UTD Programming Contest for High School Students
April 6, 2019

We will use HackerRank today.

- **Time Allowed: three hours.**
- **Each team must use only one computer - one of UTD's.**
- **Solve the problems in any order.**
- **Your programs must read from System.in and output to System.out (cin and cout for C/C++ programmers.)**
- **Do not access the web except to access programming language documentation.**
- **Do not use any recording devices containing code, other than UTD's computer.**
- **Your solutions must be entirely yours, typed by you during the time allowed for the contest.**
- **For time and memory limits, unless stated otherwise, refer to the hackerrank's environment page.**

Scoring

All questions are worth the same number of points, even though they may not be equally difficult.

In order to break ties, we will use penalty points. The penalty points for a question will be zero if the question is not answered correctly. If a correct submission occurs for a question at time T minutes, then T penalty points will be added, plus 20 penalty points for each incorrect submission for that question.

The top ranked three teams, irrespective of category (Novice or Advanced) will be deemed 1st, 2nd and 3rd place-winners of the Advanced Contest. This rule in effect promotes any Novice teams to Advanced Status. Those teams cannot also be place-winners in the Novice Contest.

Of the remaining Novice teams, the top three ranked teams will be place-winners of the Novice category.

Appealing Numbers

Time limit: (according to HackerRank's environment)
Standard Input, Standard Output

Problem Statement
Andrew is new in school, and has devised a scheme to make friends quickly. Each person at school has a favorite number, and Andrew knows that if his favorite number has a high bit-wise XOR with theirs, they are likely to become friends. Andrew is eager to make new friends, and to make this process faster he has an entire list of favorite numbers, from which he will pick the best one for each potential friend. Given a list of Andrew's N favorite numbers, as well as the favorite numbers of his M potential friends, determine which favorite number he should pick to tell to each one.

Input
The first line consists of two integers N and M, how many favorite numbers Andrew has, and the number of potential friends, respectively. The next line consists of N unique space separated integers, each one of Andrew's favorite numbers. The next M lines will each consist of a single integer, the favorite number of each potential friend.

Constraints
$1 \leq N \leq 10^5$
$0 \leq M \leq 10^5$

All favorite numbers will be in the range $[0, 2^{30}]$.

Output
Output M lines, each containing the best number for Andrew to pick for that respective friend.

Sample I/O

Input 1	Output 1
8 3 1 2 3 4 6 7 8 9 7 2 10	8 9 4

Explanation
For the first friend, his number is 7, which is 111 in binary. Andrew would choose 8, which is 1000 in binary to maximize his chance, since $8 \oplus 7 = 15$ ($\oplus$ denotes XOR) and you cannot do better than this.

Similarly, with the second friend it would be best to present the number 9, giving an XOR of 11.

With the last friend, Andrew would give him the number 4, giving a value of 14.

Camping

Time limit: (according to HackerRank's environment)
Standard Input, Standard Output

Problem Statement
You are the manager of a camping grounds made up of N by M *camp sites*, and it is your job to make sure that the site is set up properly. A *camp site* is a space just large enough to place a tent. Some camp sites are occupied by trees, making it impossible to place tents at those locations.

Campers don't like camping next to each other, so tents must not be immediately adjacent in the four cardinal directions, N, E, S, W, or diagonally adjacent (where the corners of their camp sites touch).

Write a program to determine if the layout of tents given in the problem is valid.

Input
The first line of input contains a single integer P, denoting the number of test cases to follow.

The first line of each test case will consist of two integers, N and M, denoting the size of the campground.

The following lines of each test case will each consist of characters in {'.', 'T', 't'}, where 'T' represents a tree, 't' represents a tent, and '.' represents an open camp site.

Constraints
$1 \leq P \leq 50$
$1 \leq N, M \leq 50$

Output
For each test case, print "Valid" if the campsite is valid. Otherwise, print "Invalid".

Sample I/O

Input 1	Output 1
3 2 4 ttTt T... 2 3 TtT ... 3 3 T.t .t. T.T	Invalid Valid Invalid

Explanation
In the first case, there are two tents next to each other, making the campsite invalid.

In the second case, each tent has enough space.

In the third case, two trees are diagonal from each other, making the campsite invalid.

Cards

Time limit: (according to HackerRank's environment)
Standard Input, Standard Output

Problem Statement
Andrew likes to play Unique Trade-Duel, a trading card game where every single card is unique. Despite them all being equally rare, there is one that he wants more than all others: Blue Eyes Unique Dragon. Sammy has this card, and is willing to trade it for a certain set of S other cards. Andrew might not have all of these cards to begin with, but he can go around town asking to trade with each of his friends. Unlike Sammy, Andrew's friends are only interested in trading cards one to one. Given the cards that Andrew starts with, the cards Sammy wants, and a description of trades Andrew can make with his friends, determine whether or not Andrew can make the trade with Sammy.

Input
The first line of input will consist of a single integer T, denoting the number of test cases. The first line of each test case will consist of three space separated integers N, M, and S, denoting the total number of cards, the number of cards Andrew starts with and the number of cards Sammy wants. the next two line will consist M and S space separated integers, denoting the set of cards that Andrew has, and the set that Sammy wants. The next N lines will each consist of a space separated list of x integers between 1 and N, denoting the cards that can be traded to obtain card C_i, or -1 if there are no cards to trade for C_i or if C_i is a card Andrew starts with.

Constraints
$1 \leq T \leq 30$
$1 \leq M, S \leq N \leq 30$

Output
A single line with "Yes" or "No", depending on if Andrew can make the trade.

Sample I/O

Input 1	Output 1
2 6 2 1 2 3 6 -1 -1 -1 1 2 1 3 5 5 2 2 1 2 4 5 -1 -1 1 2 3 3	Yes No

Explanation

In the first case, Andrew has cards 2 and 3, and Sammy wants card 6. Andrew trades his 3 for a 5, then his 5 for a 6.

In the second case, Andrew has cards 1 and 2, and Sammy wants cards 4 and 5. Sammy wants 3 for each of his cards, while only one 3 exists, making this trade impossible.

Carpet Buying

Time limit: (according to HackerRank's environment)
Standard Input, Standard Output

Problem Statement
Andrew is carpeting his living room, but would like to be as cost effective as possible. He knows that the room is N by M feet, but the carpeting store will only sell carpeting in perfect square increments. Andrew can cut and shape the carpet in any way he feels fit, He just needs to make sure he buys enough carpet square.

Andrew is going to buy one carpet square. He will then cut it to fit the dimensions of his room.

Given the dimensions of Andrew's living room, determine the minimum amount of carpeting he needs to purchase in order to cover the entire room.

Input
The first line of input contains a single integer T, denoting the number of test cases to follow. The following T lines will consist of two integers, N and M, denoting the width and depth of Andrew's living room for each test case.

Constraints
$1 \leq N, M \leq 10^4$

Output
For each test case, print a single integer, the amount of carpet Andrew must buy such that he can cover his room and it is a perfect square.

Sample I/O

Input 1	Output 1
4 1 17 6 6 5 10 99 100	25 36 64 10000

Falling Snow

Time limit: (according to HackerRank's environment)
Standard Input, Standard Output

Problem Statement

The weather forecast is in, and there is soon to be some heavy snow! But you have to get to class soon, and can't do that if there's too much snow on the sidewalk! The sidewalk leading to class is very straight, and can be modeled as a line along the X-axis. Luckily the weather forecast is very detailed, and tells you how much snow will fall from positions x_s to x_e at each time. It is very cold, so this snow will not melt at any time during the problem. Given this information and Q queries of the form (x_i, t_i) determine how much snow is on the sidewalk at position x_i and time t_i

Input

The first line of input will consist of two space separated integers, N and Q, denoting the number of snowfall intervals and the number of queries, respectively. The next N lines will consist of 4 space separated integers, i_s, i_e, i_a, and i_t, denoting the starting position of snowfall, the ending position of snowfall (inclusive), the height of snow in that interval, and the time at which it fell. No two snowfall intervals will overlap at any given time. The next Q lines will consist of two space separated integers, q_x and q_t, denoting the position and time of each query. If a query and a snowfall interval happen at the same position and time, include that amount of snow in the result.

Constraints

$1 \leq i_s \leq i_e \leq 10^5$
$1 \leq N, Q, i_t, i_a, q_x, q_t \leq 10^5$

Output

For each query Q_x, output the amount of snow at the position and time given in the query

Sample I/O

Input 1	Output 1
5 5 3 6 1 5 8 17 2 3 5 11 3 2 10 20 4 1 1 13 5 4 11 5 10 3 1 4 5 3 18 5	14 9 5 3 4

Explanation:
These are the heights for each X coordinate and time for the sample data:

X:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
T = 1	0	0	0	0	0	0	0	0	0	4	4	4	4	4	4	4	4	4	4	0
T = 2	0	0	0	0	3	3	3	3	3	7	7	4	4	4	4	4	4	4	4	0
T = 3	0	0	0	0	3	3	3	5	5	9	9	6	6	6	6	6	6	6	4	0
T = 4	5	5	5	5	8	8	8	10	10	14	14	11	11	6	6	6	6	6	4	0
T = 5	5	5	6	6	9	9	8	10	10	14	14	11	11	6	6	6	6	6	4	0

Garden Watering

Time limit: (according to Hackerrank's environment)
Standard Input, Standard Output

Problem Statement
Anne owns a garden on the edge of Byteland. Most of the time, she plants only plump helmets, but this time she has opted for sweet pods instead that she'll use to brew some Dwarven wine.

There is just one problem: the amount of water that each sweet pod requires can be different, and some can live on the water in the earth alone. Anne's farm is an N by M grid of unit squares with a natural water level of G. Each square is a plot where plants can be placed.

Plants in the square (i,j) of the farm grid need the same amount of water, which we will refer to as $W_{i,j}$. If $W_{i,j} \leq G$, then Anne does not need to water the plant at position (i, j).

Anne chooses to water all the plants that need watering ($W_{i,j} > G$). However, nothing separates adjacent plots that need watering, so water can flow freely between them. Anne decides to water the plants such that if there are two plots a, b connected by a path of water-needing plots, then the amount of water in both is at least $\max(W_{i_a,j_a}, W_{i_b,j_b}) - G$.

Anne also does not know the water level of her farm. For each water level G from 1 to $N \times M$, compute the amount of water she will need to use to water all the plants.

Input
The first line of the input will contain N and M, the dimensions of the farm.
The next N lines will contain M integers each, the j^{th} integer on the i^{th} line denotes $W_{i,j}$.

Constraints
$1 \leq N, M \leq 10^3$
$1 \leq W_{i,j} \leq N \times M$

Output
Output $N \times M$ lines, the i^{th} of which has the amount of water Anne needs if her farm has water level i.

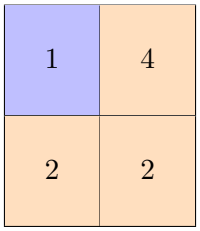
Sample I/O

Input 1	Output 1
2 2 1 4 2 2	9 2 1 0

Input 2	Output 2
3 3 1 4 5 2 3 6 2 7 1	42 25 13 7 3 1 0 0 0

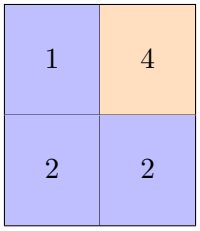
Explanation
Let's examine the first test case. Let a grid position be orange if Anne has to water it and blue if she does not.

For G = 1, we have this figure:



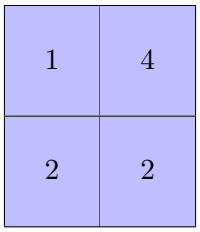
Since all but the top-left cell are connected, Anne has to fill them with the maximum amount required in all grids, namely 4. However, since all positions has already has 1 unit of water in the soil, this means she needs 3 more for each position, giving an answer of $3 \cdot 3 = 9$

For G = 2 and G = 3, we have this figure:



Only the top right cell needs watering, so 4 units are needed. In the case where x=2, this means $4 - 2 = 2$ units of water. In the case where x=3, this means $4 - 3 = 1$ unit of water.

For G = 4, we have:



Nothing needs watering, so the cost is 0.

Halfway Point

Time limit: (according to HackerRank's environment)
Standard Input, Standard Output

Problem Statement
Andy and Bob just bought the world's longest licorice rope, which consists of N points that are connected sequentially by line segments. Since they split the cost of the rope evenly, they want to divide it in half. Unfortunately, it gets tangled on the travel back home, so they aren't sure where the halfway point is. Can you help them figure it out?

Input
The first line of input is a single integer N, the number of points that make up the rope.

The next N lines each have two space-separated integers, x_i and y_i, which are the coordinates of the i^{th} point.

Constraints
$2 \leq N \leq 10^3$
$-10^4 \leq x_i, y_i \leq 10^4$

Line segments may overlap, but no segments will have 0 length.

Output
Output the coordinates of the halfway point along the licorice rope on a single line. Your answer will be accepted if it is within an absolute error of 10^{-6}.

Sample I/O

Input 1	Output 1
3 0 0 1 0 1 1	1.000000 0.000000

Input 2	Output 2
4 0 0 1 1 1 0 0 1	1.000000 0.500000

Explanation
In the first example, the rope is made of two segments, each of length 1. Hence, the 2nd point $(1, 0)$ is the halfway point.

In the second example, the 1st and 3rd line segments have a length of $\sqrt{2}$ and the 2nd has length one, so the halfway point is halfway down the 2nd segment $(1, 0.5)$.

Hopping Stones

Time limit: (according to HackerRank's environment)
Standard Input, Standard Output

Problem Statement
A young boy named Lucas is visiting his grandmother over the weekend. After arriving at her front gate, he must hop across N stepping stones arranged in a line in order to reach the front door. Since he visits very frequently, he decides to always hop the stones differently every time.

He starts on the pavement before the first stone, and the front door is at the last stone. At each stone, he can hop forward $M_1, M_2, ...,$ or M_K stones. How many ways can Lucas hop across the stones so that he arrives at the last one?

Input
The first line contains a single integer, T, the number of test cases.

Each test case consists of two lines. The first line contains 2 space-separated integers N and K, the number of stones and the number of different moves respectively. The second line contains K space-separated integers $M_1, M_2, ..., M_k$, the numbers of stones Lucas can hop forward on each hop sorted in ascending order.

Constraints
$1 \leq T \leq 10$
$1 \leq N \leq 10^6$
$1 \leq K, M_i \leq 50$

$M_i < M_{i+1}$ for all $1 \leq i < K$
$1 \leq M_i \leq N$ for all $1 \leq i \leq K$

Output
Output the number of ways Lucas can hop across all N stones, modulo $10^9 + 7$.

Sample I/O

Input	Output
2 4 2 1 2 1 1 1	5 1

Explanation
In the first test case, the 5 ways Lucas can hop the stones are $(1, 1, 1, 1)$, $(1, 1, 2)$, $(1, 2, 1)$, $(2, 1, 1)$, and $(2, 2)$.

In the second test case, since there is one stone, the only way is (1).

Last Stand

Time limit: (according to HackerRank's environment)
Standard Input, Standard Output

Problem Statement
The aliens have your base surrounded! They have setup laser shields all around you in preparation for their assault. Luckily for you, you have a trick up your sleeve (or rather in your gun storage): the Shield Breaker MK III! This ray gun will destroy any shield that its ray touches. Bringing the gun outside your base is not possible because the gun is heavy, so you can only fire the gun from your base. Unfortunately, the gun has been used a lot during the war and only has one charge remaining. What is the maximum number of shields that you can destroy with the Shield Breaker MK III?

Your base is located at the origin on a 2D plane. Each laser shield is a line segment connecting its starting point and ending point. The ray will destroy the laser shield if it intersects the shield at either endpoint or anywhere in between. The ray will start from the origin and continue infinitely in any direction you choose.

You've only got one shot at this, so make it count! (Note: "one shot" refers to the ray gun, not the number of submissions allowed)

Input
The first line of input contains a single integer N, the number of laser shields the aliens have setup.

The next N lines each contain 4 space-separated integers, x_1, y_1, x_2, and y_2, which are each laser shield's endpoints.

Constraints
$1 \leq n \leq 10^5$
$-10^9 \leq x_1, y_1, x_2, y_2 \leq 10^9$

No laser shield intersects with your base at the origin, but the shields can cross each other.

Output
Output the maximum number of laser shields you can destroy from a single shot using the Shield Breaker MK III.

Sample I/O

Input	Output
4 0 2 2 0 0 3 3 3 4 0 4 4 -1 1 -1 -1	3

Explanation
Firing the ray gun 45° counter-clockwise from the positive x-axis direction allows you to dissipate shields (0, 2)-(2, 0), (0, 3)-(3, 3), and (4, 0)-(4, 4).

Tower Breaker

Time limit: (according to HackerRank's environment)
Standard Input, Standard Output

Problem Statement
Anne and Puffy are playing a game. Initially there are N pebbles in a pile. Puffy and Anne take turns removing from the pile until none are left. To make things interesting, they use the following rules to determine how many pebbles each player can choose to remove on their turn:

1. Players choose to remove d pebbles, where d is a divisor of the number of pebbles currently in the pile, but not 1 or the size of the pile itself.
2. If a player has no other option, they must take 1 pebble.

If on their turn, Anne or Puffy have no more pebbles to take, they lose. Anne, of course, lets Puffy move first. Given the number of pebbles initially in the pile, determine who will win if Anne and Puffy both play optimally.

Input
The first line contains T, the number of test cases.
Each test case consists of one line containing a single integer N, the number of pebbles on the pile.

Constraints
$1 \leq T \leq 10^4$
$1 \leq N \leq 2 * 10^6$

Output
For each test case on a separate line, output either "Puffy" or "Anne".

Sample I/O

Input 1	Output 1
6 1 2 4 20 23 771	Puffy Anne Puffy Puffy Anne Puffy

Wicked Wind

Time limit: (according to HackerRank's environment)
Standard Input, Standard Output

Problem Statement
Anne's parents are gone for the weekend, and assigned her the task of cleaning up the leaves outside on their front yard.

Now given m leaves lying across the yard at positions $l_0, l_1, l_2, l_3, ..., l_{m-1}$ such that $l_i < l_{i+1}$, the annoyance level of Anne after collecting all the leaves is:

$$\sum_{i=1}^{m-1} \left(l_i - l_{i-1}\right)^2$$

or to put it in words, the sum of the square of distances between adjacent leaves.

Of course, Anne procrastinates and ended up doing it the day before her parent would return. By that time, each leaf is blown away with probability $\frac{1}{2}$. Given N, the number of leaves, and the positions of all the leaves, find out the expected annoyance level Anne gains after collecting the leaves. Output this value multiplied by 2^N modulo $10^9 + 7$. It can be proven that this value is an integer.

Input
The first line contains T, the number of test cases.
Next T pairs of lines denotes each test case. The first in the two contains one number N, the number of leaves there are originally.
The next line contains N integers. The i^{th} of which denotes l_i. It is guaranteed that no two l_i are the same, and they are given in increasing order.

Constraints
$1 \leq T \leq 10$
$1 \leq N \leq 10^6$
$0 \leq l_i \leq 10^9$

Output
Output Anne's expected annoyance level multiplied by 2^N modulo $10^9 + 7$.

Sample I/O

Input 1	Output 1
2 5 1 2 3 4 5 4 1 3 6 1000000000	132 1042