅ Topological Sort — Dependencies / DAGs

11. **Determine valid order of courses given prerequisite relations.**
12. **Check if there's a cycle in a directed graph (DAG check).**
13. **Find lexicographically smallest topological sort.**
14. **Topologically sort tasks with dependency constraints.**
15. **Count number of valid topological orderings.**
16. **Calculate earliest start times for tasks with dependencies.**
17. **Find number of ways to finish all tasks given a dependency graph.**
18. **Check if two topological sorts are possible.**
19. **Minimum time to complete all jobs with parallel processing.**
20. **Detect cycle in directed graph using DFS and recursion stack.**

✓ Connected Components, Grids, Simple Trees

21. **Label and count connected components in a 2D matrix.**
22. **Check if a graph is a tree (connected + N-1 edges).**
23. **Find diameter (longest path) of a tree using two BFS.**
24. **Find lowest common ancestor (LCA) of two nodes in tree.**
25. **Determine articulation points (cut vertices) in undirected graph.**
26. **Find bridges (cut edges) in a graph.**
27. **Perform DFS traversal to flatten tree (Euler tour).**
28. **Sum of distances from each node to all others in a tree.**
29. **Construct adjacency list from edge list and run DFS.**
30. **Implement graph traversal to detect disconnected subgraphs.**

Would you like C++/Java templates for DFS/BFS/TopoSort or want **Codeforces/AtCoder problems matched to each type** for practice?

You said:

Combinatorics + Math Binomial coefficient, mod inverse, number construction

ChatGPT said:

Here are **30 one-line problem statements** ideal for practicing **Combinatorics + Math** in a competitive programming setting (e.g., Codeforces). These cover **binomial coefficients, modular arithmetic (modular inverse), and number construction under constraints**:

✓ Binomial Coefficients (nCr, nPr) + Counting

1. **Find C(n, r) % mod for large n and r.**
2. **Count number of ways to choose K elements from N distinct elements.**
3. **Count number of binary strings of length N with exactly K ones.**
4. **Count ways to divide N candies among K kids (non-negative).**
5. **Count number of lattice paths from (0,0) to (n,m).**
6. **Count number of unique BSTs that can be formed with N nodes.**
7. **Compute n! / (k1! * k2! * ... * kn!) % mod (multinomial coefficient).**
8. **Number of ways to arrange N people in a circle.**
9. **Count number of derangements (no element in original position).**
10. **Count non-negative integer solutions to x1 + x2 + ... + xk = N.**

✓ Modular Arithmetic + Modular Inverse

11. **Compute modular inverse of a number modulo prime.**
12. **Solve (a / b) % mod using modular inverse.**
13. **Compute C(n, r) % mod where mod is not prime (Lucas Theorem).**

14. **Find number of sequences with sum S and elements ≤ X, modulo M.**
15. **Find modular exponentiation of a^b % mod for large a and b.**
16. **Find total number of divisors of N! modulo mod.**
17. **Sum of first N factorials modulo mod.**
18. **Evaluate sum(1^k + 2^k + ... + n^k) % mod.**
19. **Count total permutations of string with duplicate characters modulo mod.**
20. **Compute modular division in combinatorics: C(n, k) / C(m, r) % mod.**

✅ Number Construction + Permutations

21. **Count number of N-digit numbers divisible by K.**
22. **Count N-digit numbers that have sum of digits equal to S.**
23. **Form the largest/smallest number using given digits and constraints.**
24. **Find number of palindromic numbers of N digits.**
25. **Construct lexicographically smallest string from character frequencies.**
26. **How many numbers < X can be formed using only digits {A, B, C}.**
27. **Count number of ways to rearrange digits to form number divisible by M.**
28. **Count permutations of string such that no two vowels are adjacent.**
29. **Number of ways to assign + and - signs to reach target sum (subset sign DP).**
30. **Count ways to construct numbers with given digit sum divisible by K.**

Would you like a breakdown of **formulas (like Fermat’s Little Theorem, Lucas, Modular Inverse)** or links to **Codeforces/AtCoder problems** for these types next?