The 2026 ICPC Asia Topi Online Preliminary Programming Contest MOCK

Instructions

- Do not open the booklet unless you are explicitly told to do so. You can only read these instructions below.
- Do not create disturbance or move around unnecessarily in the arena.
- If you have any question regarding the problems, send a clarification from the judges using DOMJudge.
- There would be no internet access and mobile phones are also not allowed.
- Before submitting a run, make sure that it is executable via command line. For Java, it must be executable via "javac" and for GNU C++ via "g++". Java programmers need to remove any "package" statements and source code's file name must be the same as of the main class. C++ programmers need to remove any getch() / system("pause") like statements.
- Do not attach input files while submitting a run, only submit/attach source code files, i.e., *.java or *.cpp or *.py.
- Language supported: C/C++, Java and Python3
- Source code file names should not contain white space or special characters.
- You must take input from Console i.e.: Standard Input Stream (stdin in C, cin in C++, System.in in Java, stdin in Python)
- You must print your output to Console i.e.: Standard Output Stream (stdout in C, cout in C++, System.out in Java)
- Please, don't create/open any file for input or output.
- Please strictly meet the output format requirements as described in problem statements, because your program will be auto judged by computer. Your output will be compared with the judge's output byte-by-byte and not tolerate even a difference of single byte. So, be aware! **Pay special attention to spaces, commas, dots, newlines, decimal places, case sensitivity etc.**
- Your programs must meet the time constraints given at each problem.
- The decision of judges will be absolutely final.
- ***Plagiarism or use of Generative AI tools is strictly prohibited and will result in immediate disqualification.***

Problem 01: Treasure Trees

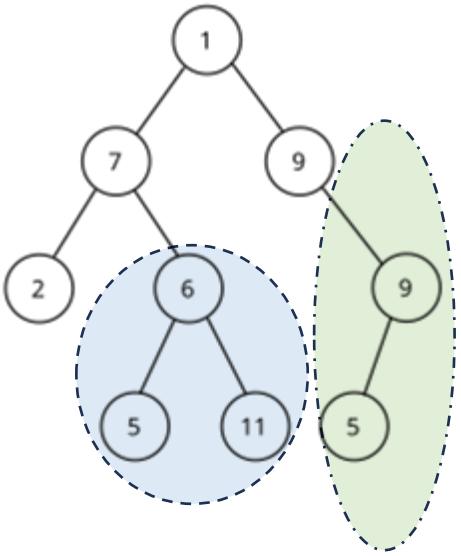
Time limit: 3 seconds

Raju and Rani are exploring a magical forest filled with enchanted trees. Each tree has numbers carved into its nodes, and some of these trees follow a very special rule—they are **Perfect Order Trees** (just like Binary Search Trees).

A **Perfect Order Tree** must obey these rules:

1. Every node on the **left side** must have numbers **strictly smaller** than the number on the parent node.
2. Every node on the **right side** must have numbers **strictly larger** than the parent node.
3. Both left and right sides must themselves also be Perfect Order Trees.

Raju discovers that each subtree hides a **treasure**, with the treasure value equal to the **sum of all numbers** in that subtree.



Rani's challenge to Raju is: *"From this entire magical tree, find the subtree that is a Perfect Order Tree and gives the **maximum treasure value**. Tell me what the highest treasure sum is!"*

So your task is the same as Raju's: **Given the tree, find the maximum sum of any subtree that satisfies the Perfect Order (BST) rules.**

Input

Node values separated by spaces. Positive values showing node values and negative values showing null/ empty child.

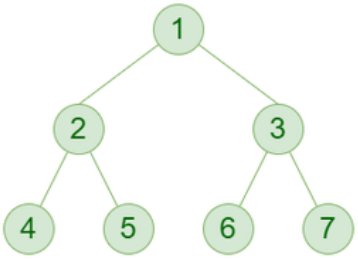
Output

Print value of valid maximum binary search tree.

Input Example 1

Input: 1 7 9 2 6 -1 9 -1 -1 5 11 5
Output: 22
Explanation: There are two binary search trees. The maximum binary search tree has sum 22.

Input Example 2



Input: 1 2 3 4 5 6 7
Output: 7
Explanation: Maximum sum in a valid Binary search tree is 7

Constraints:

The number of nodes in the tree is in the range **1 to 10^5**.
All node values are in range **-10^6 to 10^6**

Problem 02: Prompt Pit

Time limit: 1.5 seconds

Two prompt-engineers face off in the Prompt Pit, a pick-ban arena where they assemble teams of AI models from a shared model pool. Each model has a known score, and once a model is chosen or banned, it disappears from the pool forever – thrown into the deprecated weights folder.

The draft follows a fixed sequence of actions. On each turn, the active prompt engineer must perform exactly one action:

1. Pick a model
2. Ban a model

If an engineer misses a pick, the system automatically assigns them a random model. If they miss a ban, nothing happens.

Each engineer tries to maximize the final score difference in their favour.

Given all model strengths and the full pick/ban sequence, determine which engineer (**A** or **B**) has the advantage at the end of the draft, and by how much.

It is guaranteed that there are no ties.

Input:

The first line contains a single integer **N (2<=N<=100)** – the number of models in the model pool.

The second line contains **N** integers $a_1, a_2, a_3, \ldots, a_N$ $(1 \leq a_i \leq 10^8)$ – the score of all the models.

The third line contains a single integer M $(2 \leq M \leq min(N, 20))$ – the number of actions the prompt engineers must perform.

Next M lines look like **"ENGINEER ACTION"**, where engineer is denoted by the letters **A** or **B**, and action is either a **"pick"** or **"ban"**

It is guaranteed that both engineers make at least **1** pick, and both have the same number of picks and same number of bans

Output:

Print the engineer (**A** or **B**) which wins and the absolute difference in score if both perform actions optimally.

Sample Input 1	**Sample Output 1**
5 3 7 2 9 5 4 A pick B pick A ban B ban	A 2

Sample Input 2	Sample Output 2
6 10 1 9 8 3 2 6 A ban B ban A pick B pick A pick B pick	A 3

Problem 03: Gossip Chain

Time limit: 5 seconds

In a small town, gossip spreads fast. Whenever one friend talks to another, gossip travels along that direction. Each record of gossip is written as: $u\ v\ t$ meaning that friend u told friend v something at time t.
Over time, the same people may exchange information many times in different directions. We would like to understand *how gossip typically flows among groups of three friends* within a short period of time.
Every gossip exchange is a directed message **(u → v)** occurring at a certain time. We are interested in three-message gossip bursts involving exactly three distinct friends **(A, B, C)**, with strictly increasing times ***t1 < t2 < t3***, and where all three messages happen within **Δ minutes**, i.e. $t3 - t1 \leq \Delta$.

The following three gossip patterns are of interest:

1. **A → B, B → C, C → A**
2. **A → B, A → C, B → C**
3. **A → B, C → B, B → A**

Given all gossip events recorded for a single day, count how many distinct three-message sequences correspond to **Type 1, Type 2, and Type 3** gossip patterns.

Input:

The first line contains three integers:
N M Δ

- $N\ (1 \leq N \leq 5000)$**:** number of friends
- $M\ (1 \leq M \leq 50000)$**:** number of gossip events
- $\Delta\ (1 \leq \Delta \leq 5000)$**:** maximum time window (in minutes)

Each of the next **M** lines contains three space-separated integers: $u\ v\ t$, indicating that friend **u** told friend **v** something at time $t\ (0 \leq u, v < N, u \neq v, 1 \leq t \leq 10^9)$. All gossip events are distinct.

Output:

Print three integers separated by spaces:
CountType1 CountType2 CountType3

Sample Input	Sample Output
3 12 2 0 1 1 1 2 2 2 0 3 0 1 6 0 2 7 1 2 8 0 1 11 2 1 12 1 0 13 0 1 16 1 2 17 2 0 18	2 1 1

Problem 04: Crop Types Along the Mighty River Indus

Time limit: 2 seconds

The Indus River is the seventh-longest river in Asia, with a length of approximately 3,180 km (1,980 mi). Along River Indus, which is by far the largest river in Pakistan, the seventh largest in Asia and among the 50 largest rivers by annual flow, the farmlands are arranged in a straight line. Each farm grows exactly one type of crop. The crops are represented by integers, where different integers correspond to different crop types.
The Government of Khyber Pakhtunkhwa wants to estimate the number and type of crops which are planted by their farmers so that they can plan for import/export of food items in the next cycle. They are particularly interested in identifying the dominant crop, and have sought help from you.
You are given the crop type of each farm in order from north to south of the mighty river Indus. A crop type x is said to be dominant in a segment of consecutive farms **[*L, R*]** if x appears strictly more than half of the farms in that segment.

Your task is to process several queries. Each query gives a range of farms **[*L, R*]**, and you must determine whether there is any crop type that is dominant in that segment. If such a crop exists, output its type. If no crop is dominant, output 0.

Input:

The input consists of:

- A line containing two space-separated integers $N$and Q $(1 \leq N, Q \leq 200{,}000)$, the number of farms and number of queries.
- A line containing N integers $c_1, c_2, \ldots, c_N$ representing the crop type of each farm. Crop types satisfy $| c_i | \leq 10^9$.
- Q lines follow. Each line contains two integers L and R $(1 <= L, R <= N)$, describing a query segment.

Output:

For each query **[L, R],** output a single integer:

- the type of the dominant crop in that segment, if it exists; or
- **0** if no crop type is dominant.

Sample Input	Sample Output
7 5 1 2 2 2 3 2 1 1 7 2 6 3 5 1 3 5 7	2 2 2 2 0

Sample Explanation:

- In the segment [1,7], the crop type *2* appears 4 times, which is more than half of 7.
- In the segment [5,7] = [3,2,1], no crop appears more than once; length is 3, so no crop is dominant.

Problem 05: Karachi Food Festival

Time limit: 2 seconds

The annual Karachi Food Festival has ***N*** unique food stalls along the promenade.
Each stall, ***i***, has a distinct *flavour rating* (an integer between **1** and ***M***, inclusive) that reflects how popular its signature dish is.

You are asked to assign to each stall a positive integer number of *ticket-packs* ***t[i]*** (each pack is an identical block of customers) to send to that stall for the opening day.
Each ***t[i]*** must be at most ***M***.

Your goal is to choose ticket-packs so that:

1. The overall **weighted average rating** of the stalls (weighted by their ticket-packs) is an integer, i.e.

$$\frac{\sum_{i=1}^{N} r[i] \cdot t[i]}{\sum_{i=1}^{N} t[i]}$$

2. No three stalls in **increasing** stall-index order may receive exactly the same number of ticketpacks, that is, there must **not** exist indices $i < j < k$ with $t[i] = t[j] = t[k]$.
3. Every ***t[i]*** is an integer satisfying $1 \le t[i] \le M$.

If such an assignment is impossible, output **-1**.

Input:

The input consists of multiple test cases.

- The first line contains an integer $T\ (1 \le T \le 1000)$ which is the number of test cases.
- For each test case:
  - The first line contains two space-separated integers $N(1 \le N \le 10^6)$ and $M(1 \le M \le 10^9)$.
  - The second line contains ***N*** distinct integers $r[1], r[2], \ldots, r[N]\ (1 \le r[i] \le M)$ denoting the flavour ratings.
- It is guaranteed that the sum of ***N*** across all test cases is at most 10^6.

Output:

For each test case, output one line:

- If an assignment exists, print ***N*** space-separated integers $t[1], t[2], \cdots, t[N]\ (1 \le t[i] \le M)$ representing a valid ticket-pack assignment (any valid answer will be accepted).
- Otherwise, print **-1**.

Note: Sample test case is on the next page.

Sample Input	Sample Output
3	2 1 2

3 6 4 1 3 2 2 1 2 4 100000 1 2 5 9	-1 1 1 3 4

Explanation of Samples:
Case 1:
r = [4, 1, 3], t = [2, 1, 2]
Weighted sum = $4 \cdot 2 + 1 \cdot 2 + 3 \cdot 2 = 15$
Total packs = 5
Weighted Average = 15/5 = 3 (an integer). No triple of equal ***t*** exists.

Case 2:

With ***N = 2***, ***M = 2,*** and ***r = [1, 2]***, none of the possible combinations of ***t[i]*** $\in$ ***{1, 2}*** make the average an integer, hence **-1**.

Case 3:
r = [1, 2, 5, 9], t = [1, 1, 3, 4]
Weighted sum = $1 \cdot 1 + 2 \cdot 1 + 5 \cdot 3 + 9 \cdot 4 = 54$
Total packs = 9
Weighted Average = 54/9 = 6 (an integer). No triple of equal ***t*** exists.

Problem 06: Counting Problem

Time limit: 1.5 seconds

You are given **N boxes**. The **i-th** box contains:

- $\mathbf{A_i}$ items of type **A**,
- $\mathbf{B_i}$ items of type **B**.

All items of type **A** are identical to each other. All items of type **B** are identical to each other.

To form a sequence, you select **two different boxes**, take **all items** from both boxes, and then arrange all these items into **sequences**. You can do this operation with any two boxes. The relative order of items in each sequence matters.

Two results are considered different if:

- The chosen pair of boxes is different, **or**
- The ordered sequences produced are different.

Compute the total number of different results, **modulo 1,000,000,007**.

Input

The first line contains an integer N **(**$2 \leq N \leq 200{,}000$**)**, the total number of boxes. ***N*** lines follow, each of which contain 2 space-separated integers $\mathbf{A_i}$ **(**$1 \leq A_i \leq 2000$**)** and $\mathbf{B_i}$**(**$1 \leq B_i \leq 2000$**).**

Output

Print **one integer** — the total number of different results, **modulo 1,000,000,007**.

Sample Input	Sample Output
3 1 1 1 1 1 1	18

Explanation:

1 & 2 box: AABB, ABBA, ABAB, BBAA, BABA, ABBA

1 & 3 box: AABB, ABBA, ABAB, BBAA, BABA, ABBA

2 & 3 box: AABB, ABBA, ABAB, BBAA, BABA, ABBA

Total Count: 18

Problem 07: The Last Traveler of the Clockwork Kingdom

Time limit: 1 second

In the ancient Clockwork Kingdom, time itself flows through a massive circular chain of **arcane pylons**. Each pylon contains a reservoir of **temporal essence**, and traveling from one pylon to the next consumes a certain amount of that essence.

You are the kingdom's last Temporal Traveler capable of drawing essence from the pylons to sustain your journey.

But there's a catch:
You must begin your journey at **one** of these pylons with no essence at all, and attempt to complete a full rotation around the Clockwork Ring. At every pylon:

- You **absorb** the temporal essence stored there
- You **spend** a fixed amount of essence to move to the next pylon in the ring

Your task is to determine **which pylon** can serve as the starting point that allows you to complete the entire circular journey **without your essence ever dropping below zero**.

If multiple such pylons exist, output the pylon with the **lowest index**.
If no pylon allows a full rotation, you must return **-1**, meaning the Clockwork Ring cannot be traversed.

Input:

The input consists of the following:

- First line: integer **n**, the number of pylons in the Clockwork Ring $(1 \leq n \leq 100000)$
- Second line: **n integers**, where the ith integer is the **essence[i]** stored in pylon *i* *(1<=essence[i]<=10^9)*
- Third line: **n integers**, where the ith integer is the **drain[i]** (1<=drain[i]<=10^9), the essence required to move from pylon ***i*** to pylon **(i+1)** (wrapping to 1 at the end)

Output:

A single integer i.e. the **lowest index** of the pylon from which a full temporal rotation is possible, or **-1** if the journey is doomed from the start.

Sample Input	Sample Output
5 1 2 3 4 5 3 4 5 1 2	4

Sample Input	Sample Output
3 2 3 4 3 4 3	-1

Problem 08: Flood-Safe Corridors

Time limit: 1 seconds

Every monsoon season, several regions across Pakistan face severe flooding. The National Disaster Management Authority (NDMA) wants to plan robust evacuation routes along an existing road network connecting major cities and towns. Some roads are built to modern standards and can safely be used, while others are older and may become unusable during floods. New roads can also be constructed if necessary, but at a cost.

You are given an undirected graph with **n** cities (numbered **1 to n**) and **m** road segments. NDMA has identified a subset of "evacuation cities" – locations such as Karachi, Lahore, Peshawar, Quetta, and others – which must remain mutually reachable by reliable paths even if any single road fails due to flooding.

Formally, a subset **S** of cities is given. NDMA wants to choose a set of roads (some existing, some newly constructed) so that:

1. All cities in **S** lie in a single connected component.

2. Between every pair of distinct cities in **S**, there exist at least two edge-disjoint paths.

3. The total construction/upgrade cost of roads used in the final network is minimized.

Input:

The first line contains three integers **n, m,** and **s ($2 \le n \le 1100$, $1 \le m \le 2000$, $1 \le s \le n$)**: the number of cities, the number of potential road segments, and the number of evacuation cities.

The next line contains s distinct integers **e1, e2, ..., es ($1 \le ei \le n$)**, the labels of the evacuation cities that must be robustly connected.

Each of the next m lines describes a road with four integers **u, v, t,** and **c**:

u and **v ($1 \le u, v \le n, u \ne v$)** are the endpoints of the road.

t ($0 \le t \le 2$) is the type of the road:

t = 0: an existing weak road that must be upgraded at cost **c** to be considered reliable.

t = 1: an existing reliable road that can be used at no additional cost; the given **c** should be ignored.

t = 2: a new road that does not currently exist but can be constructed with cost **c**.

c ($1 \le c \le 10^9$) is the upgrade or construction cost for **t = 0** or **t = 2**. For **t = 1**, **c** is present but irrelevant.

Multiple roads between the same pair of cities may exist.
You may choose any subset of roads. The final network consists of all evacuation cities and any other cities you choose to connect with roads, as long as the three conditions above are satisfied using only reliable roads (those chosen, and, if needed, upgraded or built).

Output:

Print a single line.
If it is impossible to satisfy the given constraints print **-1**
Otherwise, print a single integer: the minimum total cost needed to upgrade/build roads so that the evacuation cities satisfy all requirements.

Sample Input 1	Sample Output 1
6 7 3 1 3 5 1 2 1 10 2 3 0 5 3 4 2 4 4 5 1 7 5 6 0 3 2 6 2 6 1 5 2 8	9

Sample Input 2	Sample Output 2
4 3 3 1 2 3 1 2 1 5 2 3 2 7 3 4 2 10	-1