Here are **30 one-line problem statements** covering advanced Dynamic Programming (DP) techniques:

□ 1–5: DP with Segment Trees / BIT

1. Find the maximum sum increasing subsequence with fast range max queries.
2. Count the number of increasing subsequences ending at each index.
3. Range DP: Compute the optimal cost of partitioning an array into K segments.
4. For each prefix, find LIS length using Segment Tree in O(N log N).
5. Count number of elements less than current on the left using BIT.

⚔ 6–10: Divide and Conquer DP

6. Minimize total cost of assigning tasks to workers with convex cost.
7. Compute minimum edit distance with cost matrices using D&C optimization.
8. Optimize merging stones with divide-and-conquer DP approach.
9. Find minimal total penalty for dividing string into palindromes.
10. DP[i][j]: cost of placing j partitions in i elements, optimize using divide & conquer.

□ 11–15: Knuth Optimization

11. Matrix chain multiplication with optimal parenthesis order using Knuth's rule.
12. Minimize cost of merging files with prefix sums and Knuth optimization.
13. Optimal BST construction using frequency of searches and Knuth's trick.
14. Palindromic partitioning cost minimized using Knuth optimization.
15. Minimize score for combining elements with associative cost function.

📉 16–20: Convex Hull Trick (CHT)

16. Minimize total cost for linear cost functions over dynamic ranges.
17. Compute DP[i] = min(DP[j] + A[i]*B[j]) using convex hull.
18. Solve a problem where choosing optimal j gives linear transition for DP[i].
19. Find min total fuel to reach i stations with line transitions.
20. Optimize quadratic DP relations with line hull trick.

□ 21–25: Bitmask DP (Trees / Graphs)

21. Count number of Hamiltonian paths in a graph using bitmasking.
22. Held-Karp algorithm for exact TSP on N ≤ 20 using Bitmask DP.

23. Find minimum cost to color tree nodes under bitmask constraints.
24. DP[state]: max value collected in a game with visited-mask and current node.
25. Count ways to form target string from dictionary using mask & trie-based DP.

□ 26–30: Slope Trick (Hard)

26. Minimize sum of absolute differences from median using slope trick.
27. Maintain minimum value of a convex piecewise linear function with updates.
28. Minimize convex cost with non-decreasing constraint using slope optimization.
29. Compute optimal jumps with increasing penalty using slope technique.
30. Solve dynamic programming over hills with decreasing slopes and penalties.

Let me know if you'd like these grouped by difficulty, platform links, or detailed explanations!

You said:

✅ Advanced Graph Algorithms • Dynamic Connectivity (Link-Cut Trees, Euler Tour Trees) • 2-SAT • Minimum Cost Flow (Cycle Canceling, Primal-Dual) • Persistent Disjoint Set Union (DSU) • Heavy-Light Decomposition (HLD) • Tree Isomorphism • Centroid Decomposition

ChatGPT said:

Here are **30 one-line problem statements** based on **Advanced Graph Algorithms**:

🔗 1–5: Dynamic Connectivity (Link-Cut Trees / Euler Tour Trees)

1. Support online edge insertions/deletions in a forest and answer connectivity queries.
2. Maintain dynamic minimum spanning forest with fast updates and queries.
3. Answer ancestor-descendant queries after dynamic subtree rerooting.
4. For a dynamic forest, track the size of connected components after updates.
5. Determine if two nodes are connected in a forest with edge deletions and additions.

□ 6–10: 2-SAT

6. Determine if a boolean formula in CNF with 2 literals per clause is satisfiable.
7. Find one valid assignment for a satisfiable 2-SAT formula.
8. Determine if a set of implication rules has a contradiction.
9. Compute the strongly connected components of implication graph in 2-SAT.
10. Identify forced assignments in 2-SAT (which variables must be true/false).

💰 11–15: Minimum Cost Flow (Cycle Canceling / Primal-Dual)

11. Find the cheapest way to send K units of flow from source to sink.
12. Determine the minimum cost to assign N workers to N jobs with constraints.
13. Solve transshipment problem with flow costs between multiple sources/sinks.
14. Compute minimum cost of transporting goods with capacity and cost per edge.
15. Use successive shortest paths (Primal-Dual) for cost-constrained flow delivery.

🕰 16–20: Persistent DSU

16. After each union, answer connectivity queries for any previous state (rollback).
17. Maintain historical versions of DSU for offline query resolution.
18. Count number of connected components at any point in past timeline.
19. Answer whether u and v were in same component at time t.
20. Track size of component of any node at historical time using persistent DSU.

□ 21–25: Heavy-Light Decomposition (HLD)

21. Support path queries and point updates in a tree using HLD and segment trees.
22. Find sum of weights along path from u to v in a tree.
23. Change weights on edges and query max weight on a path efficiently.
24. Find LCA of any two nodes using HLD decomposition.
25. Support subtree updates and path max/min queries in trees using HLD.

□ 26–27: Tree Isomorphism

26. Check if two rooted trees are isomorphic using canonical form hashing.
27. Count the number of non-isomorphic rooted trees of size n.

□ 28–30: Centroid Decomposition

28. Count number of pairs of nodes in a tree whose distance is ≤ K.
29. Solve tree path queries involving constraints by dividing tree into centroids.
30. Find total number of paths in a tree whose sum is divisible by M using centroid decomposition.

Let me know if you want these with difficulty levels, resources, or implementation guidance!

You said:

Geometry (Hardcore) • Rotating Calipers • Minkowski Sum • Convex Hull in 3D • Dynamic Convex Hull • Line Intersection Counting (Sweep Line) • Geometry with Integer Coordinates (careful precision)

ChatGPT said:

Here are **30 one-line problem statements** covering **Hardcore Geometry** topics:

🔄 1–5: Rotating Calipers

1. Find the diameter (farthest pair of points) of a convex polygon.
2. Compute the minimum width of a convex polygon using calipers.
3. Find the maximum area triangle that can be inscribed in a convex polygon.
4. Calculate the max distance between any two edges of convex polygons.
5. Determine antipodal pairs in convex polygons using rotating calipers.

➕ 6–10: Minkowski Sum

6. Compute the Minkowski sum of two convex polygons.
7. Check if origin lies within the Minkowski sum of A and −B (collision check).
8. Use Minkowski sum for motion planning in a 2D polygonal environment.
9. Count lattice points inside the Minkowski sum of two convex integer polygons.
10. Given two polygons, determine if their Minkowski sum overlaps a third.

□ 11–15: Convex Hull in 3D

11. Construct the convex hull of N points in 3D space.
12. Count the number of faces, edges, and vertices in a 3D convex polytope.
13. Check if a point lies inside the 3D convex hull of given points.
14. Compute surface area of convex polyhedron formed by given points.
15. Handle degenerate coplanar point sets when building 3D convex hulls.

🔧 16–20: Dynamic Convex Hull

16. Maintain upper/lower convex hull with insertions and deletions of points.
17. Answer max dot product queries with vectors in dynamic convex hull.
18. Given online point additions, maintain convex hull perimeter.
19. Find convex hull of points added dynamically and queried on-the-fly.
20. Efficiently support min/max slope queries after adding lines dynamically.

✂ 21–25: Line Intersection Counting (Sweep Line)

21. Count number of intersections between N line segments in O(N log N).
22. Report all intersections of N segments using plane sweep.
23. Given vertical and horizontal lines, count total intersection points.
24. Find all overlapping segments between two sets using sweep line.
25. For a set of segments, find if any two intersect using segment tree.

📐 26–30: Integer Geometry (Precision-Critical)

26. Determine if three integer-coordinate points form a right triangle.
27. Compute the area of a polygon with integer vertices using shoelace formula.
28. Find intersection point of two lines with exact rational coordinates.
29. Count number of lattice points on the boundary and inside a polygon.
30. Check if a point with integer coordinates lies strictly inside an integer polygon.

Let me know if you want implementation hints, test cases, or references for any of these!

You said:

Combinatorics & Counting • Polya Enumeration Theorem • Burnside’s Lemma • Stirling Numbers (1st and 2nd kind) • Inclusion-Exclusion on sets and masks • Generating Functions (Exponential and Ordinary) • FFT-based polynomial multiplication

ChatGPT said:

Here are **30 one-line problem statements** covering **Combinatorics & Counting** (Advanced Topics):

🔁 1–5: Polya Enumeration Theorem

1. Count distinct colorings of the vertices of a square under rotation and reflection.
2. Given N beads on a necklace, count the unique colorings with K colors.
3. Count unique grid fillings under row/column permutations using Polya’s theorem.
4. How many ways can you paint the faces of a cube with C colors, considering symmetry?
5. Count the number of non-isomorphic graphs with labeled edges and vertex symmetries.

□ 6–10: Burnside’s Lemma

6. Find the number of distinct bracelets (circular sequences) with color constraints.
7. Count ways to tile a circular table with colored plates, modulo symmetry.
8. Count number of inequivalent arrangements of numbers on a circular board.
9. Determine the number of non-equivalent permutations under group actions.
10. Use Burnside’s lemma to count non-equivalent colorings under rotation and flips.

🔢 11–15: Stirling Numbers of the First Kind

11. Count permutations of N elements with exactly K disjoint cycles.
12. Find the number of ways to write a permutation with a given number of cycles.
13. Compute expected number of cycles in a random permutation of N elements.
14. Count number of derangements with fixed number of cycles.
15. Evaluate sum over Stirling numbers of the first kind for identity checks.

📚 16–20: Stirling Numbers of the Second Kind

16. Count ways to partition N elements into K non-empty unlabeled subsets.
17. Find the number of ways to divide students into K groups for a project.
18. Compute Bell numbers using sum of Stirling numbers of the second kind.
19. Solve recurrence involving S(n, k) for recursive partition models.
20. Count total set partitions of size N (Bell numbers via Stirling numbers).

🔍 21–25: Inclusion-Exclusion (Sets / Masks)

21. Count integers ≤ N divisible by at least one of a given list of primes.
22. Count permutations with at least one fixed point using inclusion-exclusion.
23. Given bitmasks for properties, count subsets satisfying at least one condition.
24. Find number of subsets where none of the elements violate a banned pair rule.
25. Count the number of strings avoiding given forbidden substrings using IEP.

🔢 26–30: Generating Functions & FFT

26. Compute number of ways to make sum N using given set of coin denominations.
27. Find coefficient of xNx^NxN in the product of generating functions of sequences.
28. Multiply two polynomials efficiently modulo large prime using FFT.
29. Given N items and constraints, count valid combinations using exponential generating functions.
30. Solve a partition problem via polynomial exponentiation using FFT-based convolution.

Let me know if you'd like visual aids, code templates, or links to platforms like Codeforces, AtCoder, or USACO problems for any of these!

You said:

Data Structures (Expert Level) • Persistent Segment Tree / Trie • Mergeable Heap • Link-Cut Tree (Splay Tree or Euler Tour Tree) • Segment Tree Beats • Mo’s Algorithm on Tree / Queries on Tree with Updates • Wavelet Trees

ChatGPT said:

Here are **30 one-line problem statements** for **Expert-Level Data Structures**:

🌲 1–5: Persistent Segment Tree / Persistent Trie

1. Maintain multiple versions of an array with point updates and range sum queries.
2. Given k versions of an array, find the number of elements less than x in version v.
3. Find the k-th smallest number in a subarray using persistent segment trees.
4. Implement undo/rollback for previous modifications using persistent structures.
5. Store versions of a Trie to answer prefix count queries in versioned dictionaries.

🌀 6–10: Mergeable Heap

6. Maintain multiple disjoint heaps that support merge, insert, and extract-min.
7. Use leftist or binomial heaps to simulate a battle between armies with strength.
8. Merge k priority queues in logarithmic time to answer minimum queries.
9. Perform Union-Find with mergeable heaps to find min/max in each component.
10. Support meldable heaps to track median in dynamic sets.

🔗 11–15: Link-Cut Tree (Splay Tree / Euler Tour Tree)

11. Dynamically add and remove edges in a forest and answer path queries.
12. Maintain the diameter of a tree under link and cut operations.
13. Answer subtree sum/path min queries with dynamic connectivity.
14. For a tree with dynamic edge weights, answer path max queries using LCT.
15. Implement Euler Tour Tree to check if two nodes are connected after updates.

⚡ 16–20: Segment Tree Beats

16. Support range min update (set min on segment) and range sum queries.
17. Maintain segment tree for range chmin, chmax, add, and sum in one structure.
18. Perform conditional range updates with constraints on upper/lower bounds.
19. Solve RMQ with conditional replace for segments in O(log² N).
20. Count how many values fall within a certain range and update only if they exceed a threshold.

🌳 21–25: Mo’s Algorithm on Tree / Tree Queries with Updates

21. Answer queries on number of distinct colors in a subtree.
22. Count number of nodes with a certain value on the path between two nodes.
23. Solve subtree XOR queries with node updates using Euler Tour + Mo’s algorithm.
24. Answer kth color/node on path u–v with updates using Mo on Tree.
25. Handle subtree queries and point updates with MO's offline batching on trees.

📶 26–30: Wavelet Trees

26. Find the k-th smallest element in a subarray in O(log N) using Wavelet Tree.
27. Count number of elements in a subarray within a given value range.
28. Compute frequency of an element in a given range.
29. For a range [L, R], return the number of elements less than or equal to K.
30. Support dynamic insertion and deletion in sequences while answering order-statistics queries.

Let me know if you'd like categorized links to practice problems (Codeforces, AtCoder, etc.), implementation guides, or visual aids for these complex structures.

You said:

Math (Advanced Number Theory) • Discrete Logarithm (Baby Step Giant Step) • Chinese Remainder Theorem (non-coprime moduli) • Lucas Theorem • Fast Walsh-Hadamard Transform • Matrix Exponentiation in Rings • Elliptic Curve Arithmetic (Rare)

ChatGPT said:

Here are **30 one-line problem statements** for **Advanced Number Theory** topics:

🔐 1–5: Discrete Logarithm (Baby Step Giant Step)

1. Solve for xxx in ax≡bmod ma^x \equiv b \mod max≡bmodm where a,b,ma, b, ma,b,m are given.

2. Compute the minimum exponent xxx such that gx≡ymod pg^x \equiv y \mod pgx≡ymodp (primitive root).
3. Given a,b,ma, b, ma,b,m, determine if discrete logarithm exists and compute it.
4. Solve xk≡amod px^k \equiv a \mod pxk≡amodp for smallest xxx using discrete log tricks.
5. Break a cryptographic challenge involving modular exponentiation inversion.

□ 6–10: Chinese Remainder Theorem (non-coprime moduli)

6. Solve a system of congruences with non-coprime moduli.
7. Determine the smallest xxx such that x≡aimod mix \equiv a_i \mod m_ix≡aimodmi for given (ai,mi)(a_i, m_i)(ai,mi).
8. Given multiple periodic events with cycles (not coprime), find first collision time.
9. Determine if a solution exists for a modular system with overlapping constraints.
10. Use extended CRT to compute synchronized time in distributed systems.

🔄 11–15: Lucas Theorem

11. Compute C(n,k)mod pC(n, k) \mod pC(n,k)modp for large n,kn, kn,k and small prime ppp.
12. Given n,k≤1018n, k \leq 10^{18}n,k≤1018, compute binomial coefficient modulo p.
13. Count combinations in modular arithmetic where values are massive.
14. Implement Lucas Theorem for combinatorial identities mod p.
15. Count paths in a grid modulo p where dimensions exceed standard ranges.

⚡ 16–20: Fast Walsh-Hadamard Transform (FWHT)

16. Perform XOR convolution between two integer arrays.
17. Given two sets A and B, find number of pairs such that a⊕b=ca \oplus b = ca⊕b=c.
18. Count subsets with XOR equal to a target value using FWHT.
19. Optimize bitwise convolution operations in signal processing context.
20. Compute subset convolution using FWHT efficiently.

□ 21–25: Matrix Exponentiation in Rings

21. Compute the nth Fibonacci number modulo a composite number.
22. Raise a 2×2 matrix with modular entries to large power efficiently.
23. Solve linear recurrences under ring modulo operations.
24. Simulate population growth using matrix recurrence modulo m.
25. Implement matrix fast power over Z/mZ for generalized recurrences.

□ 26–30: Elliptic Curve Arithmetic (Rare)

26. Add two points on an elliptic curve over a finite field.
27. Compute scalar multiplication kPkPkP on elliptic curves modulo p.
28. Given curve equation and point, determine order of the point.
29. Solve ECDLP: find kkk such that P·k=QP \cdot k = QP·k=Q on a given curve.
30. Use elliptic curve arithmetic to factor a number using ECM (Elliptic Curve Method).

Let me know if you'd like detailed tutorials, implementation strategies, or references for Olympiad or ACM ICPC-style contests involving these topics.

You said:

Strings (Advanced) • Suffix Automaton + DP on Automaton • Suffix Tree with Ukkonen's Algorithm • Z-algorithm & KMP in combinatorics • Aho-Corasick Automaton (multi-pattern) • Palindromic Tree (Eertree)

ChatGPT said:

Here are **30 one-line problem statements** for **Advanced String Algorithms**:

📈 1–5: Suffix Automaton + DP on Automaton

1. Count the number of different substrings of a given string using a suffix automaton.
2. Find the length of the longest common substring between two strings using SAM.
3. Given a string, count how many times each substring appears using DP on SAM.
4. Compute the lexicographically smallest substring of length k using suffix automaton traversal.
5. Find the number of times a pattern occurs as a substring in a large string using SAM.

🌲 6–10: Suffix Tree (Ukkonen's Algorithm)

6. Build the suffix tree of a string in linear time using Ukkonen's algorithm.
7. Find the longest repeated substring of a string using its suffix tree.
8. Count total number of different substrings using suffix tree node depth.
9. Compute LCS (Longest Common Substring) of two strings using generalized suffix tree.
10. Search for all occurrences of a pattern in a string using a suffix tree.

🔍 11–15: Z-algorithm & KMP in Combinatorics

11. Compute the number of times a pattern appears in a text using the Z-algorithm.
12. Find the minimal string period using the prefix function (KMP).
13. Count number of borders (prefix = suffix) in a string using Z-array.
14. Determine all occurrences of a pattern in linear time using KMP.
15. Check if a string is composed of repeated patterns using prefix function.

□ 16–20: Aho-Corasick Automaton (Multi-pattern Matching)

16. Search for multiple patterns simultaneously in a large text using Aho-Corasick.
17. Count total number of pattern occurrences in a string using Aho-Corasick.
18. Report all positions of any of the given dictionary words in a long text.
19. Detect banned or sensitive words in real-time chat using multi-pattern matching.
20. Count total distinct patterns matched at each position in the text using failure links.

🔄 21–25: Palindromic Tree (Eertree)

21. Count the number of distinct palindromic substrings in a string.
22. Maintain a dynamic structure to insert characters and get palindromic count in real-time.
23. For each suffix of a string, report the longest palindromic substring ending there.
24. Find the longest palindromic prefix/suffix using Eertree traversal.
25. Count how many times each palindrome occurs in the string.

🎯 26–30: Mixed Advanced Tasks

26. Given a string, find the longest palindromic substring using Manacher's algorithm.
27. Build suffix array in O(n log n) and find LCP array efficiently.
28. Compress all suffixes of a string into a trie and analyze space complexity.
29. Use hashing + Z/KMP to check string equivalence under rotations.
30. For multiple strings, find the shortest common superstring using suffix-prefix matches.