

Here are **30 one-line problem statements** categorized by **Advanced Dynamic Programming** techniques:

✓ Divide & Conquer DP

1. Minimize cost to partition an array into k segments with defined cost per segment.
 2. Given n matrices, find the minimum multiplication cost using optimal order (Matrix Chain Multiplication).
 3. Find optimal binary search tree construction with minimal expected search cost.
 4. Partition a sequence into k parts minimizing sum of costs of each part.
 5. Segment n elements such that total penalty from segment transitions is minimized.
-

✓ DP with Bitmasks

6. Count the number of ways to assign n jobs to n workers with constraints.
 7. Find minimum cost to visit all cities exactly once (Traveling Salesman Problem).
 8. Compute number of Hamiltonian paths in a directed graph.
 9. Maximize the score of selecting a subset of items with non-overlapping properties.
 10. Count the number of valid ways to tile an $n \times m$ grid using dominos.
-

✓ Tree DP (including rerooting)

11. For each node in a tree, compute the sum of distances to all other nodes.
 12. Find size of largest independent set in a tree.
 13. Count number of subtrees with even number of nodes.
 14. Compute max path sum (diameter) for a tree.
 15. Reroot tree at every node and calculate answer for each new root (e.g., sum of distances).
-

✓ Slope Trick (Convex DP Optimization)

16. Minimize cost of aligning n positions with minimum movement cost.
 17. Dynamic programming with $dp[i] = \min(dp[j] + \text{cost}(j, i))$ where cost is convex.
 18. Optimal checkpoint placement on a line to minimize total deviation from k centers.
 19. Scheduling jobs to minimize weighted tardiness penalty (Convex cost).
 20. Place facilities on a line with penalty increasing quadratically from optimal location.
-

✓ Knuth Optimization

21. Optimal matrix parenthesization with cost function satisfying quadrangle inequality.
 22. Partition a string into palindromic substrings with minimum cuts.
 23. Merge stones with minimal total cost when merging only k at a time.
 24. Minimize total cost to concatenate n strings where merging cost is proportional to length.
 25. Find minimal edit cost when operations follow associative/triangle inequality rules.
-

✓ Monotonic Queue Optimization

26. Maximize score by jumping k steps forward at each move with cost per jump.
 27. Find minimum cost to paint fence sections with restrictions on adjacent colors.
 28. Largest sum of subarray with at most k elements using sliding window DP.
 29. Maximize profit from stock trading with cooldown using windowed profit tracking.
 30. Minimize cost of climbing stairs with jump limits using deque-based DP.
-

Let me know if you want full problems with constraints, explanations, or code.

You said:

. Advanced Graph Algorithms • Strongly Connected Components (Kosaraju/Tarjan) • Biconnected Components • 2-SAT Problem • Bridges and Articulation Points • Minimum Cost Flow • Flow with Lower Bounds • Maximum Matching in Bipartite Graph (Kuhn's, Hopcroft-Karp) • Min-Cost Max Flow

ChatGPT said:

Here are **30 one-line problem statements** categorized by **Advanced Graph Algorithms** topics:

✓ Strongly Connected Components (Kosaraju / Tarjan)

1. Find the number of strongly connected components in a directed graph.
 2. Identify which nodes belong to the same SCC in a graph.
 3. Shrink a graph to its component DAG using SCC decomposition.
 4. Find the minimal number of edges to add to make a directed graph strongly connected.
 5. Determine if each node in a graph can reach every other node (strong connectivity check).
-

✓ Biconnected Components

6. Find all biconnected components in an undirected graph.
 7. Count the number of articulation points in a graph.
 8. Identify critical edges whose removal increases number of connected components.
 9. Check if a graph is biconnected (no articulation point).
 10. List all edge-biconnected components of a graph.
-

✓ 2-SAT Problem

11. Check if a set of 2-variable boolean constraints can be satisfied.
 12. Find one satisfying assignment for a given 2-SAT formula.
 13. Minimize number of true variables satisfying all 2-variable clauses.
 14. Check satisfiability of constraints like "if A is true, then B must be false".
 15. Determine feasibility of scheduling two types of events with exclusive overlaps.
-

✓ Bridges and Articulation Points

16. Find all bridge edges in an undirected graph.
 17. Count number of connected components formed after removing one articulation point.
 18. Identify articulation points in a graph that disconnect it on removal.
 19. List bridge edges whose removal increases number of connected components.
 20. Build a bridge tree from a given undirected graph.
-

✓ Minimum Cost Flow

21. Given supply and demand at nodes, compute minimum cost to send all required flow.
 22. Transport goods across a weighted network minimizing shipping cost.
 23. Compute cost of fulfilling all orders in a supply chain with capacity and price constraints.
 24. Route data packets from sources to sinks minimizing total transmission cost.
 25. Solve project task dependencies with resource flow minimizing total cost.
-

✓ Flow with Lower Bounds

26. Given a network with lower and upper bounds on flow, check if feasible flow exists.
27. Find feasible flow in a network respecting both lower and upper capacity bounds.
28. Route flow through a network where each edge has a minimum flow requirement.
29. Model flow constraints with mandatory delivery quotas per edge.
30. Balance a supply chain network where each route must carry at least a baseline flow.

✔ Maximum Matching in Bipartite Graph (Kuhn's / Hopcroft-Karp)

31. Find maximum number of job assignments for workers and jobs.
32. Match students to projects such that no two students share the same project.
33. Determine if a perfect matching exists in a bipartite graph.
34. Calculate maximum matching between doctors and patients based on availability.
35. Maximize number of pairings between applicants and apartments with preferences.

✔ Min-Cost Max Flow

36. Assign workers to tasks minimizing total cost while maximizing number of assignments.
37. Match buyers to sellers maximizing trades and minimizing cost of transactions.
38. Optimize resource allocation with both quantity and cost constraints.
39. Schedule k projects with minimum cost and maximum profit flow.
40. Find max flow of goods through a network with the least total delivery cost.

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

You said:

Data Structures • Mergeable Heaps (e.g., Leftist Tree, Skew Heap) • Persistent Segment Trees • Treap / Splay Tree (Implicit Treap for sequence manipulation) • Heavy-Light Decomposition (HLD) • Mo's Algorithm (with updates) • Segment Tree Beats

ChatGPT said:

Here are **30 one-line problem statements** categorized by advanced **Data Structures** topics:

✔ Mergeable Heaps (Leftist Tree, Skew Heap)

1. Perform a sequence of merge and extract-min operations on disjoint heaps efficiently.
2. Merge two priority queues in logarithmic time and extract the minimum.
3. Maintain a set of disjoint heaps and support deletion of any element.
4. Simulate a multiway tournament with players stored in mergeable heaps.
5. Support insertion and deletion from multiple heaps with quick merges.

✔ Persistent Segment Trees

6. Support range sum queries on historical versions of an array.
 7. Rollback or jump to any previous version of the array and perform queries.
 8. Count how many numbers $\leq x$ in a subarray $[l, r]$ using persistence.
 9. Find the k -th smallest number in a range $[l, r]$ over multiple versions.
 10. Track versioned data structure for undo/redo operations and interval queries.
-

✓ Treap / Splay Tree (Implicit Treap for sequence manipulation)

11. Perform insert, erase, split, merge on an array-like sequence in logarithmic time.
 12. Reverse a subarray in a dynamic array using implicit treap.
 13. Move a subsegment $[l, r]$ to front/middle/end using sequence manipulation.
 14. Maintain a dynamic string with cut-paste-reverse operations.
 15. Find and delete the k -th element in a mutable sequence.
-

✓ Heavy-Light Decomposition (HLD)

16. Answer path queries (e.g., sum/max) between any two nodes in a tree efficiently.
 17. Update values on paths or subtrees in a tree with logarithmic complexity.
 18. Find lowest common ancestor (LCA) of any two nodes using HLD.
 19. Process a tree where you need to handle queries like "maximum value on a path".
 20. Support range updates and queries on rooted tree paths.
-

✓ Mo's Algorithm (with updates)

21. Count the number of distinct elements in range $[l, r]$ for multiple queries.
 22. Answer frequency of most occurring element in a range $[l, r]$.
 23. Support adding/removing updates and range queries with offline processing.
 24. Answer queries like "how many values appear at least k times in range $[l, r]$?"
 25. Process queries with range add and range count using Mo's with updates.
-

✓ Segment Tree Beats

26. Support range minimum/maximum updates and queries where lazy propagation fails.
 27. On a range, apply "set all values $\leq x$ to x " efficiently.
 28. Support "minimize" or "maximize" a range with conditional updates.
 29. Find the sum, min, and count of min in a range with conditional modifications.
 30. Solve range-chmin, range-chmax, and range-add with complex segment logic.
-

Let me know if you'd like to expand any of these into full problems or want implementations.

You said:

Matrix Exponentiation (for Linear Recurrences) • Fast Walsh–Hadamard Transform (FWHT) • Fast Fourier Transform (FFT) • Number Theoretic Transform (NTT) • Diophantine Equations • Gaussian Elimination (Modulo and Floating Point) • Combinatorics with Generating Functions • Mobius Inversion • Burnside's Lemma and Polya Enumeration

ChatGPT said:

Here are **30 one-line problem statements** categorized under advanced **Math & Algebraic Techniques** in competitive programming:

✓ **Matrix Exponentiation (for Linear Recurrences)**

1. Find the N-th Fibonacci number modulo $1e9+7$.
2. Compute N-th term of a linear recurrence with given initial terms and coefficients.
3. Count number of ways to tile a $2 \times N$ board using dominoes and trominoes.
4. Calculate population growth in N years modeled by a linear recurrence.
5. Simulate linear recurrence for stocks/investments for very large N.

✓ **Fast Walsh–Hadamard Transform (FWHT)**

6. Compute XOR convolution of two binary arrays efficiently.
7. Count number of subsets whose XOR is less than a given value.
8. Solve subset convolution problems where operation is XOR instead of sum.
9. Perform subset convolution in exponential time with bitmask-based transforms.
10. Count the number of ways to select disjoint sets with a specific XOR value.

✓ **Fast Fourier Transform (FFT)**

11. Multiply two large polynomials or integers efficiently.
12. Find the number of pairs of elements from two arrays that sum up to a given value.
13. Count number of subarrays with a fixed sum using convolution.
14. Perform pattern matching using polynomial multiplication.
15. Multiply two numbers each with up to $1e6$ digits using FFT.

✓ **Number Theoretic Transform (NTT)**

16. Multiply large polynomials under a modulo (like 998244353) using NTT.

17. Compute convolution of integer sequences modulo a prime using NTT.
 18. Solve polynomial equations in modular arithmetic using NTT.
 19. Find the number of ways to make a sum using coins (modulo prime) using NTT.
 20. Efficiently multiply polynomials for combinatorics problems under mod.
-

✓ Diophantine Equations

21. Solve for integer solutions of $ax + by = c$.
 22. Find smallest positive solution of $ax \equiv b \pmod{m}$.
 23. Count number of solutions of $x^2 + y^2 = z^2$ for given z .
 24. Solve $ax + by + cz = d$ in integers using parameterization.
 25. Determine if it's possible to express a number as a linear combination of two given integers.
-

✓ Gaussian Elimination (Modulo and Floating Point)

26. Solve system of linear equations under mod p .
 27. Find rank of a matrix under modulo arithmetic.
 28. Compute inverse of a matrix modulo prime.
 29. Determine consistency and number of solutions of a system of equations.
 30. Perform Gaussian elimination to solve circuit flow or current balance equations.
-

✓ Combinatorics with Generating Functions

31. Count number of integer partitions of n into odd parts.
 32. Use exponential generating functions to count labeled trees.
 33. Solve linear recurrence using generating function roots.
 34. Count compositions of n with given part sizes using polynomial multiplication.
 35. Use generating functions to solve constrained dice roll sums.
-

✓ Möbius Inversion

36. Count the number of coprime pairs in a given range using Möbius function.
 37. Compute sum over divisors efficiently using Möbius inversion.
 38. Invert a Dirichlet convolution for number-theoretic functions.
 39. Count number of integers whose GCD with n is exactly k .
 40. Use Möbius inversion to compute primitive necklaces of given length.
-

✓ Burnside's Lemma and Polya Enumeration

41. Count distinct colorings of a bracelet considering rotation symmetry.
 42. Count number of non-equivalent ways to color faces of a cube.
 43. Compute number of unique tilings of a board under rotation.
 44. Count labeled graphs modulo automorphism using Burnside's lemma.
 45. Use Polya enumeration to count distinct chemical compound structures.
-

Let me know if you want **full problems**, **code**, or **visualizations** for any topic.

You said:

. String Algorithms • Suffix Automaton • Z-function, KMP, Aho-Corasick • Manacher's Algorithm • Lyndon Factorization • Palindromic Tree / Eertree • Hashing with Collision Control • Trie with Bitmask Compression

ChatGPT said:

Here are **30 one-line problem statements** categorized under advanced **String Algorithms**:

✓ Suffix Automaton

1. Count the number of distinct substrings of a given string.
 2. Find the length of the longest common substring between two strings.
 3. Count occurrences of each substring in the input string.
 4. Given a string and its suffix automaton, answer if a pattern exists as a substring.
 5. Find the lexicographically smallest substring of length k .
-

✓ Z-function, KMP, Aho-Corasick

6. Find all occurrences of a pattern in a text using Z-algorithm.
 7. Preprocess a pattern for fast pattern matching using KMP.
 8. Given multiple patterns, find all their occurrences in a large text (Aho-Corasick).
 9. Find the smallest period of a string using Z or KMP.
 10. Detect if any of the banned words (from a list) appear in a message.
-

✓ Manacher's Algorithm

11. Count total number of palindromic substrings in a string in linear time.
12. Find the longest palindromic substring in a given string.
13. Check if a string can be made palindrome by changing at most one character.

14. Identify the maximum even/odd-length palindrome centered at every index.
 15. Find the minimal characters to add to make the string a palindrome.
-

✓ Lyndon Factorization

16. Break a string into lexicographically smallest Lyndon substrings.
 17. Generate minimal string rotation using Lyndon decomposition.
 18. Sort all rotations of a string using Lyndon factorization.
 19. Compress a string using canonical Lyndon factor sequences.
 20. Find the number of distinct Lyndon factors in a string.
-

✓ Palindromic Tree / Eertree

21. Count how many distinct palindromic substrings exist in a string.
 22. Insert characters one by one and maintain list of all palindromic suffixes.
 23. Track longest palindromic suffix after each insertion in a stream.
 24. For each prefix, find how many new palindromes were formed.
 25. Calculate palindromic factorizations of a string.
-

✓ Hashing with Collision Control

26. Use double hashing to avoid false positives in substring comparison.
 27. Compare two substrings for equality in constant time using polynomial hashing.
 28. Detect repeated substrings of length k using rolling hash.
 29. Find longest prefix which is also a suffix (not equal to full string).
 30. Count how many times a pattern appears using hash and sliding window.
-

✓ Trie with Bitmask Compression

31. Store dictionary of words with compressed bitmask nodes to save memory.
 32. Insert and search words using compressed trie.
 33. Count how many strings share a given prefix using bitmask trie.
 34. Find the most frequent prefix in a list of strings using trie + bitmasks.
 35. Auto-suggest completions based on input prefix using compressed trie structure.
-

Let me know if you'd like code templates, visualizations, or deep-dive tutorials on any of these.

You said:

Geometry (Advanced) • Convex Hull Trick (CHT) • Rotating Calipers • Line Sweep for Intersections • Segment Intersection using Order Set • Point in Polygon / Convex Hull Inclusion • Circle Intersection and Geometry Tricks

ChatGPT said:

Here are **30 one-line problem statements** categorized under **Advanced Geometry Algorithms**:

✓ Convex Hull Trick (CHT)

1. Maintain a set of linear functions and answer min/max at point x .
 2. Optimize dynamic programming recurrence: $dp[i] = \min(dp[j] + a[i] * b[j])$.
 3. Compute the minimum cost path through a series of linear decisions.
 4. Find minimal cost with linear penalty functions over sorted inputs.
 5. Solve slope-based DP problems efficiently using dynamic or static CHT.
-

✓ Rotating Calipers

6. Find the diameter (farthest pair of points) of a convex polygon.
 7. Compute the width of a convex polygon (minimum bounding strip).
 8. Given a convex polygon, find the pair of sides with the greatest distance.
 9. Find the maximum area of a triangle that can be inscribed in a convex polygon.
 10. Use rotating calipers to find maximum rectangle inside a convex polygon.
-

✓ Line Sweep for Intersections

11. Count number of intersections among a set of horizontal and vertical segments.
 12. Determine if any pair of segments in the plane intersect.
 13. Find the leftmost intersection point among a set of line segments.
 14. Report all intersection points between a set of segments (plane sweep).
 15. Given rectangles, count how many pairs overlap.
-

✓ Segment Intersection using Order Set

16. Detect if any two segments intersect using a balanced BST (sweep line).
17. Maintain active segments while sweeping to detect event ordering.
18. Report the first intersection point in left-to-right sweep of segments.
19. Find all events (insert/remove/intersect) while sweeping through segments.
20. Handle dynamic segment insertion and intersection query with ordered sets.

✔ Point in Polygon / Convex Hull Inclusion

21. Determine whether a given point lies inside, outside, or on the boundary of a polygon.
22. Check if a point lies inside a convex polygon in $O(\log N)$.
23. For each query point, check if it lies within the convex hull of given points.
24. Classify all points from a list as inside/outside/on polygon.
25. Compute convex hull and check if multiple points lie within it.

✔ Circle Intersection and Geometry Tricks

26. Count number of intersection points between two circles.
27. Compute common chord (line) of two intersecting circles.
28. Find area of intersection between two circles.
29. Determine number of distinct regions formed by multiple intersecting circles.
30. Given multiple circles, find all pairwise intersecting circle pairs.

Let me know if you want **code**, **formulas**, or **visualizations** for any problem type.

You said:

Bitmask & Brute Force • SOS DP (Sum over Subsets) • Meet in the Middle • Inclusion-Exclusion over subsets • Bitwise DP • Digit DP (Harder variations)

ChatGPT said:

Here are **30 one-line problem statements** categorized under **Bitmask & Brute Force** techniques:

✔ SOS DP (Sum Over Subsets DP)

1. Given $f[mask]$, compute $g[mask] = \sum(f[submask])$ for all $submask \subseteq mask$.
2. Count how many subsets of numbers XOR to a given value.
3. For each subset of size k , find the total sum of elements from all smaller subsets.
4. Preprocess frequency of elements to quickly answer XOR subset queries.
5. Find the number of supersets of every subset satisfying a given property.

✔ Meet in the Middle

6. Count the number of subsets whose sum equals a target value when $n > 30$.

7. Find closest possible sum to `target` using subset sums of two halves.
 8. Given weights, divide them into two sets with minimal difference (balanced partition).
 9. Count number of triplets summing to target using first-half + second-half pairs.
 10. Solve bounded knapsack for large constraints using half enumeration.
-

✓ Inclusion-Exclusion over Subsets

11. Count number of integers $\leq N$ divisible by at least one of a given set of primes.
 12. Count number of ways to place objects where some combinations are forbidden.
 13. Count subsets that satisfy at least one condition from a list.
 14. Compute exact number of strings that violate none of the forbidden patterns.
 15. Count permutations avoiding certain positions using principle of inclusion-exclusion.
-

✓ Bitwise DP

16. Count number of ways to place tiles in a grid using mask-based DP.
 17. Solve TSP using `dp[mask][last]` for up to $n = 20$ cities.
 18. Count ways to partition set into groups with no conflicting pairs (bitmask compatibility).
 19. For a set of strings, find maximum chain such that each word is a subset of the next.
 20. DP on subsets to build optimal combination of items with bitwise constraints.
-

✓ Digit DP (Harder Variations)

21. Count numbers $\leq N$ having exactly K non-zero digits.
 22. Count numbers between L and R such that sum of digits is divisible by D .
 23. Count numbers in range $[L, R]$ with no adjacent equal digits.
 24. Count numbers $\leq N$ where digits are strictly increasing.
 25. Count numbers $\leq N$ with at most two distinct digits.
-

✓ Mixed / Bonus Problems

26. Count valid configurations in a puzzle using full bitmask brute force ($n \leq 20$).
27. Find number of set bits in binary representations from 1 to N .
28. Count palindromes in a number range using digit DP.
29. Partition n elements into k groups such that each satisfies a bitmask property.
30. Count the number of valid ways to fill a $2 \times n$ board using dominoes with transitions via bitmask DP.