

ChatGPT said:

Here are **30 one-line advanced DP problem topics** often seen in Div 2 D/E (1700–2200) contests:

1. Knapsack with multiple constraints (2D/3D DP)
2. Digit DP (count numbers with properties)
3. Bitmask DP (TSP, set states)
4. DP on trees (rooted trees, rerooting)
5. DP with memoization + state encoding
6. DP with prefix sums (e.g., range-based recurrence)
7. DP with sliding window optimization
8. DP with convex hull trick (CHT optimization)
9. Monotonic queue optimization in DP
10. SOS DP (Sum over Subsets DP)
11. Matrix exponentiation in DP transitions
12. Divide and conquer DP optimization
13. Knuth optimization in interval DP
14. DP on intervals (merge segments, game theory)
15. DP on subsequences (e.g., LIS in $O(N \log N)$)
16. Probabilistic DP (expected values)
17. Palindromic DP (count/partition substrings)
18. DP with combinatorics ($nCr \bmod p$, inclusion-exclusion)
19. Bit DP + Trie (bitmask + structure optimization)
20. Game DP (Grundy numbers, winning states)
21. DP with binary lifting (compressed transitions)
22. Tree DP with subtree merging
23. DP with cyclic states (modular state compression)
24. DP with event-based simulation (sweep line + DP)
25. Subset DP with mask pruning (e.g., dominating sets)
26. DP over permutations (count permutations with constraints)
27. DP with frequency table (count ways with value limits)
28. DP on strings with automaton (KMP/Z + transitions)
29. Functional graph DP (cycle+tree structure)
30. DP with time compression (scheduling + range DP)

Let me know if you want problems for practice on any of these types.

You said:

✓ DP on Trees Subtree sums, rerooting, bottom-up + top-down

ChatGPT said:

Here are **30 one-line DP on Trees problem types** — focused on subtree sums, rerooting, and bottom-up/top-down ideas:

-
1. Find subtree sum for each node (bottom-up).
 2. Count nodes in each subtree.

3. Longest path from each node to any leaf (bottom-up).
4. Longest path in tree (diameter using DP).
5. Number of ways to select K nodes in a subtree.
6. Max weighted path sum from root to leaf.
7. Max weighted path in subtree of each node.
8. DP to find LCA using binary lifting.
9. Rerooting: sum of distances from each node to all others.
10. Number of nodes at even depth in subtree.
11. Rerooting: max depth of all subtrees after rerooting.
12. DP to find max independent set in a tree.
13. Number of ways to color a tree (2-color/3-color).
14. Number of paths of length K starting at each node.
15. DP for minimum vertex cover on tree.
16. DP for maximum matching in tree.
17. DP for longest increasing/decreasing path on tree.
18. Count number of leaf nodes in subtree.
19. Count palindromic paths (via bitmask).
20. Count paths where sum of node values is divisible by K.
21. Tree centroid using subtree size DP.
22. Rerooting: compute contribution of each node to total sum.
23. Count of paths with even/odd length.
24. Expected path length from root to leaf.
25. DP for min/max cost to reach all leaves.
26. Count valid assignments in a tree with constraints.
27. Rerooting: number of subtrees rooted at each node.
28. Bottom-up DP: longest common prefix in subtree strings.
29. Find total number of ancestor-descendant pairs.
30. Compute GCD of subtree node values.

Let me know if you want full problem statements or implementation examples (in Java or C++).

You said:

✓ Graphs + Toposort + SCC Cycles, dependencies, DAGs

ChatGPT said:

Here are **30 one-line Graph DP + Toposort + SCC problem types** focused on **cycles, dependencies, and DAGs**:

1. Topological sort of a DAG (Kahn's/DFS).
2. Detect cycle in directed graph using DFS stack.
3. Count number of topological orderings.
4. Longest path in a DAG (Topo + DP).
5. Minimum time to finish all tasks with dependencies.

6. Detect if a graph is a DAG.
 7. Number of paths from source to destination in DAG.
 8. Lexicographically smallest topological ordering.
 9. Find strongly connected components (Kosaraju's).
 10. Condense SCCs to form a DAG.
 11. Count SCCs in directed graph.
 12. Find nodes in a cycle using SCC.
 13. Determine whether each node is in a cycle.
 14. Compute reachability between nodes in DAG.
 15. DP on DAG: max/min cost path to each node.
 16. Toposort-based scheduling of jobs/tasks.
 17. Count of paths that visit each node exactly once (in DAG).
 18. Longest path in SCC-condensed DAG.
 19. Maximum number of edges that can be added to remain DAG.
 20. Enumerate all SCCs and their sizes.
 21. Detect all entry/exit points in DAG components.
 22. Count number of source/sink nodes in DAG.
 23. Minimum path cover in DAG using matching.
 24. All ancestors/descendants of each node in DAG.
 25. Find safe nodes (that don't lead to cycles).
 26. SCC condensation + topological order processing.
 27. Number of cycles in graph (via SCC size > 1).
 28. Compress graph via SCC and solve on compressed DAG.
 29. Reachability matrix in DAG using bitsets.
 30. Shortest path in DAG (Topo + Relaxation).
-

Want full problems, visual explanation, or C++/Java code templates for any of these?

You said:

✓ Binary Search on Function + Monotonicity Used with DP or Greedy to optimize

ChatGPT said:

Here are **30 one-line problems based on Binary Search on Answer (Function) + Monotonicity**, often used with **DP or Greedy** for optimization:

1. Minimize max subarray sum when splitting into K parts.
2. Minimize the largest distance to place K gas stations.
3. Maximize minimum distance between placed elements (aggressive cows).
4. Find smallest time to finish all tasks with given speed.
5. Find smallest X such that a task is doable in X units (Greedy check).
6. Maximize median after selecting K elements.
7. Allocate minimum number of pages to K students (partition-based).
8. Minimize max load on a machine with job splitting.
9. Minimize the cost of assigning tasks under constraints.

10. Maximize smallest element you can keep in array after K removals.
11. Binary search on number of operations needed to reach a goal.
12. Binary search on minimum max jump (frog jump problem).
13. Find min max height after performing K operations.
14. Optimize threshold value such that sum of elements $\leq X$.
15. Find max length of subarray with avg $\geq X$.
16. Maximize the minimum sweetness of chocolate pieces.
17. Minimize the max number of products a person handles.
18. Minimize max wait time in job scheduling.
19. Minimum eating speed to finish bananas in H hours.
20. Max value of X such that DP condition holds.
21. Binary search on max tower height with increasing width.
22. Max X such that sum of floor(A[i]/X) $\geq K$.
23. Minimum fuel needed for all trucks to reach destination.
24. Maximize length of rope pieces from a given length.
25. Find lowest difficulty path using binary search on edge weights.
26. Maximize k such that at least K elements $\geq k$ (Monotonic check).
27. Find min max cost to cut a rod in pieces.
28. Min time needed to paint boards with K painters.
29. Maximize distance with limited jumps (Greedy + BS).
30. Max budget X such that total profit is \geq target.

Let me know if you want categorized templates (DP-based, Greedy-based), or specific problems with test cases.

You said:

Math (Modular Arithmetic, nCr, Fermat, Totient) Needed in construction, counting

ChatGPT said:

Here are **30 one-line Math problems involving Modular Arithmetic, nCr, Fermat's Little Theorem, and Euler's Totient Function**, commonly used in **construction and counting** problems:

1. Compute $a^b \bmod m$ efficiently (modular exponentiation).
2. Find modular inverse of $a \bmod m$ (using Fermat or Extended GCD).
3. Compute $n! \bmod p$ for large n .
4. Calculate $nCr \bmod p$ using Fermat's theorem.
5. Precompute factorials and inverse factorials modulo ppp .
6. Count subsets of size k in a set modulo 10^9+7 .
7. Find number of combinations $C(n,r) \bmod m$ for multiple queries.
8. Compute Euler's Totient $\phi(n)$ for a single or all numbers up to NNN .
9. Count coprime integers $\leq N$.
10. Find last non-zero digit of factorial modulo ppp .
11. Compute large Fibonacci number modulo mmm .

12. Count permutations modulo 10^9+7 .
13. Count derangements $(n!) / e \bmod p$.
14. Compute $a/b \bmod m$ using modular inverse.
15. Find smallest primitive root modulo p .
16. Construct number with given remainder mod m .
17. Count ordered/unordered partitions with constraints mod p .
18. Calculate binomial coefficient modulo non-prime m .
19. Solve linear congruence $ax \equiv b \pmod m$.
20. Find sum $1^k+2^k+\dots+n^k \bmod m$.
21. Count integer solutions to $ax+by=n$.
22. Find number of divisors of n .
23. Construct number divisible by m with specific digits.
24. Compute sum over all subsets modulo m .
25. Calculate inverse of every number up to N mod p .
26. Use Lucas Theorem to compute $\binom{n}{r} \bmod p$ for large n .
27. Count distinct sequences modulo symmetry.
28. Construct matrix such that determinant is divisible by m .
29. Count permutations avoiding fixed points mod p .
30. Use CRT (Chinese Remainder Theorem) for multi-mod construction.

Let me know if you want categorized practice (e.g., Fermat-only, Totient-specific), or code templates in C++ or Java.

You said:

DSU with path compression Connected components, Kruskal's algorithm

ChatGPT said:

Here are **30 one-line problem types** based on **Disjoint Set Union (DSU)** with **path compression**, commonly used in **connected components**, **Kruskal's algorithm**, and related graph topics:

1. Count connected components in an undirected graph.
2. Union-Find with path compression and union by size/rank.
3. Detect cycle in an undirected graph using DSU.
4. Kruskal's algorithm to find Minimum Spanning Tree (MST).
5. Number of redundant edges (extra edges forming cycles).
6. Check if all nodes are in the same connected component.
7. Count number of disjoint sets after union operations.
8. Restore graph connectivity with DSU after removing edges.
9. Merge cities with roads and track components over time.
10. Offline dynamic connectivity queries using DSU.
11. Check bipartiteness with DSU + 2-coloring trick.
12. Group elements by common properties (same value, modulo, etc).
13. Find maximum number of edges without disconnecting graph.
14. Count components with even/odd size.

15. Track size of component after each merge.
16. DSU with rollback for persistent operations.
17. Kruskal's algorithm to find second-best MST.
18. Union of people by friendship, find group sizes.
19. Find if adding an edge creates a bridge (Kruskal-like logic).
20. Minimum number of swaps to make arrays identical (via DSU cycles).
21. Dynamic union operations with path compression.
22. Minimum cost to connect all nodes with optional roads.
23. Identify friend groups using DSU (e.g., social networks).
24. Count number of islands (grid + DSU).
25. Find if two elements are in same set (online queries).
26. Compress strings/arrays based on connected characters/values.
27. Grouping people with constraints (e.g., parity, weight class).
28. DSU on trees to merge subtree info efficiently.
29. Tracking connected cities after edge deletions (reverse Kruskal).
30. Offline LCA queries via Tarjan's Algorithm using DSU.

Let me know if you'd like Java/C++ code templates for **DSU with path compression**, **Kruskal's**, or **union by rank**.

You said:

Segment Trees / Fenwick Trees Range queries, updates, lazy propagation

ChatGPT said:

Here are **30 one-line problem types** based on **Segment Trees** and **Fenwick Trees (Binary Indexed Trees)** used for **range queries, updates, and lazy propagation**:

1. Range sum query and point update.
2. Point query and range add update (Fenwick Tree with difference array).
3. Range minimum query (RMQ) and point update.
4. Range maximum query and point update.
5. Count of elements in a range (frequency/count queries).
6. Range XOR query with point update.
7. Range GCD query with update.
8. Range sum with lazy propagation for range update.
9. Range add and range sum using lazy segment tree.
10. Range assignment and range sum (lazy propagation + overwrite).
11. Kth order statistic in subarray using BIT or Wavelet Tree.
12. Find first/last index where prefix sum satisfies condition (binary lifting on segtree).
13. Number of inversions in array using Fenwick Tree.
14. Persistent Segment Tree for historical queries.
15. Dynamic Segment Tree for large index ranges (up to $1e9$).
16. 2D Fenwick Tree for submatrix sum queries.
17. Segment Tree Beats for min/max with complex constraints.
18. Coordinate compression + BIT for frequency/count queries.

19. Lazy propagation to handle multiple range updates.
20. Range increment and range minimum query.
21. Count number of elements $\leq k$ in prefix/suffix (use BIT).
22. Range sum with modulo operations.
23. Maximum subarray sum (Kadane's in segment tree).
24. Segment Tree with custom merge (e.g., pair sum, product).
25. Range parity queries (even/odd count).
26. Frequency of each element in subarray (with Mo's or segment tree of maps).
27. Update all multiples of X in a range (seg tree or sieve-like propagation).
28. Range product queries with modulo.
29. Segment tree on trees (HLD + subtree queries).
30. Difference array + BIT for range addition and point queries.

Let me know if you want categorized code templates in **C++ or Java** for standard segment tree, **lazy segment tree**, or **Fenwick Tree** with coordinate compression.

You said:

Bitmask DP For state compression when $n \leq 20$

ChatGPT said:

Here are **30 one-line problem types** based on **Bitmask Dynamic Programming (Bitmask DP)** — ideal for **state compression when $n \leq 20$** :

1. Count number of Hamiltonian paths in a graph (TSP-style).
2. Solve Traveling Salesman Problem (TSP) using DP + bitmask.
3. Count number of ways to assign N jobs to N people (assignment problem).
4. Minimum cost to cover all elements using subsets.
5. Minimum path length to visit all cities (complete graph).
6. Bitmask DP to count ways to tile a grid (e.g., dominos).
7. Max score by assigning elements based on matching rules.
8. Partition set into 2 groups with equal sum using bitmask.
9. Count number of valid configurations using previous states.
10. Count independent sets in a small graph using mask.
11. Subset sum problem using DP with bitmask.
12. Count number of perfect matchings in a bipartite graph.
13. Maximize profit selecting subsets under constraints.
14. DP to find max number of tasks done under time/energy limits.
15. Count permutations with adjacency constraints using bitmask.
16. Count valid bracket sequences using DP with mask state.
17. DP on strings: count palindromic permutations using mask.
18. Count subset pairs with specific bitwise AND/OR/XOR.
19. Count bitmask subsets satisfying custom rules (e.g., primes).
20. Find max number of disjoint subsets satisfying some rule.
21. Minimum steps to make all bits 1 using toggles (toggle mask).
22. DP with mask + position (e.g., travelling with time windows).

23. Assign K elements to N slots with dependency masks.
 24. Bit DP on subsets with fixed number of bits set.
 25. Min cost to remove all elements where removing one affects others.
 26. DP with memo[state][last] for subset permutations.
 27. Tiling problem where each tile depends on prior positions.
 28. DP on DAG with bitmask over visited nodes.
 29. Count subset paths where node i can be visited only if j is.
 30. Solve Sudoku-like constraints using mask state propagation.
-

Want categorized templates (TSP-type, Subset Matching, Assignment), or code in **Java/C++** for these styles?

You said:

Digit DP / Trie / Bitwise Tricks Subset constraints, XOR matching

ChatGPT said:

Here are **30 one-line problems** covering **Digit DP**, **Trie**, and **Bitwise Tricks** — ideal for handling **subset constraints**, **XOR matching**, **counting**:

✔ **Digit DP (Counting numbers with digit constraints):**

1. Count numbers $\leq N$ with exactly K digits.
 2. Count numbers $\leq N$ with no repeated digits.
 3. Count numbers $\leq N$ where sum of digits = K.
 4. Count numbers divisible by D up to N.
 5. Count numbers $\leq N$ having digits only from {1, 3, 5}.
 6. Count palindromes $\leq N$.
 7. Count numbers with alternating even/odd digits.
 8. Count numbers with K non-zero digits.
 9. Count numbers where number of set bits = K.
 10. Count numbers with digits in non-decreasing order.
-

✔ **Trie (Bitwise and prefix-based problems):**

11. Insert binary strings and check prefix existence.
12. Find max XOR of any two numbers in an array.
13. Count number of pairs (i, j) such that $A[i] \text{ XOR } A[j] < K$.
14. Find max XOR of a prefix with any suffix.
15. Store binary strings and count prefix matches.
16. Minimum XOR pair among all array elements.
17. Find minimum unique prefix for each string.
18. Check if any string is a prefix of another (prefix code).

19. Count pairs with common prefix of length L.
 20. Maximize XOR of a number with elements in Trie.
-

✔ Bitwise Tricks & Subset Masks:

21. Count number of subsets with XOR = K.
 22. Find subset with max XOR value.
 23. Count bits set in all subsets of N bits (Hamming weight).
 24. Generate all subsets of a bitmask.
 25. Enumerate submasks of a bitmask efficiently.
 26. Fast SOS DP: compute $f(\text{mask}) = \text{sum over all submasks}$.
 27. Maximize $A[i] \text{ XOR } A[j]$ under some index conditions.
 28. XOR basis: find linearly independent set from XORs.
 29. Subset DP over XOR masks with frequency counts.
 30. Count good subsets where XOR is divisible by K.
-

Let me know if you want grouped templates or practice problems (e.g., “Digit DP + Modulo Constraints” or “Trie + XOR Query”) in **Java/C++**.

You said:

✔ Constructive Algorithms Build output string/array under logic constraints

ChatGPT said:

Here are **30 one-line Constructive Algorithm problem types**, where you're required to **build strings/arrays/numbers** under specific **logic or constraints**:

1. Construct a binary string with no two consecutive 1s.
2. Build a permutation with exactly K inversions.
3. Construct array such that prefix sums are strictly increasing.
4. Rearrange elements so that adjacent elements differ by at least K.
5. Build a palindrome from a given string if possible.
6. Construct an array where $\text{sum} = X$ and $\text{max} \leq M$.
7. Build a string with exactly A 0s and B 1s, no two 1s together.
8. Construct the lexicographically smallest/largest string under constraints.
9. Rearrange array such that all even-indexed elements are greater than odd.
10. Construct tree with N nodes and exactly K leaves.
11. Create array with pairwise $\text{GCD} > 1$.
12. Construct number divisible by D with digits in a given set.
13. Generate valid parenthesis string of length 2n.
14. Build a magic square / Latin square for given N.
15. Construct array with alternating signs and $\text{sum} = 0$.
16. Distribute values into K groups with equal or near-equal sum.

17. Rearrange elements so that no element stays in its original position (derangement).
18. Construct permutation such that each element is the sum of neighbors.
19. Create number with exactly X divisors.
20. Construct grid with alternating characters (e.g., chessboard).
21. Build array such that bitwise AND/OR/XOR = target.
22. Rearrange to get required prefix/suffix GCDs.
23. Construct binary string with exactly K substrings "01".
24. Build tree with a given diameter and constraints.
25. Build an array with K peaks or valleys.
26. Create string of length N with no palindromic substring of length $\geq K$.
27. Build the smallest number with digit sum = S and length N.
28. Construct N numbers such that their XOR is zero.
29. Build a graph with N vertices and M edges and no triangles.
30. Fill matrix such that each row and column has unique values (Latin matrix).