Contest Editorial
CoU CSE Fest 2025 - Inter University Programming Contest (Divisional)

Programming Club of CSE, Comilla University

Practice contest link: `Codeforces Gym 106057`
Main contest link: `CoU IUPC 2025 Divisional`

Problem A — Decreasing Trees

Author: Mehedi Hasan Arafat (Codeforces profile)

Difficulty: Easy–Medium **Tags:** Trees, Counting, Factorials

Problem Statement. Count the number of rooted labeled trees on n vertices with root 1, where labels strictly increase along any path from the root to a node.

Solution: Hints and Insights

- The root is always node 1.
- For each node $i > 1$, its parent must be chosen from $\{1, \ldots, i-1\}$.
- These choices are independent and form a valid rooted tree: parent pointers decrease labels and end at 1, ensuring no cycles and connectivity.
- The total number of such trees is:

$$\prod_{i=2}^{n}(i-1) = (n-1)!.$$

Solution: Implementation

- Compute $(n-1)!$ for each test case.
- For $n \leq 20$, the result fits in a 64-bit integer.

Complexity: Time: $O(n)$ per test case. Space: $O(1)$.

Solution: Reference Code

- `Factorial-based solution`

Problem B — Dartboard

Author: Mehedi Hasan Arafat (Codeforces profile)

Difficulty: Medium **Tags:** Geometry, Polygon Area, Probability, Expected Value, Modular Arithmetic

Problem Statement. Given N concentric strictly nested convex polygons (1 = innermost), each with an integer score S_i, a dart landing in multiple polygons scores the innermost one. Alice throws M darts: K precision darts land uniformly in a chosen polygon t, and $M-K$ darts land uniformly in the outermost polygon N. Compute the maximum expected total score modulo 10^9+7 as a fraction $P \cdot Q^{-1}$.

Solution: Key Observations

- Define $\text{Area}(i)$ as the geometric area of polygon i. The ring area is:

$$R_j = \text{Area}(j) - \text{Area}(j-1), \quad \text{with Area}(0) = 0.$$

- For a dart thrown in polygon i, the probability of landing in ring $j \le i$ is $R_j/\text{Area}(i)$. The expected score is:

$$E_i = \sum_{j=1}^{i} \frac{R_j}{\text{Area}(i)} S_j = \frac{\sum_{j=1}^{i} R_j S_j}{\text{Area}(i)}.$$

- Normal darts (in polygon N) have expected score:

$$E_N = \frac{\sum_{j=1}^{N} R_j S_j}{\text{Area}(N)}.$$

- The optimal total expected score is:

$$K \cdot \max_{1 \le i \le N} E_i + (M-K) \cdot E_N.$$

Solution: Computing Areas

- Use the shoelace formula for polygon i with vertices (x_t, y_t):

$$A_i^{\text{raw}} = \sum_{t=0}^{v_i-1} (x_t y_{t+1} - x_{t+1} y_t), \quad \text{area} = |A_i^{\text{raw}}|/2.$$

- Use doubled areas to avoid floating-point issues:

$$\text{Area2}(i) = |A_i^{\text{raw}}|, \quad R_j^{(2)} = \text{Area2}(j) - \text{Area2}(j-1).$$

- Then, $E_i = \frac{\sum_{j=1}^{i} R_j^{(2)} S_j}{\text{Area2}(i)}$.

Solution: Maximizing Expected Score

- Compute for each i:

$$\text{num}_i = \sum_{j=1}^{i} R_j^{(2)} S_j, \quad \text{den}_i = \text{Area2}(i).$$

- Find $i^* = \arg\max_i(\text{num}_i/\text{den}_i)$ using cross-multiplication:

$$E_a > E_b \iff \text{num}_a \cdot \text{den}_b > \text{num}_b \cdot \text{den}_a.$$

- Use 128-bit integers for products to avoid overflow.

Solution: Final Answer

- Compute the total expected score:

$$T = K \cdot \frac{\text{num}^*}{\text{den}^*} + (M - K) \cdot \frac{\text{num}_N}{\text{den}_N}.$$

- Compute modulo $M_0 = 10^9 + 7$ using modular inverses:

  $\text{ans} = (K \cdot (\text{num}^* \bmod M_0) \cdot \text{inv}(\text{den}^* \bmod M_0) + (M - K) \cdot (\text{num}_N \bmod M_0) \cdot \text{inv}(\text{den}_N \bmod M_0)) \bmod M_0$,

  where $\text{inv}(x) = x^{M_0 - 2} \bmod M_0$.

Solution: Algorithm Summary

1. Read N, M, K and scores $S_1, \ldots, S_N$.
2. For each polygon i, compute $\text{Area2}(i)$ using the shoelace formula.
3. Compute ring areas $R_j^{(2)} = \text{Area2}(j) - \text{Area2}(j-1)$.
4. Compute $\text{num}_i = \sum_{j=1}^{i} R_j^{(2)} S_j$ and $\text{den}_i = \text{Area2}(i)$.
5. Find i^* maximizing $\text{num}_i / \text{den}_i$ via cross-multiplication.
6. Compute the final answer using modular arithmetic.

Complexity: Time: $O(\sum v_i + N + \log M_0)$. Space: $O(N)$.

Solution: Implementation Notes

- Use 64-bit integers for shoelace sums and 128-bit for cross-multiplication.
- Ensure $R_j^{(2)} > 0$ (guaranteed by strict nesting).
- Avoid floating-point arithmetic; use integer-based computations.

Solution: Reference Code

- `Integer-based modular solution`

Problem C — Prime Dominion

Author: Md Raihanul Islam (Codeforces profile)

Difficulty: Medium **Tags:** GCD, Sparse Table, Binary Search, Dynamic Programming

Problem Statement. Given an array $a_1, a_2, \ldots, a_n$, find the maximum length of a subarray whose GCD is a prime number, or return -1 if none exists.

Solution: Key Observations

- The GCD of a subarray decreases or stays the same when extended.
- Once the GCD becomes 1, it cannot become prime again.
- Precompute primes up to $2 \cdot 10^5$ using a sieve.

Solution: Approach 1: Sparse Table + Binary Search

- Build a sparse table for range GCD queries in $O(n \log n)$.
- For each index i, binary search the farthest index j such that $\gcd(a_i, \ldots, a_j) > 1$.
- Check if the GCD is prime and update the maximum length.

Complexity: Time: $O(n \log^2 n)$. Space: $O(n \log n)$.

Solution: Approach 2: Logarithmic Subarray Aggregator (LSA Trick)

- Normally, you would use a sparse table and binary search, leveraging that there are $O(\log(\text{MAX}))$ different GCDs ending at each index. However, a more elegant approach exists with minimal code.
- Store a vector or map $v[r]$ containing all GCDs of subarrays ending at r. There are $O(\log(\text{MAX}))$ different GCDs at each index.
- Iterate over all elements in $v[r]$ to compute $v[r+1]$, effectively performing a brute-force in $O(\log n)$ time per index.
- This approach also works for subarray AND, OR, etc.
- Update the answer if a prime GCD is found.

Complexity: Time: $O(n \log A)$, where A is the maximum element. Space: $O(n)$.

Remarks: Notes

- The LSA Trick is faster in practice due to fewer logarithmic factors and simpler code.
- Handle edge cases: return -1 if no valid subarray exists.

Solution: Reference Code

- `Sparse Table solution`
- `LCA Trick solution`

Problem D — Zero is not an Option!

Author: Md Raihanul Islam (Codeforces profile)

Difficulty: Easy **Tags:** Bitwise Operations, Greedy

Problem Statement. Given an $N \times M$ grid of non-negative integers, determine if it's possible to choose one number from each row such that their bitwise AND is strictly greater than 0. Output "YES" or "NO".

Solution: Key Idea

- The bitwise AND is positive if there exists a bit set in at least one chosen number from each row.
- Compute the bitwise OR of each row to get a mask of all possible bits.
- Take the bitwise AND of all row masks. If non-zero, a common bit exists, so output "YES"; otherwise, "NO".

Solution: Algorithm

1. For each row, compute the bitwise OR of all numbers.
2. Compute the bitwise AND of all row OR-masks.
3. Output "YES" if the result is non-zero, else "NO".

Complexity: Time: $O(\sum NM)$. Space: $O(N)$.

Solution: Alternative Approach

- Check each bit position (up to 47 bits for values up to 10^{14}). Verify if every row has a number with that bit set. If any bit satisfies this, output "YES".
- Time: $O(47 \cdot N)$.

Solution: Reference Code

- `Bitwise OR-AND solution`

Problem E — Treasures in the Interval

Author: Mahabub Rahman (Codeforces profile)

Difficulty: Medium **Tags:** Difference Array, Prefix Sum, Sorting

Problem Statement. Given an array of length N and Q range updates (L, R, d) that add d to elements $A[L \dots R]$, apply all updates and answer M queries for the K-th largest element in the final array.

Solution: Approach

1. Initialize a difference array df of size $N + 1$ with zeros.
2. For each update (L, R, d) (1-indexed):

$$df[L] += d, \quad df[R+1] -= d.$$

3. Compute prefix sums of df to get increments, and apply to array A.
4. Sort A in ascending order.
5. For each query K, output $A[N-K]$ (0-based indexing).

Complexity: Time: $O(N \log N + Q + M)$. Space: $O(N)$.

Solution: Correctness

- The difference array ensures each update affects exactly $[L, R]$.
- Sorting allows $O(1)$ query answers for the K-th largest element.

Solution: Reference Code

- `Difference array and sorting solution`

Problem F — A Perfect Path

Author: Mahabub Rahman (Codeforces profile)

Difficulty: Hard **Tags:** Tree, LCA, Hashing, Perfect Square

Problem Statement. Given a tree with node values, for each query (u, v), determine if the product of values on the path from u to v is a perfect square.

Solution: Key Observations

- If any node value is 0, the product is 0 (a perfect square).
- If the number of negative values on the path is odd, the product is negative (not a perfect square).
- For positive products, the product is a perfect square if the XOR of squarefree kernels (primes with odd exponents) is empty.

Solution: Approach (Binary Lifting + Prefix-XOR)

- Precompute smallest prime factors up to $2 \cdot 10^5$.
- Assign random hash values to primes; compute the hash of each node's squarefree kernel.
- Use DFS to maintain prefix XORs of hashes and counts of zeros/negatives for each node.
- For each query (u, v):
  - Compute LCA $w = \text{LCA}(u, v)$.
  - Path XOR = `pref[u]` $\oplus$ `pref[v]` $\oplus$ `pref[w]` $\oplus$ `pref[parent(w)]`.
  - Count negatives and zeros along the path.
  - The path is perfect-square if there is any zero or if XOR=0 and negatives are even.

Complexity: Time: $O(\log n)$ per query due to LCA binary lifting. Preprocessing: $O(n \log n + M \log \log M)$. Space: $O(n \log n)$.

Solution: Alternative Approach (Heavy-Light Decomposition)

- Heavy-Light Decomposition splits the tree into chains.
- Use a segment tree on the HLD array to store for each segment:
  - XOR of squarefree-hashes,
  - Count of negatives,
  - Count of zeros.
- For a query (u, v), split the path into at most $O(\log n)$ chains and merge segment tree results.
- The path is perfect-square if XOR=0 and negatives even, or if any zero is present.

Complexity: Preprocessing: $O(n + M \log \log M + \text{total\_factor\_cost})$ (where $M = 2 \cdot 10^5$).
Query: $O(\log^2 n)$ per query due to at most $O(\log n)$ segments, each queried in $O(\log n)$.
Space: $O(n)$ for segment tree + HLD arrays.

Solution: Reference Code

- `Prefix XOR with LCA`
- `Heavy-Light Decomposition Alternative (Author: Abdullah500)`

Problem G — MeX is not Max

Author: Mestu Paul (Codeforces profile)

Difficulty: Easy **Tags:** MEX, Array

Problem Statement. Find the MEX (smallest non-negative integer not in the array) of a sequence of length N.

Solution: Approach

- Use a boolean array or hash set to mark presence of numbers in $[0, N]$.
- Scan from 0 to N and return the first absent number.

Complexity: Time: $O(N)$ per test case. Space: $O(N)$.

Solution: Reference Code

- `MEX solution`

Problem H — Mr. Benzene's Bachelor Trip

Author: Tuhin Ahmed (Codeforces profile)

Difficulty: Medium **Tags:** Combinatorics, Binary Search, Stars and Bars

Problem Statement. Find the smallest non-negative integer n such that the number of ways to express n as a sum of exactly k non-negative integers is at least m, or return -1 if impossible.

Solution: Key Idea

- The number of ways is given by the stars-and-bars formula:

$$\binom{n+k-1}{k-1}.$$

- Binary search the smallest $n \geq 0$ with $\binom{n+k-1}{k-1} \geq m$.

Solution: Approach

- Binary search over $n \in [0, 10^{18}]$.
- Compute $\binom{n+k-1}{k-1}$ using the multiplicative formula:

$$\prod_{i=1}^{r} \frac{N-i+1}{i} \quad \text{with } N = n+k-1,\ r = k-1,$$

  and stop early once the running value $\geq m$.
- Use a 128-bit integer type (e.g. `__int128`) to avoid overflow when multiplying, and perform the early-stop check after every multiplication/division to keep numbers small.
- Edge cases: if $m = 1$ the answer is 0. If $k = 1$ then there is exactly one way to write any n (so if $m > 1$ the answer is -1).

Complexity: Time: $O\big(T \cdot \log(10^{18}) \cdot \min(k, n)\big)$ (early stop usually makes it much faster). Space: $O(1)$.

Solution: Reference Code

- `Binary search with stars and bars`

Problem I — AnapalinDrome

Author: Md. Rakib Hossain (Codeforces profile)

Difficulty: Hard **Tags:** Dynamic Programming, Bitmasking, String Partitioning

Problem Summary

We are given a string s. A substring is an **AnapalinDrome** if its letters can be rearranged into a palindrome. We need to count all partitions of s into such substrings and output the **sum of scores** of all partitions (score = number of substrings). Answer modulo $10^9 + 7$.

Key Observation

A string can be rearranged into a palindrome **iff at most one character has odd frequency**.

Thus, we only need to track the **parity of frequencies** (even/odd), not exact counts.

Step 1 – Represent substrings by masks

- Assign each character a random 64-bit number $h[c]$.
- Define prefix XOR:

$$pre[i] = h[s_0] \oplus h[s_1] \oplus \cdots \oplus h[s_{i-1}]$$

- For substring $s[l..r]$:

$$pre[r+1] \oplus pre[l]$$

  encodes its parity pattern.

The substring is valid iff this value is either 0 (all even) or equal to some $h[c]$ (exactly one odd). Therefore, we only need to check against 27 masks: $\{0, h[a], h[b], \ldots, h[z]\}$.

Step 2 – DP definition

Let

$$dp[i] = (\text{count}, \text{score})$$

where:

- count = number of valid partitions of the suffix starting at i,
- score = total sum of scores of those partitions.

Base case:

$$dp[n] = (1, 0)$$

(empty suffix).

Transition: if $s[i..j]$ is valid,

$$dp[i].\text{count} \mathrel{+}= dp[j+1].\text{count}$$

$$dp[i].\text{score} \mathrel{+}= dp[j+1].\text{score} + dp[j+1].\text{count}$$

(The extra $dp[j+1].\text{count}$ accounts for the additional substring added at this step.)

Step 3 – Efficient transition

- Maintain a map $m[\text{mask}] = (\text{count}, \text{score})$ storing results of suffixes.
- Iterate i from right to left:
  1. For each candidate mask (27 options), compute $nxt = pre[i] \oplus mask$.
  2. If nxt exists in m, update $dp[i]$ using $m[nxt]$.
  3. Insert $dp[i]$ into $m[pre[i]]$.

Thus, all transitions are handled in $O(27)$ per position.

Complexity

$$O(27n) \quad \text{per test case.}$$

Since the total length across all test cases is at most 10^5, this solution is efficient.

Example

Consider $s =$ `aab`.

Valid partitions:

1. `a | a | b` $\rightarrow$ score $= 3$
2. `aa | b` $\rightarrow$ score $= 2$
3. `aab` $\rightarrow$ score $= 1$

Total $= 3 + 2 + 1 = 6$.

Implementation Notes

- Random hashing avoids collisions (practically safe).
- Modular arithmetic is applied at every DP step.

Final Answer

The result is simply:

$$\text{Answer} = dp[0].\text{score}$$

Solution: Reference Code

- `Hashing-based solution`

Problem J — Co-Primal Ancestor

Author: Md. Rakib Hossain (Codeforces profile)

Difficulty: Hard **Tags:** Tree, LCA, Inclusion-Exclusion, Persistent Segment Tree

Problem Statement. Given a tree with node values, for each query (u, v) (XOR-encrypted with the last answer), count common ancestors whose value is coprime with both a_u and a_v.

Solution: Key Observations

- Common ancestors are nodes on the path from the root to $\text{LCA}(u, v)$.
- A node x is valid if no prime dividing a_u or a_v divides a_x.
- Use inclusion-exclusion over the set of primes P dividing a_u or a_v.

Solution: Approach

- Precompute primes up to $5 \cdot 10^4$.
- For each node, compute squarefree products of its prime factors and update a persistent segment tree (PST).
- Use DFS to propagate PST versions to children.
- For each query: decode u, v, compute $\text{LCA}(u, v)$, apply inclusion-exclusion on PST.

Complexity: Time: $O(2^{|P|} \log A)$ per query, where $|P|$ is small. Space: $O(n \cdot D \log A)$.

Solution: Reference Code

- `Persistent segment tree solution`

Problem K — Dreaming of National IUPC

Author: Md. Rakib Hossain (Codeforces profile)

Difficulty: Easy **Tags:** Output Only

Problem Statement. This is a giveaway problem. Output the exact line: `We want national level IUPC in CoU`.

Solution: Approach

- Print the string with exact spelling, spacing, and capitalization.
- No input processing required.

Complexity: Time: $O(1)$. Space: $O(1)$.

Comment:
We have always dreamt of hosting a national-level IUPC at CoU. We hope this aspiration will be achieved with enthusiasm and support.

Solution: Reference Code

- `Simple output solution`