

Here are **30 one-line problem statements** from **Codeforces-style** problems, covering advanced **Dynamic Programming techniques** related to trees, segment trees, divide & conquer DP, and other optimization tricks like Knuth, Slope Trick, Alien's Trick, etc.

□ **Dynamic Programming on Trees**

1. Find the number of ways to select a subtree of a rooted tree such that it includes at most K leaves.
 2. For each node in a tree, compute the sum of distances to all other nodes.
 3. Count the number of rooted subtrees with even number of nodes.
 4. Compute the maximum path sum where no two adjacent nodes are chosen in the tree.
 5. Given weights on nodes, select a subset of nodes such that no two adjacent nodes are selected and maximize sum.
 6. Calculate the size of the largest independent set in a tree.
 7. Find the diameter of the tree with a constraint that every path node must satisfy a given value condition.
 8. Count the number of valid colorings of a tree such that no two adjacent nodes share the same color.
 9. For each node, find the maximum weight of a path starting from that node and moving downward.
 10. Maximize the value of choosing a subset of nodes such that chosen nodes form a connected subtree.
-

□ **DP with Segment Trees / BIT**

11. Given an array, find number of increasing subsequences of length K using BIT.
 12. Maintain the number of increasing triplets in an array under updates.
 13. Find the maximum sum of non-overlapping subarrays of fixed length using Segment Tree.
 14. For each position, find the length of the longest increasing subsequence ending there using BIT.
 15. Count the number of pairs (i, j) such that $A[i] < A[j]$ and $i < j$ using Fenwick Tree.
 16. Given updates and range max queries on a DP recurrence, use Segment Tree to answer optimally.
-

□ **Divide and Conquer DP**

17. Partition an array into K segments such that the total cost is minimized (cost defined as $f(l, r)$).
18. Minimize the cost to merge an array of stones where cost is defined as the sum of merged parts.
19. Minimize sum of squared distances by splitting array into k segments.
20. Find the minimal total sum of absolute differences by optimally placing K centers.

□ Monotonic Queue Optimization in DP

21. Given an array, for each i , compute $dp[i] = \min(dp[j] + \text{cost}(j, i))$ with $\text{cost}(j, i)$ monotonic.
22. Maximize total score in a sliding window DP recurrence where transitions involve max/min over windows.
23. Find the longest subarray for each position such that sum in range is $\leq K$ using Monotonic Queue.
24. Solve a DP where state depends on a minimum in a sliding window of previous values.

□ Knuth Optimization

25. Given a matrix of costs, compute the optimal way to parenthesize operations to minimize total cost.
26. Compute the minimal cost of joining segments with additive costs and Knuth property.
27. Minimize the total cost of merging K piles of stones using optimal partitioning.

□ Slope Trick

28. Minimize the total cost where $dp[i] = \min(dp[j] + |A[i] - A[j]| + C)$ using slope trick.
29. Find minimal cost to convert an array into non-decreasing order with operation cost proportional to distance moved.
30. Given a sequence, compute the minimum cost to build a convex hull where cost depends on height differences.

Let me know if you want **links to specific Codeforces problems** matching these or full solutions.

Ask ChatGPT

You said:

Heavy-Light Decomposition • Link-Cut Trees / Euler Tour Trees • Treaps / Splay Trees / Rope / Cartesian Trees • Persistent Segment Trees / BIT • Dynamic connectivity • Wavelet Trees • Li Chao Tree

ChatGPT said:

Here are **30 one-line Codeforces-style problem statements** spanning **Heavy-Light Decomposition, Link-Cut Trees, Advanced Balanced Trees, Persistent Data Structures,**

Dynamic Connectivity, Wavelet Trees, and Li Chao Trees — all suitable for high-level competitive programming.

□ **Heavy-Light Decomposition (HLD)**

1. Given a tree with weights on nodes, answer path sum queries and point updates.
 2. Find the LCA of two nodes and update edge weights along a path.
 3. On a tree, support queries to find maximum weight on path $u-v$ and update node weights.
 4. Count number of nodes with value $\leq K$ on path $u-v$ in a tree.
 5. Given a rooted tree, for multiple queries $u-v$, return the number of nodes in the path having a specific color.
-

□ **Link-Cut Trees / Euler Tour Trees**

6. Support adding/removing edges in a forest and querying subtree sums dynamically.
 7. Dynamically maintain the size of connected components and return max/min component sizes.
 8. Add edge between any two nodes and check if the whole tree remains a valid forest (acyclic).
 9. Process path queries and subtree reroot operations in an online tree using link-cut trees.
 10. Maintain the depth of nodes in a tree under subtree reattachment operations.
-

□ **Treaps / Splay Trees / Rope / Cartesian Trees**

11. Support insert, delete, reverse, and substring operations on a mutable string using Rope.
 12. Maintain an array with operations: insert at position, delete at position, find k th element.
 13. Given a sequence, support range reversal and range minimum queries.
 14. Merge two ordered sets in $O(\log n)$ using Cartesian trees and answer order-statistics queries.
 15. Maintain a dynamic sorted array and support insertions, deletions, and predecessor queries.
-

□ **Persistent Segment Trees / BIT**

16. For each prefix of array, build a versioned segment tree and answer k th smallest in subarray $[l, r]$.

17. Maintain array versions after each update and answer sum queries on any historical version.
 18. Given a dynamic array, for each new version, support range XOR queries with persistence.
 19. Implement rollback-able segment tree supporting undo of previous range updates.
 20. Maintain history of prefix sums and compute historical range queries efficiently.
-

□ **Dynamic Connectivity**

21. Given a graph, support online edge insertions and deletions, and answer connectivity between nodes.
 22. Support online queries to check if a path exists between nodes u and v with at most one removal.
 23. Dynamically build a spanning forest and return the number of components after each operation.
 24. Maintain connected components under dynamic union and split of nodes.
 25. Handle offline edge additions/removals and answer if nodes are connected at time t .
-

□ **Wavelet Trees**

26. Given an array, answer k th smallest in range $[l, r]$ in $O(\log n)$.
 27. Count number of occurrences of a value x in subarray $[l, r]$.
 28. In a dynamic array, answer number of elements $\leq K$ in range $[l, r]$.
 29. Compute the mode (most frequent element) in subarray $[l, r]$ using Wavelet Tree.
 30. Find the number of distinct values in a subarray using compressed Wavelet Trees.
-

Let me know if you'd like actual **Codeforces problem links**, implementation templates, or **specific topics (e.g., persistent segment tree with rollbacks)** broken down with visuals or code.

You said:

Convex Hull (Graham scan, Andrew's Monotone Chain) • Rotating Calipers • Sweep Line Algorithms • Half-plane Intersection • Segment Intersections • Circle-Circle, Circle-Line Intersections • Computational Geometry using complex numbers

ChatGPT said:

Here are **30 one-line Codeforces-style problem statements** based on **advanced Computational Geometry topics**, including **Convex Hull (Graham scan/Andrew's)**, **Rotating Calipers**, **Sweep Line**, **Half-Plane Intersection**, **Segment Intersections**, **Circle Intersections**, and **Complex Number Geometry**.

□ **Convex Hull (Graham Scan, Andrew's Monotone Chain)**

1. Given N points, compute the convex hull and return its perimeter.
 2. For a given set of points, count the number of convex hull vertices.
 3. Given N points, return the convex hull in counter-clockwise order starting from the lowest point.
 4. Determine if a new point lies inside the convex hull of existing points.
 5. Compute the area of the convex hull of N given points.
-

□ **Rotating Calipers**

6. Find the pair of points with the maximum distance (diameter) of the convex hull.
 7. Given a convex polygon, compute its minimal bounding rectangle using rotating calipers.
 8. Find the largest triangle area that can be formed using any 3 points from a convex polygon.
 9. Compute the width of a convex polygon (minimum distance between two parallel lines enclosing it).
 10. Find the closest pair of points on a convex polygon in $O(n)$ using calipers.
-

□ **Sweep Line Algorithms**

11. Given N segments, determine if any two of them intersect using sweep line algorithm.
 12. Count the number of segment intersections among a set of N segments.
 13. Given horizontal and vertical line segments, count number of intersections.
 14. Given start and end points of buildings on a line, compute the skyline (critical points).
 15. Maintain the number of overlapping intervals at each point on a line.
-

□ **Half-Plane Intersection**

16. Compute the intersection polygon of a given set of half-planes.
 17. Given constraints in the form of half-planes, find a point inside all of them (feasibility check).
 18. Calculate the area of the feasible region defined by a set of half-plane inequalities.
 19. Determine if the feasible region defined by half-planes is bounded.
 20. Count how many points from a given set lie inside the half-plane intersection polygon.
-

□ **Segment Intersections**

21. Determine if two given line segments intersect or touch.

22. Find the intersection point (if any) between two segments.
 23. Given N line segments, return all intersection points (with multiplicity if required).
 24. Check if a polygon (given as a list of edges) is simple (no self-intersections).
 25. Count number of intersection points when chords are drawn between all point pairs on a circle.
-

□ Circle-Circle and Circle-Line Intersections

26. Given two circles, determine their intersection points or report if none.
 27. Find the common area between two overlapping circles.
 28. Compute intersection points between a circle and a line.
 29. Count number of tangents (internal/external) that can be drawn between two circles.
 30. Given a set of points, find the minimal enclosing circle.
-

◆ Computational Geometry using Complex Numbers

31. Calculate the angle between two vectors using complex numbers.
 32. Rotate a point around another point by angle θ using complex multiplication.
 33. Compute area of polygon using the complex cross product method.
 34. Given three points, compute the oriented angle using complex numbers.
 35. Implement geometric transformations (rotation, reflection, translation) using complex arithmetic.
-

Let me know if you want actual **Codeforces** / **CSES** / **AtCoder** problem links for any of the above, or **Python/C++ templates** for convex hulls, rotating calipers, or complex-based geometry operations.

You said:

Graph Algorithms • 2-SAT and Strongly Connected Components • Centroid Decomposition • Dynamic Graph Algorithms • Link/Cut Trees for dynamic graphs • Kirchhoff's Theorem (Count spanning trees) • Eulerian Path/Circuit with Hierholzer's Algorithm • Minimum Cost Max Flow • Cycle detection and contraction • Directed Minimum Spanning Trees (Chu-Liu/Edmonds)

ChatGPT said:

Here are **30 one-line Codeforces-style problem statements** based on **advanced Graph Algorithms**, including **2-SAT & SCCs**, **Centroid Decomposition**, **Dynamic Graphs**, **Link/Cut Trees**, **Kirchhoff's Theorem**, **Eulerian Paths**, **MCMF**, **Cycle Detection/Contraction**, and **Directed MSTs (Chu-Liu/Edmonds)**.

□ **2-SAT and Strongly Connected Components (SCCs)**

1. Given a 2-CNF formula with N variables, determine if it is satisfiable.
 2. Given N switches and M constraints of the form $(A \text{ or } B)$, find a valid configuration.
 3. Count number of SCCs in a directed graph and their sizes.
 4. Compress a directed graph into its DAG of SCCs.
 5. Determine if it is possible to assign truth values to variables such that all XOR constraints are satisfied.
-

□ **Centroid Decomposition**

6. Given a tree, count the number of pairs of nodes with distance exactly K using centroid decomposition.
 7. Compute the number of node pairs whose path sum is divisible by D in a tree.
 8. Answer path-based queries in a tree under constraints using centroid decomposition for faster divide-and-conquer.
 9. Count number of pairs of nodes whose path contains exactly M red edges (binary edge coloring).
 10. On a tree, compute the total contribution of all paths satisfying a custom condition involving path XOR.
-

□ **Dynamic Graph Algorithms**

11. Support dynamic edge insertions and deletions and answer "are u and v connected?" queries.
 12. Maintain number of connected components after each edge update.
 13. Given a dynamic graph, maintain the number of bridges over time.
 14. Support dynamic queries for bipartiteness of the current graph.
 15. Determine for each edge deletion whether it disconnects the graph or not.
-

□ **Link/Cut Trees (Dynamic Trees)**

16. Given a dynamic tree, support path sum queries and subtree rerooting.
 17. Implement a dynamic forest with link, cut, and LCA queries.
 18. Maintain the maximum edge weight on the path between any two nodes under link/cut operations.
 19. Dynamically track the diameter of a forest under edge insertions/deletions.
 20. Perform reroot and update edge/node weights with queries on tree paths in logarithmic time.
-

□ **Kirchhoff's Theorem (Counting Spanning Trees)**

21. Count number of spanning trees in a given undirected graph using Matrix-Tree Theorem.
 22. Given a weighted undirected graph, compute the total weight of all spanning trees.
 23. For each edge, determine if removing it affects the total number of spanning trees.
 24. Compute the number of labeled trees on N nodes with given constraints.
 25. Calculate number of spanning trees in a graph modulo prime using Kirchhoff's theorem.
-

☐ **Eulerian Path / Circuit (Hierholzer's Algorithm)**

26. Given a directed graph, determine if an Eulerian path exists and construct it.
 27. Find an Eulerian circuit in an undirected multigraph and output the path.
 28. Reconstruct a word from overlapping pairs using Eulerian path in de Bruijn graph.
 29. Given an edge list, determine if the graph has a trail that uses every edge exactly once.
 30. Count number of Eulerian circuits possible in a multigraph.
-

☐ **Minimum Cost Max Flow (MCMF)**

31. Given a flow network with costs, compute the minimum cost of sending K units of flow.
 32. Find the optimal assignment of workers to jobs minimizing total cost (flow + cost).
 33. Given a graph, compute the minimum cost of K disjoint paths from source to sink.
 34. Build a schedule to maximize throughput while minimizing resource cost using MCMF.
 35. Model a transportation network and find the cheapest way to transport fixed goods.
-

☐ **Cycle Detection and Contraction**

36. Detect all simple cycles in a directed graph.
 37. Contract all strongly connected components into single nodes and return resulting DAG.
 38. In a graph, find the minimal set of nodes to remove to make it acyclic.
 39. Detect and remove negative-weight cycles from a graph.
 40. Compute the minimal cost of contracting a graph with cycle components into a tree.
-

☐ **Directed Minimum Spanning Tree (Chu-Liu/Edmonds Algorithm)**

41. Given a directed graph with root r , find the minimum spanning arborescence rooted at r .
42. Find the minimal cost of connecting all nodes from a fixed root with directed edges.

43. In a directed multigraph, select edges to form a minimum-cost rooted tree with reachability.
 44. Compute the minimum-cost strongly connected subgraph using Edmonds' algorithm.
 45. Optimize a one-way road network by selecting edges to minimize maintenance cost while maintaining reachability.
-

Let me know if you'd like **Codeforces links**, editorial breakdowns, or **C++/Python templates** for algorithms like **Chu-Liu/Edmonds**, **Kirchhoff's Matrix**, **2-SAT**, or **Hierholzer's algorithm**.

You said:

Number Theory & Math • Nimber / Sprague-Grundy Theory (Game Theory) • Continued Fractions • Berlekamp–Massey Algorithm • Gaussian Elimination in modular arithmetic • FFT and NTT (Fast Fourier Transform / Number Theoretic Transform) • Burnside's Lemma / Polya Enumeration • Chinese Remainder Theorem (advanced use cases) • Discrete Logarithm • Modular Inverse via Extended Euclid/Fermat • Mobius Function, Inclusion-Exclusion

ChatGPT said:

Here are **30 one-line Codeforces-style problem statements** based on **advanced Number Theory and Math**, including **Game Theory (Nimbers, Sprague-Grundy)**, **Continued Fractions**, **Berlekamp–Massey**, **Gaussian Elimination**, **FFT/NTT**, **Burnside's Lemma**, **CRT**, **Discrete Log**, **Modular Inverse**, and **Mobius/Inclusion-Exclusion**.

□ **Nimber / Sprague-Grundy Theory (Game Theory)**

1. Given N piles with various rules per pile, compute if the first player has a winning strategy.
 2. For a DAG game, compute Grundy numbers for each node and determine the winner.
 3. Given a game with custom move rules, find the Grundy number for initial state.
 4. Two players alternate deleting nodes from a tree – who wins?
 5. Determine the winner in a composite game made of multiple Nim-like subgames.
-

□ **Continued Fractions**

6. Find the best rational approximation to a real number with denominator $\leq K$.
 7. Given a rational number, convert it into a simple continued fraction representation.
 8. Using continued fractions, find the closest convergent under a custom constraint.
 9. Compute floor of a/b and represent its expansion as continued fraction.
 10. Solve Pell's Equation using continued fraction expansion of \sqrt{N} .
-

□ Berlekamp–Massey Algorithm

11. Given a linear recurrence of length N , predict the M -th term in the sequence modulo P .
 12. Find the minimal linear recurrence that generates a given sequence.
 13. Given partial sequence outputs of a hidden generator, reconstruct the recurrence relation.
 14. Use Berlekamp-Massey to compute the n th Fibonacci number mod p efficiently.
 15. Detect the period of a linear feedback shift register (LFSR)-like sequence.
-

□ Gaussian Elimination (Modular)

16. Solve a system of N modular linear equations modulo a prime P .
 17. Count number of solutions to $Ax = b$ over $GF(p)$.
 18. Find the inverse of a matrix modulo a prime using Gauss-Jordan elimination.
 19. Given a matrix, determine rank modulo a prime.
 20. Solve for vector x in $Ax = b$ where A is over integers mod M (non-prime, use CRT where needed).
-

□ FFT and NTT (Polynomial Multiplication)

21. Multiply two large polynomials modulo 998244353 using NTT.
 22. Given binary strings, count number of (i, j) such that convolution gives target pattern.
 23. Find number of pairs of integers whose sum is k using polynomial multiplication.
 24. Use FFT to compute the number of ways to sum k using elements from a multiset.
 25. Raise a polynomial to a large power modulo a prime using NTT.
-

□ Burnside's Lemma / Polya Enumeration

26. Count number of distinct necklaces of length N using K colors up to rotation.
 27. Given N beads and K colors, count how many distinct bracelets can be formed (rotation + reflection).
 28. Count colorings of a graph under symmetries using Burnside's Lemma.
 29. Count the number of non-isomorphic colorings of a regular polygon using Polya Enumeration.
 30. Compute the number of unique cube colorings with K colors.
-

□ Chinese Remainder Theorem (Advanced Use Cases)

31. Solve a system of congruences where moduli may not be coprime.
32. Find smallest x such that $x \equiv a_i \pmod{m_i}$ for multiple i (with non-coprime m_i).

33. Compute n th term of a sequence mod M by solving terms modulo co-prime moduli and recombining.
 34. Use CRT to merge modular results from multiple FFTs into one result mod arbitrary large M .
 35. Combine solutions of different modular equations where moduli share factors.
-

□ Discrete Logarithm

36. Solve for x in $a^x \equiv b \pmod{m}$ using Baby-Step Giant-Step.
 37. Find the smallest k such that $a^k \equiv b \pmod{m}$ for non-prime modulus.
 38. Given $x = a^y \pmod{p}$, compute y using discrete log techniques (modulo safe primes).
 39. Use Pohlig–Hellman algorithm to solve discrete log mod prime power.
 40. Compute primitive root of a prime and solve x such that $g^x = y \pmod{p}$.
-

□ Modular Inverse (Extended Euclid / Fermat)

41. Compute modular inverse of A modulo M , where M is not necessarily prime.
 42. Given array $A[1..N]$, compute product of inverses modulo prime P .
 43. Use modular inverse to solve linear congruence equations.
 44. Compute $C(n, r) \pmod{p}$ using Fermat's Little Theorem and inverse factorials.
 45. For non-prime M , compute inverse of $A \pmod{M}$ using Extended Euclidean algorithm.
-

□ Mobius Function, Inclusion-Exclusion

46. Count integers $\leq N$ that are coprime to K using Möbius function.
 47. Find number of integers $\leq N$ that are divisible by any of given set of primes using Inclusion-Exclusion.
 48. Compute the sum of $\phi(n)$ for $1 \leq n \leq N$ using Möbius inversion.
 49. Count pairs (a, b) such that $\gcd(a, b) = 1$ for $1 \leq a, b \leq N$.
 50. Given a multiplicative function $f(n)$, compute $\sum f(d)$ over all divisors d of n using Möbius transform.
-

Let me know if you'd like:

- **Implementation templates** (e.g., Berlekamp–Massey, Discrete Log)
- **Codeforces/AtCoder problems** matching these
- **A practice sheet with difficulty tags (Div1/2)** for these topics.

You said:

Strings and Suffix Structures • Suffix Automaton (SAM) • Suffix Tree • Z Algorithm, KMP • Aho-Corasick Automaton • Manacher's Algorithm • Palindromic Tree / Eertree • Duval's Algorithm (Lyndon factorization)

ChatGPT said:

Here are **30 one-line Codeforces-style problem statements** based on advanced **String Algorithms & Suffix Structures**, including **Suffix Automaton (SAM)**, **Suffix Tree**, **Z-algorithm**, **KMP**, **Aho-Corasick**, **Manacher's Algorithm**, **Palindromic Tree (Eertree)**, and **Duval's Algorithm (Lyndon Factorization)**.

□ Suffix Automaton (SAM)

1. Given string S , find the number of distinct substrings using SAM.
 2. Count how many times each substring of S appears as a substring using SAM + DP.
 3. Find the longest common substring of S and T using SAM built on one and walked on the other.
 4. Count the number of different substrings of length K in string S using SAM.
 5. Given a set of forbidden substrings, compute the longest substring of S that avoids all of them using SAM traversal.
-

□ Suffix Tree

6. Build a suffix tree for S and find the number of distinct substrings.
 7. Find the number of times a pattern P appears in string S using its suffix tree.
 8. Compute the longest repeated substring in S using suffix tree LCA traversal.
 9. Given string S , find the minimal lexicographic rotation using suffix tree suffix links.
 10. Count how many times each prefix of S appears as a substring.
-

□ Z-Algorithm, KMP

11. For each prefix of S , compute the length of the longest substring starting at that position that is also a prefix.
 12. Count the number of occurrences of pattern P in text T using Z-algorithm.
 13. Given string S , find the minimal period of S using KMP prefix function.
 14. Preprocess S using KMP and answer multiple pattern match queries in $O(m + n)$.
 15. Use KMP to count number of times a string occurs as a suffix and a prefix simultaneously.
-

□ Aho-Corasick Automaton (Multi-pattern Matching)

16. Given a text T and set of patterns $\{P_1, \dots, P_k\}$, report all patterns that occur in T .

17. Count how many times each pattern from the set occurs in text T using Aho-Corasick.
 18. Given a virus signature database, detect all occurrences in a network log.
 19. Compute the number of unique patterns matched as substrings of T using Aho-Corasick.
 20. Build a trie of patterns and use failure links to compute match frequency in T .
-

☐ **Manacher's Algorithm**

21. Given string S , find the longest palindromic substring.
 22. Count the total number of palindromic substrings in S in linear time.
 23. Find the length of the longest even-length palindrome in a string.
 24. For every position i , find the length of the longest palindrome centered at i .
 25. Determine if a string can be split into k palindromes in $O(n)$.
-

☐ **Palindromic Tree / Eertree**

26. Count all distinct palindromic substrings of a string using Eertree.
 27. Build a palindromic tree for S and find the most frequent palindromic substring.
 28. For each prefix of S , count how many distinct palindromes it contains.
 29. Find the longest palindromic suffix of each prefix of a string.
 30. Compute the minimal number of palindromic substrings that partition S .
-

☐ **Duval's Algorithm (Lyndon Factorization)**

31. Find the Lyndon factorization of string S and output all factors in order.
 32. Given string S , find its minimal lexicographic rotation using Duval's algorithm.
 33. Count the number of different Lyndon words in the prefix of S .
 34. For every suffix of S , determine whether it's a Lyndon word.
 35. Decompose a repeated string into its Lyndon factors and analyze periodicity.
-

Let me know if you'd like:

- ★ Codeforces/AtCoder problem links
- ⚙ Implementation templates (e.g., SAM in C++/Python, Eertree)
- ☐ Topic-wise difficulty ladder or practice sheet
- 📖 Explanations and visuals for any of the algorithms listed above.

You said:

Combinatorics & Probability • Advanced Inclusion-Exclusion • Stirling Numbers / Bell Numbers • Expected Value Computation under Constraints • Multinomial Coefficients • DP + Combinatorics Hybrid Problems

ChatGPT said:

Here are **30 one-line Codeforces-style problem statements** based on **Combinatorics & Probability**, including **Advanced Inclusion-Exclusion**, **Stirling/Bell Numbers**, **Expected Value under Constraints**, **Multinomial Coefficients**, and **DP + Combinatorics Hybrid Problems**.

□ **Advanced Inclusion-Exclusion**

1. Count integers from 1 to N divisible by any of a given set of primes.
 2. Count the number of surjective functions from a set of size N to K elements.
 3. Find the number of permutations with at least one fixed point using inclusion-exclusion.
 4. Given M forbidden substrings, count the number of strings of length N that avoid them.
 5. Count the number of integer arrays of size N with elements in $[1, K]$ having no two equal elements.
-

□ **Stirling Numbers / Bell Numbers**

6. Compute the number of ways to partition a set of N elements into exactly K non-empty subsets (Stirling II).
 7. Find the total number of partitions of a set with N elements (Bell Number).
 8. Count the number of surjective mappings from a set of size N to a set of size K (Stirling + factorial).
 9. For a given N , compute the number of ways to divide it into unordered subsets.
 10. Compute the sum of Stirling numbers of the second kind for all $k = 1$ to N .
-

□ **Expected Value Computation under Constraints**

11. A coin with P probability of heads is tossed until first head — find expected number of tosses.
12. Find the expected number of distinct values in an array of length N filled with integers from 1 to K .
13. Roll a die until the sum exceeds T — compute expected number of rolls.
14. From a deck of N distinct cards, compute expected number of draws to get first repeated card.
15. Compute the expected length of the longest increasing subsequence in a random permutation of N elements.

□ Multinomial Coefficients

16. Count number of ways to distribute N identical balls into K distinct boxes with constraints on each box.
17. Find the number of ways to rearrange a multiset of characters in a string (e.g., "aabbcc").
18. Compute the number of ways to reach (x, y, z) from origin using steps in 3D grid.
19. Count the number of words of length N made from K different letters, each used a specified number of times.
20. Compute multinomial coefficient for given frequencies of elements and return it modulo prime.

□ DP + Combinatorics Hybrid Problems

21. Count the number of ways to color a grid of size $N \times M$ with K colors such that no two adjacent cells share the same color.
22. Compute the number of valid bracket sequences of length $2N$ using DP + Catalan Numbers.
23. Given a target sum and a set of coins, count the number of combinations to reach that sum (unbounded).
24. Count the number of integer arrays of length N whose sum is exactly K and each element is $\leq X$ using DP.
25. Find the number of partitions of N into at most K parts using combinatorics and DP.
26. Count ways to fill an array such that sum is even and no two adjacent elements are equal.
27. Count number of paths from $(0, 0)$ to (n, m) avoiding obstacles using DP and combinatorics.
28. Compute number of binary strings of length N with no two consecutive ones.
29. For a grid, count the number of paths such that the number of turns is $\leq T$.
30. Given constraints on frequency of characters, compute number of valid strings of length N using DP.

Let me know if you'd like:

- □ Full explanations or derivations for Stirling/Bell/Multinomial formulas
- ↔ Specific problem links from **Codeforces**, **AtCoder**, or **CSES**
- 📄 Templates or modular arithmetic utilities (e.g., factorials mod prime)
- 📋 A curated practice sheet with levels and tags (Div1/2, topic-focused)