RUET
NATIONAL COLLEGIATE
PROGRAMMING CONTEST
2015
Hosted by
Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology

Hosted By

Rajshahi University of Engineering and Technology (RUET)

Rajshahi, Bangladesh

September 12, 2015

You get 22 Pages
11 Problems
&
300 Minutes

Rules for NCPC 2015

a) Solutions to problems submitted for judging are called runs. Each run is judged as accepted or rejected by the judge, and the team is notified of the results. Submitted codes should not contain team or University name and the file name should not have any white space.

b) A contestant may submit a clarification request to judges. If the judges agree that an ambiguity or error exists, a clarification will be issued to all contestants.

c) Contestants are not to converse with anyone except members of their team and personnel designated by the organizing committee while seated at the team desk. But they cannot even talk with their team members when they are walking around the contest floor to have food or any other purpose. Systems support staff may advise contestants on system-related problems such as explaining system error messages. Coaches should not try to enter the contest area under any circumstances.

d) While the contest is scheduled for a particular time length (five hours), the contest director has the authority to alter the duration of the contest in the event of unforeseen difficulties. Should the contest duration be altered, every attempt will be made to notify contestants in a timely and uniform manner.

e) A team may be disqualified by the Contest Director for any activity that jeopardizes the contest such as dislodging extension cords, unauthorized modification of contest materials, distracting behavior of communicating with other teams.

f) Contestants will have foods available in their contest room during the contest. So they cannot leave the contest room during the contest without permission. The contestants are not allowed to communicate with any contestant (Even contestants of his own team) or coach while are outside the contest floor.

g) Teams are not allowed to bring calculators, mobile phones or any machine-readable devices like CD, DVD, Pen-drive, IPOD, MP3/MP4 players, floppy disks etc.

h) With the help of the volunteers, the contestants can have printouts of their codes for debugging purposes. Passing of printed codes to other teams is strictly prohibited.

i) The decision of the judges is final.

j) Teams should inform the volunteers if they don't get reply from the judges within 10 minutes of submission. Volunteers will inform the judges for further action. Teams should also notify the volunteers if they cannot log in into the PC^2 system. This sort of complains will not be entertained after the contest.

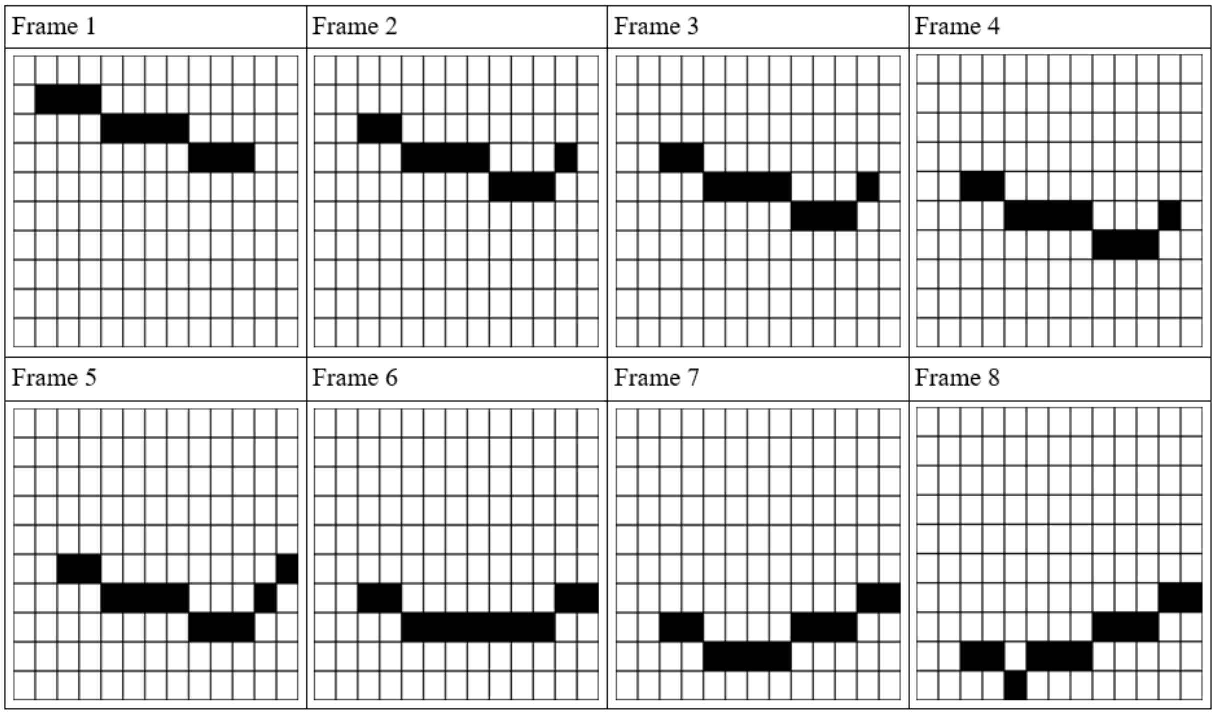
A Morphing Bitmaps

Most animation tools that support 2D animation support defining some keyframes where the user will specify what the image will like at a particular time for that particular frame. The user tells the tool how many frames to use (or how long the animation will run). The tool would then generate the intermediate frames so that the transition from one frame to another looks smooth.

For this particular problem, we would address the problem from a different perspective. We would supply a source frame and a target frame as keyframes. The problem is to find out the minimum number of frames needed to go from source to target.

To keep it simple, we are only dealing with bitmap keyframes. Here each pixel will be either black or white. The pixels can travel from one location to another from frame to frame – infact, multiple black pixels can travel going from one frame to the next one. Each pixel can move one cell up, down, left or right. While traveling a pixel is allowed to collide with another. As the target keyframe can have more (or less) black pixels compared to the source keyframe, we need some sort of mechanism to generate new (or to eat up existing) black pixels. We will allow a black pixel in one frame to turn any combination of it's four neighboring pixels (up, down, left and right) black to generate new black pixels when needed. We will also allow a black pixel in one frame to turn itself white in the next frame if needed. Without the help of a neighboring black pixel, a pixel won't turn black by itself. To further simplify, we'll also assume that the source and target key frames will have at least one black pixel. For example, a black pixel can turn its top, left and right pixels black (not modifying the bottom pixel) and turn itself white – all of this going from one frame to the next. However, a black pixel cannot travel two pixels to the right in one frame as it's allowed to travel only one cell in a frame.

The following illustration may help explain the process:



Input

There can be several test cases (less than **500**). Each test case will start with 2 integers, the number of rows, **R (1 <= R <= 100)** and columns **C (1 <= C <= 100)** for the source and target bitmap keyframes. The next **R** lines will contain **C** characters each, defining the source keyframe. The **R** lines right after that will define the target keyframe in the same fashion. Each character of the keyframe will either be a '**.**' denoting white pixels or be '**#**' denoting a black pixel. We can assume that the maximum number of black pixels in a keyframe will be less than **4,000**. A test case with **R = 0** and **C = 0** will signify the end of input.

Output

Output for each test case will start with the case label in the format "**Case k:** " where k is the case number (starting from 1) followed by the number of frames needed to go from the source keyframe to the target keyframe.

Sample Input	Output for Sample Input
10 13 ....###......... ....####..... ....###.. ....## ....###.. .....###..... ..###........ 2 2 .. .# .. .# 0 0	Case 1: 7 Case 2: 0

Problem Setter: Monirul Hasan
Special Thanks: Muhammed Hedayet

B Palindromic Bases

A palindrome is a number which looks the same when reversed. 1, 11, 343 etc. are examples of palindromes, while 12, 1122 etc. are not. Interesting thing is, some number may not be palindrome in decimal but in other bases. For example, if we convert 12 in base 5, $(12)_{10} = (22)_5$. So we can see that 12 is a palindrome in base 5. Length of a palindrome is the number of digits in that number. No leading zeroes are allowed in palindromes.

Given a number **N**, how many bases **B** are there, where **N** will be an even length palindrome if represented as a **B** base number.

Input

First line will contain an integer number **T** **(0<T≤100)**, number of test cases. Each case contains one line with an integer **N** **(0≤N≤10^14)**.

Output

For each case print the case number then the answer. If there are infinite bases possible for a particular **N**, you should output "Infinity". See sample I/O for more clarification.

Sample Input	Output for Sample Input

`3`
`3`
`4`
`12`
	`Case 1: 1`
	`Case 2: 1`
	`Case 3: 2`

Explanation for sample inputs:
Case 1: the base is 2.
Case 2: the base is 3.
Case 3: the bases are 5 and 11.

Problem Setter: Hasnain Heickal
Special Thanks: Muhammad Ridowan

C Farey Sequence

The Farey sequence of order n is the sequence of completely reduced fractions between 0 and 1 which, when in lowest terms, have denominators less than or equal to n, arranged in ascending order. Farey sequence for different values of n are shown in the figure on the left below:

$$F_1 = \left\{ \frac{0}{1}, \frac{1}{1} \right\}$$

$$F_4 = \left\{ \frac{0}{1}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{1}{1} \right\}$$

$$F_7 = \left\{ \frac{0}{1}, \frac{1}{7}, \frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{2}{7}, \frac{1}{3}, \frac{2}{5}, \frac{3}{7}, \frac{1}{2}, \frac{4}{7}, \frac{3}{5}, \frac{2}{3}, \frac{5}{7}, \frac{3}{4}, \frac{4}{5}, \frac{5}{6}, \frac{6}{7}, \frac{1}{1} \right\}$$

Figure 1:

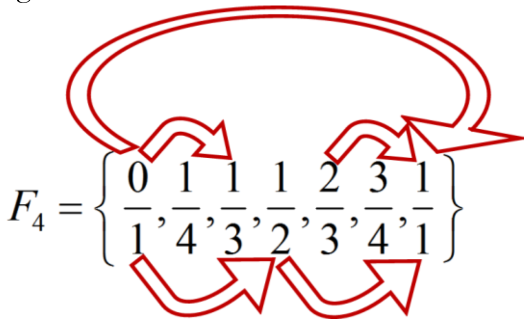


Figure 2: Five desired pairs in F4

It is very well known that if $\frac{m_1}{n_1}$ and $\frac{m_2}{n_2}$ are two consecutive fractions of a Farey Sequence then $m_2 n_1 - m_1 n_2 = 1$. But many fractions which are not consecutive also show this property. For example, in F7, $\frac{2}{5}$ and $\frac{1}{2}$ also show this property although they are not consecutive fractions in F7. Given the value of n, your job is to find number of pair of non-consecutive fractions $\frac{m_i}{n_i}$ and $\frac{m_j}{n_j}$, such that $m_j n_i - m_i n_j = 1$.

Input

Input file contains at most 20000 lines of input. Each line contains a positive integer which denotes the value of n (0 < n < 1000001). Input is terminated by a line containing a single zero. This line should not be processed.

Output

For each line of input produce one line of output. This line contains number of pair of non-consecutive fractions $\frac{m_i}{n_i}$ and $\frac{m_j}{n_j}$, (j - i > 1) in Farey Series F_n, such that $m_j n_i - m_i n_j = 1$

.

Sample Input	Output for Sample Input
1 4 0	0 5

Problem Setter: Shahriar Manzoor
Special Thanks: Syed Shahriar Manjur

D Ultimate Mango Challenge

Rajshahi is one of the oldest cities in Bangladesh. Its land is mother of one of the most delicious fruits in the world, Mango. You can find mango in various parts of the world, but the mango from Rajshahi is famous worldwide for its unique and delicious taste. People are so obsessed with these mangoes that sometimes they hear "রাজশাহীর আম আছে" in japanese shop (well… they actually say "irasshaimase" which means welcome). The variety of mangoes that grows around Rajshahi is truly overwhelming. As an outsider, it might be really surprising to see people of Rajshahi eating mangoes (or food prepared from mangoes) three times a day. I was surprised first time to see such enthusiasm. But I was not ready to face what was coming next.

One of my relative from Rajshahi challenged me to eat mangoes like they do. When I tried to get out of it I was mocked in various ways, and was called a coward. So I decided to give it a shot. But as a CSE major, I wanted to know beforehand whether I am going to pass or fail the challenge. It will be really easy to find out using a program. But I can't use laptop keyboard and only you can help me out from this big trouble.

Input

First line contains **T (T<10)**, number of test cases. Each case starts with **N (0<N<10)**, different types of mangoes and **L (0<=L<=20)**, my limit of eating maximum number of mangoes at one meal. Next line contains **N** integers, all of them will be less than **10**. Each of these integers indicates number of mangoes of each kind in the plate in front of me. Next line will contain **N** integers, all of them less than **10**. Each of these integers indicates my limit of eating a particular type of mango. I need to finish every mango in the plate to win the challenge. Total number of mangoes I eat cannot exceed my limit **L**. Also number of a particular type mangoes eaten should also not exceed the limit for that particular type.

Output

For each case, output "**Case X:**" where **X** is the case number. After that print **Yes** or **No** based on whether you can win the challenge or not.

Sample Input	Output for Sample Input
3	Case 1: No
3 5	Case 2: No
1 2 3	Case 3: Yes
1 3 3	
3 6	
1 2 3	
1 3 2	
3 6	
1 2 3	
3 3 3	

Problem Setter: Hasnain Heickal
Special Thanks: Md. Shiplu Hawlader

E Train Station

Not that long ago a new railway station was built at Rajshahi, a nice city in Bangladesh. The old one was very small in size. After searching for a while I got a picture of the old station. As you can see the station was mostly open ended in both sides back then. Current railway station is mostly closed in the inner end. In closed end case, the advantage is, we always know where the train will stop and we also know the location of the exit of the train station. So if you want to optimize your time, you can cleverly choose the train bogie and thus exit the whole station at the earliest possible time.

However the situation is not that simple for open ended train stations. The bogie may stop at anywhere at the line. Well, not exactly anywhere, but if we travel for several days we kind of know which bogie usually stops near the exit. Under this circumstances, we would like to compute expected amount of time one requires to exit the station.

For our convenience let us assume the width of a train bogie is negligible and the train line is y-axis. Let the exit of the train station be **(x, 0)**. After traveling several days I know which bogie I should ride. Suppose the length of that bogie is **d**. We also know that the train randomly stops in the station in such a way that the entire bogie is completely within **(0, y1)** and **(0, y2) [y1 - y2 >= d]**. A bogie has 2 exits, one at the very beginning and another at very end. My sitting location is uniformly random within the bogie. However, when the train stops I look outside through the window and decide which end of the bogie I should exit through to get out of the station at the earliest possible time. If I can walk 1 unit distance per second, what is the expected time for me to get out of the station?

Input

First line of the input will be the number of test cases **T (≤ 1,000)**. Hence **T** test cases follow. Each case consists of a single line with four integers: **x, y1, y2, d [-100 ≤ y1, y2 ≤ 100, 0 < d, x ≤ 100, y1 - y2 >= d]**.

Output

For each test case output: "**Case t: ans**" where **t** is the case number and **ans** is the expected amount of time. Precision error of **10^-4** will be accepted.

Sample Input	Output for Sample Input

3
1 1 -1 2
1 1 -1 1
1 10 8 1
	Case 1: 1.914214
	Case 2: 1.382597
	Case 3: 9.058678

Problem Setter: Md. Mahbubul Hasan
Special Thanks: Zobayer Hasan

F Tree Weights

A rooted tree with **N** nodes is given. Nodes are labeled 1 to **N**, 1 being the root of the tree. Each of the leaves of this tree has a value assigned to it, which is zero at the beginning. The value for each internal node **U** is calculated as the sum of the values of all the nodes in the sub-tree rooted at **U**. An internal node is a node, which has at least one child node.

You will be given two kinds of operations:

Type 1: given **U**, find the value of node **U**.

Type 2: given **U** and **X**, increase the value of the leaf **U** with **X**.

Input

First line starts with **T (0<T≤10)**, number of test cases. Each of the case starts with **N (0<N≤10^5),** number of nodes in the tree. Next there will be **N-1** lines each containing two integers **U** and **V**, indicating an edge between **U** and **V**. Next there will be **Q (0<Q≤10^5)**, number of operations. Next **Q** line will contain firstly **TP (1 or 2)**, the type of the operation. Then based on the operation type, there will be one or two integers, **U** or **U** and **X (1≤U≤N, |X|≤10^9)**. In case of **TP = 2**, **U** will always be a leaf node.

Output

For each case, print case number. Then for each operation of type 1, print the answer in a separate line. As value of the nodes can get huge, print the answer modulo **1,000,000,007**. See sample I/O for more clarification.

Sample Input	Output for Sample Input

1
4
1 2
1 3
3 4
6
2 2 1
1 1
1 3
2 4 3
1 1
1 3
	Case 1:
	1
	0
	7
	3

Problem Setter: Hasnain Heickal
Special Thanks: Muhammad Ridowan

G Defense Plan

You are playing a game where it is possible to place defensive towers on the map. They are not like any other defense towers though. They will only protect the area within the convex hull formed by them. However, you cannot place any number of towers, because that will be really unfair. The towers can only be placed on a tower mount. There are several tower mount throughout the map, only one tower can be placed on a tower mount. If you can only place exactly **N** towers out of **P** tower mount locations, where **N ≤ P**, what is the maximum area that can be covered by the towers?

Input

There will be **T (T ≤ 100)** test cases. Each case contains two integers **P** and **N (3 ≤ N ≤ P ≤ 100)** as described in the statement. Then there will be **P** pairs of integers **(x, y)** denoting the coordinates of tower mounts, **0 ≤ x, y ≤ 1000**. Test cases will be separated by blank lines. **Note that, 50% of the test cases are randomly generated.**

Output

For each case, print the test case number starting with 1, and then a real number denoting the maximum possible area. The output area should be rounded to three decimal places. It is guaranteed the area will be positive.

Sample Input

```
3

7 4
2 2
1 5
6 1
5 5
3 7
7 6
9 4

3 3
0 0
1 1
0 1

4 4
0 1000
1000 0
1000 1000
0 0
```

Output for Sample Input

```
Case 1: 24.000
Case 2: 0.500
Case 3: 1000000.000
```

Problem Setter: Zobayer Hasan
Special Thanks: Md. Shiplu Hawlader

H VIP Treatment

For any government project, VIPs are always a problem. You can't expect a high profile VIP person to wait for a month for some job to be done like a normal people. This is a big problem for government software projects as you have to create alternate ways to bypass business logic and finish some job early. Now you are working on some government software project and you have to manage some job requests, processing and management as well as have to do VIP treatment. In this problem we will solve a sub-problem of it, assigning the jobs to worker.

For the software there are **M** kinds of jobs, numbered from 1 to **M** and **N** workers numbered from 1 to **N**. As the jobs are mostly similar, so the time needed for a particular worker to do any kind of job request is same. Different worker may need different amount of time. Now for each **j-th (1≤j≤M)** kind of job, your software have to consider 3 things,

- Number of VIP requests $\mathbf{v_j}$ for this kind of job.
- Number of regular requests $\mathbf{r_j}$ for this kind of job.
- A non-empty list of workers $\mathbf{wr_j}$ who can do this kind of job.

All the VIP job request have to be processed. But the management also wants to process at least $\mathbf{K}$ $\mathbf{(0 \le K \le} \sum_{j=1}^{M} r_j)$ total regular job requests.

The software have to assign this requests to workers and have to do it in a way that total time needed is minimized. Now a worker numbered **i (1≤i≤N)** can complete any kind of job request in **Wi** time but can't do more than one job at a time. After the requested job is finished, he can do another job again, which will cost him another **Wi** time. So if any worker numbered **i** does total **Ti** jobs, it would take him **Ti*Wi** time to finish.

The total time needed to finish the project is the maximum time needed by any worker (Maximum **Ti*Wi** for all **i** such that **1≤i≤N**). Your job for this problem is to make such an arrangement that the total time needed is minimized.

Input

The first line of input contains a positive integer number **TC (TC≤200)**, number of test case. Then **TC** test cases follow.

There will be blank line before each test case.

First line of each test case will contain three space separated integers **M (1≤M≤50), N (1≤N≤50)** and **K**. Next line will contain **N** space separated integer numbers, **i-th** of those numbers will be **Wi,** time needed by worker **i** to do a single job (**1≤Wi≤100**). Each of the next **M** lines contains description of a job, where the first three integers of the **j-th** line are

$\mathbf{v_j (0 \le v_j \le 1{,}000{,}000)}$ number of VIP job requests of kind **j**, $\mathbf{r_j (0 \le r_j \le 1{,}000{,}000)}$ number of regular job requests of kind **j** and $\mathbf{n_j (1 \le n_j \le N)}$ number of workers who can do **j-th** kind of job. Then $\mathbf{n_j}$ integers follow. The **k-th** of the next $\mathbf{n_j}$ integers is $\mathbf{wr_{jk}}$ ($1 \le \mathbf{wr_{jk}} \le \mathbf{N}$), means worker numbered $\mathbf{wr_{jk}}$ can do **j-th** kind of job.

Output

For each test case print a line in "**Case I: T**" format where **I** is the case number and **T** is the minimum time required for doing all VIP requests and at least **K** regular job requests.

Sample Input	Output for Sample Input

```
3

3 3 10
2 4 8
2 3 1 1
2 3 1 2
2 4 1 3

2 1 4
2
2 3 1 1
3 2 1 1

2 2 4
1 2
2 3 2 1 2
3 2 2 1 2
```
	```
	Case 1: 48
	Case 2: 18
	Case 3: 6
	```

Explanation for Sample Case

In the first case, there are 3+3+4=10 regular job requests and 10 VIP job requests. Also for each type of job, only one worker can do it. So

The 1st worker get 2+3=5 jobs and with 2 unit time per job he needs 5*2=10 unit of time.

The 2nd worker get 2+3=5 jobs and with 4 unit time per job he needs 5*4=20 unit of time.

The 3rd worker get 2+4=6 jobs and with 8 unit time per job he needs 6*8=48 unit of time.

As they all can work independently, total time needed is 48.

In the second case, there in only one worker and he got total 2+3=5 VIP job requests and 4 regular job requests. So He need (5+4)*2=18 unit of time.

In the third case, worker 1 will do 6 jobs with 6*1=6 unit of time and worker 2 will do 3 jobs with 3*2=6 unit of time. So to do 9 jobs, total time needed is 6.

Problem Setter: Muhammad Ridowan
Special Thanks: Md. Shiplu Hawlader

I Jumping Frogs

At time 0, **R** red frogs and **G** green frogs are sitting on a straight line. All the positions of the frogs are non-negative integer numbers. Every second, all the frogs jump. Each of the frogs has its own velocity, i.e., every second the **i-th** frog jumps **Vi** units to its left or right depending on the color. Every red frog jumps to its right, and every green frog jumps to its left.

The line is divided into **N + 1** contiguous segment. The left end of the first segment is always 0 and the right end of the **N+1st** segment is 10^9. The segments are denoted by a sequence of **N** positive integers. For example, if **N = 1** and the sequence has 1 integer number 10, then there are two segments, one is from 0 to 10 and another is from 10 to 10^9, both inclusive.

You are given the initial positions of all the **R + G** frogs and a sequence of positive integers describing the segments. Find the minimum time it will take for all the frogs to reach a single segment. A frog is said to be on a segment if and only if it's sitting on some points inside the segment (including the endpoints). Please note that a frog is not said to be inside a segment when it's jumping.

Please note that, when a frog is on any of the N intermediate boundary points, they can be considered to be part of either the left or the right segment.

Input

Input starts with a single positive integer, **T ≤ 10**, on a single line, denoting the number of test cases.

The first line of each test cases will be a blank line. Next line will contain three positive integers **R**, **G** and **N** (**1 <= R, G <= 100,000, 1 <= N <=100,000).**

Next five lines will be as follows:

1. **R** non negative integers, where the **i-th** integer represents the position of the **i-th** red frog.
2. **R** non negative integers, where the **i-th** integer represents the velocity of the **i-th** red frog.
3. **G** non negative integers, where the **i-th** integer represents the position of the **i-th** green frog.
4. **G** non negative integers, where the **i-th** integer represents the velocity of the **i-th** green frog.
5. A sequence of N positive integers describing the segments. All the numbers are greater than 0 and are less than 10^9

Note that, every frogs' position and velocities are between 0 and 10^9, inclusive.

Please note that the input file is around 4 MB, use faster input/output routine.(i.e. scanf/printf instead of cin/cout for c++)

Output

For every case print the output in format, "**Case X: Y**", where **X** is the number of test case, starting from 1 and **Y** is the minimum time it takes for all the frogs to reach a single segment. If it's impossible for all the frogs to reach a single segment, then **Y** should be **-1**.

Sample Input	Output for Sample Input

2
 	
1 1 1
10
10000
20
10000
1000000
 	
2 2 1
1 2
99 100
1000 1001
100 200
100
	Case 1: 0
	Case 2: 1

Problem Setter: Muhammed Hedayet
Special Thanks: Hasnain Heickal

J Just Some Permutation 5

Given **N** and **K**, find the lexicographically **K-th** (1-indexed) smallest permutation **P_1, P_2 … P_N** of the first **N** positive integers (**1, 2 … N**), such that the adjacent numbers are relatively prime [**GCD(P_i, P_{i+1}) = 1**, for **$1 \le i < N$**] in the permutation. A permutation of **N** numbers **A_1, A_2 … A_N** is lexicographically smaller than another permutation **B_1, B_2 … B_N** if **$A_i < B_i$** for some **i** and **$A_j = B_j$** for all **j<i**.

Input

First line of the input contains an integer **T (≤20)**, which is the number of test cases. Each of the next **T** lines contain two space separated integers **N (1≤N≤28)** and **K (1≤K≤10^18)**.

Output

For each test case output the case number and then **N** space separated integers which is the lexicographically **K-th** smallest permutation of the first **N** positive integer numbers, such that adjacent numbers in the permutation are relatively prime. If there are less than **K** such permutations then output '**-1**'. See sample input output for exact formatting.

Sample Input	Output for Sample Input

```
3
3 3
4 2
4 20
```
	```
	Case 1: 2 1 3
	Case 2: 1 4 3 2
	Case 3: -1
	```

Problem Setter: Tasnim Imran Sunny & Md. Shiplu Hawlader
Special Thanks: Syed Shahriar Manjur

K Toll Management (III)

Government of Byteland is not happy with their yearly money collection and revenue this year. They have imposed several taxes and VATs here and there. This time the Government is going to increase the toll on highways of the country. But people of Byteland are very much dissatisfied with the Government's taxation and revenue generation policies. So Government is rethinking their policies.

There are **N** cities and **M** highways in Byteland. Each highway directly connects two different cities. Government has fixed a toll for each highway and whoever passes the highway have to pay the toll. Government wants to increase the toll from next month. But as they don't want to dissatisfy the people of Byteland anymore, they came up with a unique idea. They will change (increase or decrease) the toll of some highway such that the shortest path cost from the capital to any other city will remain same as now. So they want to know two information for each **i-th** highway:

1) $\mathbf{A_i}$: What is the maximum amount of toll they can increase without affecting the shortest path cost from capital city to all other cities?

2) $\mathbf{B_i}$: What is the maximum amount of toll they can decrease without affecting the shortest path cost from capital city to all other cities?

In other words, if $\mathbf{C_i}$ is the current toll of highway **i**, then if the Government updates the toll of the highway to $\mathbf{C_i + A_i}$ **(or** $\mathbf{C_i - B_i}$**)**, the shortest path cost from capital city to city **1, 2… N** does not change at all. Note that the Government is not imposing the new toll right away, they just want to estimate the **A** and **B** values of each highway. So when estimating the **A** and **B** values of a highway, there is no new toll imposed on any highway.

As you are the great programmer of Byteland, you are going to help the Government of Byteland to find $\mathbf{A_i}$ and $\mathbf{B_i}$ values for each **i-th** highway (for **1≤i≤M**).

Input

First line of the input contains a positive integer **T (T<10)**, denoting the number of test cases.

First line of each test contains two integer numbers **N** and **M** **(2≤N≤10,000, 1≤M≤100,000)**, denoting the number of city and number of highway respectively. Each of the next **M** lines contains the description of a highway, where the **i-th** line contains three integer numbers **Ui, Vi** and **Ci (1≤Ui, Vi ≤ N, Ui != Vi, 0≤Ci≤1,000)**, that means there is a highway from city **Ui** to city **Vi** and the toll of the highway is **Ci**. All the highways are directional, that means if there is a highway from city **x** to city **y**, not necessarily there exist a highway from city **y** to **x**, unless it is mentioned explicitly. Note that the city numbered 1 is the capital city of Byteland and every other city (2 to **N**), will be reachable from the capital city.

Warning: Large I/O file, user faster input output.

Output

For each test case, output the test case number and a single integer **S**, where

$$\mathbf{S} = \sum_{i=1}^{M} i * A_i + i^2 * B_i$$

If the value of $\mathbf{A}_i$ or $\mathbf{B}_i$ is infinite then replace the value with **-1**. The answer will be such that it will fit into 64-bit signed integer.

Sample Input	Output for Sample Input

```
2
4 4
1 2 5
1 3 5
2 4 5
3 4 5
4 4
1 2 5
1 3 5
2 4 4
3 4 5
```
	```
	Case 1: -7
	Case 2: 12
	```

Explanation of the sample test cases:

For the first test case, $(A_1, B_1) = (0\ 0)$, $(A_2, B_2) = (0\ 0)$, $(A_3, B_3) = (-1\ 0)$ and $(A_4, B_4) = (-1\ 0)$.

For the second test case, $(A_1, B_1) = (0\ 0)$, $(A_2, B_2) = (0\ 0)$, $(A_3, B_3) = (0\ 0)$ and $(A_4, B_4) = (-1, 1)$.

Problem Setter: Md. Shiplu Hawlader
Special Thanks: Monirul Hasan