

Here are **30 one-line problem statements** covering advanced **Dynamic Programming** topics:

◆ Bitmask DP ($n \leq 20$)

1. Count Hamiltonian paths in a DAG.
 2. Solve the Travelling Salesman Problem (TSP) with minimum cost.
 3. Assign n jobs to n people minimizing total cost.
 4. Partition numbers into k groups with constraints.
 5. Count the number of valid subset selections under XOR condition.
-

◆ Divide & Conquer DP (Monotonicity of $\text{opt}[i]$)

6. Partition an array into k segments minimizing segment cost.
 7. Minimize total cost of merging subarrays under specific cost function.
 8. Optimize k -interval selection problem with convex cost function.
 9. Compute min edit distance with grouped edits.
 10. Maximize points from non-overlapping segments using D&C DP.
-

◆ Convex Hull Trick / Line Container (Convex/Concave DP)

11. Minimize total cost when $\text{dp}[i] = \min(\text{dp}[j] + a[i]*b[j])$.
 12. Solve slope-based DP for linear recurrence optimization.
 13. Optimize quadratic recurrence $\text{dp}[i] = \min(\text{dp}[j] + (i - j)^2 + C)$.
 14. Determine optimal factory placement to minimize total distance cost.
 15. Compute maximum reward with linearly increasing cost penalties.
-

◆ Knuth Optimization (Quadrangle Inequality)

16. Matrix Chain Multiplication with minimum cost.
 17. Optimal Binary Search Tree construction with minimal expected cost.
 18. Merge stone piles with minimum total merge cost.
 19. Optimal parentheses grouping to minimize score.
 20. Find min total length of compressed strings with split merges.
-

◆ Monotonic Queue Optimization (Sliding Range DP)

21. Maximum sum of subarrays of size $\leq k$ with no adjacent overlap.

22. DP on array where $dp[i] = \max(dp[j] + a[i])$ for $j \in [i-k, i-1]$.
 23. Find longest subarray with sum constraints in $O(n)$.
 24. Maximize beauty from k -consecutive selection with penalties.
 25. Jump game with variable distance and monotonic optimization.
-

◆ Tree DP (Rerooting / Subtree Aggregation)

26. Count number of ways to color tree nodes with no adjacent same color.
 27. Maximize sum of values on tree paths with exactly k edges.
 28. For each node, find size of largest subtree rooted at it.
 29. Compute distances from all nodes to all others (rerooting).
 30. Maximize matching (node pairs) in a tree.
-

Let me know if you want detailed explanations or code templates for any specific set.

You said:

. Graph Algorithms (Advanced) • Max Flow / Min Cut: Dinic's Algorithm, Edmonds-Karp • Strongly Connected Components: Kosaraju / Tarjan • 2-SAT (implication graphs) • Bridges, Articulation Points • Centroid Decomposition • Heavy-Light Decomposition (HLD) • Link-Cut Trees / Euler Tour Tree (rare but appear) • Topological Order + DP

ChatGPT said:

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

◆ Max Flow / Min Cut (Dinic's, Edmonds-Karp)

1. Find the maximum bipartite matching using flow.
 2. Compute the max number of edge-disjoint paths from s to t .
 3. Determine the min number of edges to disconnect source from sink.
 4. Find min cut in a graph dividing nodes into two groups.
 5. Assign jobs to workers with capacity constraints (multi-source flow).
-

◆ Strongly Connected Components (Kosaraju / Tarjan)

6. Count the number of strongly connected components in a directed graph.
7. Collapse SCCs into a DAG and find topological order.
8. Detect cycles in a directed graph using SCC.
9. Find the size of the largest strongly connected component.
10. Check if a directed graph is strongly connected.

◆ 2-SAT (Implication Graph)

11. Determine satisfiability of given boolean formula in 2-CNF form.
12. Assign truth values to variables to satisfy all clauses if possible.
13. Build implication graph from constraints and solve using SCC.
14. Detect contradictory variable conditions in a puzzle.
15. Design a scheduling system with mutual exclusivity constraints.

◆ Bridges and Articulation Points

16. Find all bridges in an undirected graph.
17. Identify articulation points (cut vertices) in a network.
18. Count number of components after each bridge is removed.
19. Check if a graph becomes disconnected after removing a vertex.
20. Build a bridge-tree (block tree) of a graph.

◆ Centroid Decomposition (on Trees)

21. Solve path queries on tree with subtree decompositions.
22. Count pairs of nodes whose distance is $\leq k$ using centroid decomposition.
23. Perform subtree updates efficiently using centroid hierarchy.
24. Minimize time for virus to spread from multiple roots in tree.
25. Solve tree-based range sum queries with centroid divide & conquer.

◆ Heavy-Light Decomposition (HLD)

26. Answer path queries (sum/min/max) in a tree in $\log^2(n)$ time.
27. Process subtree and path updates efficiently in a tree.
28. Find kth ancestor on a tree path.
29. Query LCA (Lowest Common Ancestor) using HLD.
30. Maintain and update values along paths in a rooted tree.

Let me know if you want implementation tips or visual explanations for any of these.

You said:

Data Structures • Persistent Segment Tree • Mergeable Heaps (Treap/Splay) • Lazy Segment Tree with Complex Updates • Li Chao Segment Tree (for line intersection problems) • Sqrt-Decomposition with Offline Queries • Binary Lifting + LCA with Modifications

ChatGPT said:

Here are **30 one-line problem statements** based on **Advanced Data Structures**:

◆ Persistent Segment Tree

1. Find the k-th smallest element in a subarray over time.
 2. Maintain historical versions of an array and answer range sum queries.
 3. Count frequency of elements in a versioned array range.
 4. Perform point updates while preserving previous versions.
 5. Query number of elements $\leq x$ in a given subarray version.
-

◆ Mergeable Heaps (Treap / Splay Tree / Leftist Tree)

6. Implement a priority queue with efficient merge operation.
 7. Maintain a multiset with insertion, deletion, and split/merge operations.
 8. Support range reversal and order-statistics in a sequence.
 9. Perform dynamic set union operations efficiently.
 10. Maintain a balanced tree with support for lazy propagation.
-

◆ Lazy Segment Tree with Complex Updates

11. Range add and range multiply over an array with modulo operations.
 12. Support range min updates and range queries.
 13. Perform range assignment and range sum queries.
 14. Answer RMQ with range XOR updates.
 15. Implement a segment tree with range affine transformation.
-

◆ Li Chao Segment Tree (for line intersections / CHT)

16. Maintain a dynamic set of lines and answer min y-value at x.
 17. Add lines $y = mx + b$ and query min/max y at point x.
 18. Solve dynamic convex hull problem for monotonic x-queries.
 19. Maintain envelope of lines for convex optimization.
 20. Combine Li Chao tree with divide & conquer for DP optimization.
-

◆ Sqrt-Decomposition with Offline Queries

21. Answer range sum queries with point updates using \sqrt{N} blocks.

22. Count frequency of elements in subarray (Mo's Algorithm).
23. Find mode (most frequent number) in a subarray offline.
24. Support range updates and point queries efficiently with blocks.
25. Answer number of distinct elements in subarrays using \sqrt{N} .

◆ Binary Lifting + LCA with Modifications

26. Find LCA of two nodes in a tree using binary lifting.
27. Find kth ancestor of a node in a rooted tree.
28. Modify tree and still answer LCA queries efficiently.
29. Compute path sum between two nodes using binary lifting.
30. Maintain dynamic tree and handle parent replacement queries.

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

You said:

Mathematics / Number Theory • Discrete Log (Baby-step Giant-step) • Euler's Theorem, Fermat's Little Theorem • Extended Euclidean Algorithm • Chinese Remainder Theorem (CRT) • Primitive Roots, Orders • Combinatorics with Lucas Theorem • Polynomial Interpolation (Lagrange)

ChatGPT said:

Here are **30 one-line problem statements** based on **Mathematics / Number Theory**:

◆ Discrete Logarithm (Baby-step Giant-step)

1. Solve for x in $ax \equiv b \pmod{m}$ efficiently.
2. Find smallest exponent x such that $2^x \equiv 123 \pmod{1009}$.
3. Break RSA-like encryption with known base and modulus.
4. Crack a modular exponentiation puzzle using BSGS.
5. Detect cycles in power mod sequences using discrete log.

◆ Euler's Theorem / Fermat's Little Theorem

6. Compute $a^{-1} \pmod{p}$ using Fermat's theorem.
7. Calculate $ab \pmod{m}$ efficiently using modular exponentiation.
8. Find last digit of large exponential towers $a^{b^c} \pmod{p}$.
9. Solve modular inverse problem for non-prime modulus using Euler's theorem.

10. Count number of coprime integers less than n (Euler's Totient function).

◆ Extended Euclidean Algorithm

11. Find modular inverse of a modulo m .
 12. Solve linear Diophantine equation $ax+by=c$ $ax + by = c$.
 13. Find integers x and y such that $ax+by=\gcd(a,b)$ $ax + by = \gcd(a, b)$.
 14. Reduce fractions modulo m with $\gcd > 1$.
 15. Determine if solution exists for $ax \equiv b \pmod m$ $x \equiv b \pmod m$.
-

◆ Chinese Remainder Theorem (CRT)

16. Solve a system of congruences with pairwise coprime moduli.
 17. Reconstruct a number given its remainders mod several bases.
 18. Find smallest x such that $x \equiv a \pmod m$ $x \equiv a \pmod m$ and $x \equiv b \pmod n$ $x \equiv b \pmod n$.
 19. Implement generalized CRT for non-coprime moduli.
 20. Apply CRT in solving scheduling problems with different cycle lengths.
-

◆ Primitive Roots / Orders

21. Find a primitive root modulo a prime p .
 22. Calculate the order of an integer modulo p .
 23. Generate all primitive roots of a given prime.
 24. Count elements of given order modulo p .
 25. Use primitive roots for efficient discrete log mapping.
-

◆ Combinatorics with Lucas Theorem

26. Compute $\binom{n}{k} \pmod p$ $\binom{n}{k} \pmod p$ where $n, k \leq 1e18$.
 27. Count paths in a grid modulo a prime using Lucas Theorem.
 28. Solve large combinatorics modulo non-prime using Lucas + CRT.
 29. Evaluate binomial coefficients modulo prime powers.
 30. Count valid sequences using combinations modulo p (Lucas).
-

Let me know if you'd like code or deeper explanations for any of these topics.

You said:

Geometry • Convex Hull Trick (Dynamic Hull) • Rotating Calipers • Line Sweep with Event Queue • Half Plane Intersection • Voronoi Diagram (rare) • Geometry + Binary Search / Ternary Search

ChatGPT said:

Here are **30 one-line problem statements** based on **Computational Geometry (Advanced)**:

◆ Convex Hull Trick (Dynamic Hull / Line Container)

1. Maintain a set of lines and query minimum y-value at given x.
 2. Optimize DP where $dp[i] = \min_{j \in [0, i-1]} (dp[j] + a[i] \cdot b[j])$ $dp[i] = \min(dp[j] + a[i] \cdot b[j])$
 3. Solve range linear function minimization using a convex envelope.
 4. Track lowest cost over time with changing linear functions.
 5. Maintain max profit function over time from a set of lines.
-

◆ Rotating Calipers

6. Find the diameter of a convex polygon.
 7. Compute the minimum width of a convex polygon.
 8. Determine the farthest pair of points in a convex hull.
 9. Find the smallest enclosing rectangle of a convex polygon.
 10. Compute max area triangle inside a convex polygon.
-

◆ Line Sweep with Event Queue

11. Count number of intersections between horizontal and vertical lines.
 12. Determine if any pair of line segments intersect.
 13. Compute union length of several overlapping intervals.
 14. Calculate total area covered by multiple rectangles.
 15. Find closest pair of points in 2D in $O(n \log n)$ $O(n \log n)$ $O(n \log n)$.
-

◆ Half Plane Intersection

16. Find intersection polygon of multiple half-planes.
17. Determine if a convex region remains after cuts.
18. Compute largest convex region satisfying a set of inequalities.
19. Solve linear inequalities geometrically using half-planes.
20. Find feasible region of a system of linear constraints.

◆ Voronoi Diagram (Rare)

21. Construct a Voronoi diagram for a set of 2D points.
 22. Find nearest site to any query point using Voronoi structure.
 23. Generate equidistant region boundaries between points.
 24. Apply Voronoi to model cell tower coverage.
 25. Use Voronoi to solve the closest-pair regionally with partitions.
-

◆ Geometry + Binary Search / Ternary Search

26. Find point on curve minimizing distance to a fixed point (ternary).
 27. Binary search angle to align two points with geometric condition.
 28. Maximize height of trapezium using ternary search on length.
 29. Search for optimal circle radius to cover all points.
 30. Ternary search on convex polygon perimeter for optimal point.
-

Let me know if you want code or geometric visualizations for any topic!

You said:

. Strings • Suffix Array + LCP Array • Suffix Automaton • Palindromic Tree (Eertree) • Z-Algorithm, KMP, Aho-Corasick • Rolling Hash with Double Hashing • Lyndon Decomposition

ChatGPT said:

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

◆ Suffix Array + LCP Array

1. Construct the suffix array of a given string in $O(n \log n)$.
 2. Count the number of distinct substrings in a string.
 3. Find the longest repeated substring using the LCP array.
 4. Lexicographically sort all cyclic shifts of a string.
 5. Perform fast pattern matching using binary search on suffix array.
-

◆ Suffix Automaton

6. Count the number of distinct substrings in a string using SAM.

7. Find the number of times a substring appears in the text.
 8. Find longest common substring between two strings using SAM.
 9. Construct minimal DFA for all substrings of a string.
 10. Track all substring occurrences in real-time as the string grows.
-

◆ Palindromic Tree (Eertree)

11. Count all distinct palindromic substrings in a string.
 12. Find the longest palindromic substring efficiently.
 13. For each prefix, count palindromic suffixes.
 14. Build online palindrome detector using Eertree.
 15. Track frequency of each palindromic substring.
-

◆ Z-Algorithm, KMP, Aho-Corasick

16. Find all occurrences of a pattern in a string using KMP.
 17. Build prefix function for KMP in linear time.
 18. Compute Z-array and use it to detect pattern matches.
 19. Find number of times multiple patterns occur in a text using Aho-Corasick.
 20. Detect all dictionary words in a text string.
-

◆ Rolling Hash with Double Hashing

21. Detect if two substrings are equal in $O(1)O(1)O(1)$ using hashes.
 22. Count number of unique substrings in a string using rolling hash.
 23. Find the longest common prefix of two suffixes.
 24. Detect string rotations using hashing.
 25. Use double hashing to avoid collisions in substring comparisons.
-

◆ Lyndon Decomposition

26. Decompose string into lexicographically minimal Lyndon words.
 27. Find minimal lex rotation using Lyndon decomposition.
 28. Build suffix automaton-like structures using Lyndon decomposition.
 29. Apply Duval's algorithm to get lex segments.
 30. Use Lyndon factorization in combinatorics and counting substrings.
-

Let me know if you'd like C++ or Java templates for any of these techniques!

You said:

Constructive Algorithms • Invariants / Monovariants • Greedy with Observations • Bitwise Constructive Proofs • Interactive Problems

ChatGPT said:

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

◆ Invariants / Monovariants

1. Show if a pile of stones can be reduced to zero with given moves.
 2. Determine if a parity-based transformation leads to a stable state.
 3. Construct a sequence where sum remains constant under operations.
 4. Decide if a grid can be colored without violating row/column rules.
 5. Given allowed operations, prove if final state is reachable.
-

◆ Greedy with Observations

6. Rearrange elements to form a non-decreasing sequence with swaps.
 7. Construct the lexicographically smallest valid string.
 8. Distribute candies optimally among children based on rating.
 9. Arrange elements such that no prefix sum is zero.
 10. Build the minimum cost tree by selecting edges greedily.
-

◆ Bitwise Constructive Proofs

11. Construct two integers with given AND, OR, XOR values.
 12. Generate an array with a given XOR and subset-sum condition.
 13. Find three numbers whose sum and XOR equals given values.
 14. Construct a permutation with given bitwise properties.
 15. Build a binary matrix satisfying row and column XOR constraints.
-

◆ Interactive Problems

16. Guess a hidden number using queries with XOR feedback.
17. Determine the position of the minimum element via comparisons.
18. Reconstruct a permutation by querying element positions.
19. Find a hidden edge in a graph using minimal interaction.
20. Deduce a hidden binary string using prefix/suffix queries.

◆ Mix: Constructive Patterns

21. Construct a balanced parentheses string of given length.
22. Design a grid with no two adjacent cells the same color.
23. Build a graph with a given degree sequence.
24. Generate a permutation with maximum number of inversions.
25. Construct two strings of length n with common subsequence of length k .
26. Place rooks on an $n \times n$ chessboard without attacks.
27. Create a binary string that avoids specific forbidden substrings.
28. Design a test case that breaks naive greedy algorithms.
29. Reorder tasks to ensure no deadline is missed.
30. Construct a matrix with given row and column sums.

Let me know if you want explanation-based tags (greedy/bit tricks/etc.) for each or full code for any.

You said:

Game Theory • Grundy Numbers / Sprague-Grundy Theorem • Impartial & Partisan Games
• Nim Game Variants

ChatGPT said:

Here are **30 one-line problem statements** based on **Game Theory**:

◆ Grundy Numbers / Sprague-Grundy Theorem

1. Determine the winner of a game where players remove 1, 3, or 4 stones.
2. Compute Grundy numbers for each pile in a composite game.
3. Given a graph, decide the winner based on game positions using Grundy values.
4. Combine multiple game states and determine the winning strategy using XOR.
5. Compute the Grundy number for a tree game where each move removes a subtree.

◆ Impartial & Partisan Games

6. Given a game where each player can remove 1 or 2 stones, but rules vary by turn, determine the winner.
7. Model a game where only Player A can skip a move, and decide if A can force a win.
8. Determine winner in a directed acyclic game with allowed moves varying for each player.

9. For a 2D grid game with different rules for both players, decide who has a winning strategy.
 10. Simulate a partisan game where moves differ for each player and evaluate the outcome.
-

◆ Nim Game Variants

11. Determine winner in standard Nim with multiple piles using XOR.
 12. Solve a Nim game where players can remove up to $k\sqrt{k}$ stones from a pile.
 13. Determine the winner in "misère" Nim (last to move loses).
 14. Handle Nim with pile-size constraints or forbidden pile sizes.
 15. Generalize Nim with different rules per pile and solve using Grundy numbers.
-

◆ Mixed Combinatorial Games

16. Given multiple independent games, determine the winner using Grundy sum.
 17. Analyze a game where players alternate turning off lights under specific constraints.
 18. In a chessboard game where players can remove rows/columns, decide the winner.
 19. Determine the winner of a coin-moving game on a graph.
 20. Given a number game where each move divides by a factor, decide if the first player wins.
-

◆ Constructive / Strategy-Based Game Theory

21. Design a winning strategy for a game with moves like Fibonacci subtraction.
 22. Construct losing and winning positions in a combinatorial game.
 23. Optimize moves in a game with a circular board and limited jump range.
 24. Construct a DP table to determine the winner for all pile sizes in a custom game.
 25. Find minimal Grundy numbers to force a win in a modified Wythoff's game.
-

◆ Applications & Tricky Setups

26. Determine if a given tic-tac-toe configuration is a winning state.
 27. Simulate a card picking game where the player can pick 1 or 2 cards.
 28. Analyze a token game where removing a token affects adjacent piles.
 29. Solve a grid-cutting game where players can cut along rows/columns.
 30. Determine optimal strategy in a game where moves are made on a trie.
-

Let me know if you'd like to build a Grundy table or code simulation for any of these!

You said:

Advanced Search / Optimization • Ternary Search on Unimodal Functions • Meet in the Middle (Advanced) • Divide & Conquer Optimization • Binary Search + Simulation / Check • Slope Trick (rare)

ChatGPT said:

Here are **30 one-line problem statements** based on **Advanced Search and Optimization Techniques**:

◆ Ternary Search on Unimodal Functions

1. Find the point minimizing $f(x) = (x-a)^2 + b \cdot \sin(x)$.
 2. Minimize travel time between two points with varying speed zones.
 3. Locate the optimal launch angle for maximum projectile distance.
 4. Find minimum value of a convex function over real domain.
 5. Maximize revenue function with unique maximum under constraints.
-

◆ Meet in the Middle (Advanced)

6. Count subsets with sum closest to a target ($n \leq 40$).
 7. Find two disjoint subsets with equal sum.
 8. Maximize subset sum not exceeding W using two-half enumeration.
 9. Partition array into two halves minimizing absolute sum difference.
 10. Solve knapsack with large weights using meet-in-the-middle.
-

◆ Divide & Conquer Optimization (Convex Cost Property)

11. Partition array into k segments minimizing a convex cost function.
 12. Optimize DP for minimizing sum of squared differences between segments.
 13. Solve k -median clustering in $O(k \cdot n \log n)$.
 14. Compute minimal edit costs with segment split points.
 15. Apply D&C optimization to range merging cost problem.
-

◆ Binary Search + Simulation / Check

16. Find the smallest radius required to cover all points with k circles.
17. Minimize maximum load per worker with binary search and greedy check.
18. Determine the minimum height needed to stack boxes without collapse.

19. Binary search the smallest valid time to complete all tasks under constraints.
 20. Find max possible value X such that array can be split into k valid segments.
-

◆ Slope Trick (Rare / Convex Function Maintenance)

21. Maintain a function $f(x)$ with min and shift operations to minimize total cost.
 22. Solve "minimize cost of making array non-decreasing" using slope trick.
 23. Compute min cost of aligning points with move and penalty constraints.
 24. Handle convex piecewise-linear function minimization with updates.
 25. Solve dynamic programming where the cost depends on absolute differences.
-

◆ Mixed + Applications

26. Perform ternary search on discrete unimodal function (integer version).
27. Find minimum time to reach target with fuel constraints using binary search.
28. Optimize a real-valued convex function using ternary search on double domain.
29. Split jobs between machines minimizing max load (binary search on answer).
30. Compute optimal scheduling time window using binary + greedy simulation.