RUET Inter University Programming Contest 2019

Technocracy 2019

Hosted by ECE, RUET

Rajshahi, Bangladesh

5th July 2019

You get 20 Pages, 11 Problems & 300 Minutes

Platform Support:

CodeMarshal

Problem Set By:

Bangladesh Association of Problem Setters

Problem A

Input: Standard Input
Output: Standard Output

I have short description

Given two integers **N** & **M**, output the value of $\sum_{r=1}^{M} (N \% r)$

Input

The first line of the input contains an integer **T**, denoting the number of test cases. Then the description of the **T** test cases follow. The first and the only line of every test case will contain two integers **N** and **M**.

Constraints

- **1 ≤ T ≤ 3**
- **1 ≤ N ≤ 10^{100000}**
- **1 ≤ M ≤ 10^{5}**

Output

For each test case, print a line containing the case number and the value of $\sum_{r=1}^{M} (N \% r)$.

Sample Input

```
2
193 17
2441139 505
```

Output for Sample Input

```
Case 1: 68
Case 2: 66262
```

Problem B	Input: Standard Input Output: Standard Output
Playing with capitals	

Byteland is a very versatile country. The economic condition of this country changes a lot. That's why the government of Byteland needs to change its capital very frequently. There are **N** cities and all those cities are always connected with **N-1** bidirectional roads. Each node is labelled with integers from **1** to **N**. Each node also has a value associated with it which is called **VAT** for that specific node. When someone wants to send some goods through a city **U**, then he will have to pay $\mathbf{VAT_U}$ amount of money to the government. Initially, the capital city is the node labelled with **1**.

There is a ministry of the government to handle this type of situation when economic condition changes. But to compute a few things faster, the ministry has hired you three guys to help them out with the following works:

1. Given a node **U**. You have to destroy the road connecting **U-V** where **V** is a city which is directly connected with **U** and **V** is the closest city to present capital city **C** among all the adjacent city of **U**. And then create a new bidirectional road **C-U**. After these two steps, make **U** the capital. It is guaranteed that **U** is not the present root.
2. Given two nodes **U V**. You have to find two things -
    a. Find the closest city from the capital which lies on the simple path from **U** to **V**.
    b. Figure out the total **VAT** to parcel some goods from **U** to **V**.

The government already know that you guys are great programmers. So you have to help them now.

Input:

The first line of the input contains a single line **T(1 ≤ T ≤ 3)** denoting the number of test cases.

The first line of each test case contains two integers **N(1 ≤ N ≤ 10^5)** denoting the total number of cities in Byteland and **Q(1 ≤ N ≤ 10^5)** denoting the number of tasks you have to complete. Next line contains **N** integers denoting the **VAT** of city **i(1 ≤ i≤ N & 1 ≤ VAT_i ≤ 10^5)**. Each of the next **N-1** lines contains two integers **U V (1≤U, V≤N & U ≠ V)** which means there is a direct bidirectional road between city **U** & **V**. Each of the next **Q** lines will contain any one type of tasks described above in the following formats:

1 **U**: Denoting the task type **1** where **1 ≤ U ≤ N**.

2 **U V**: Denoting the task type **2** where **1 ≤ U, V ≤ N**.

Output:

The first line of each test case should contain the case number in the format: **"Case X:"** where **X** is the case number (without quotes). For each of the task type **2**, you have to print two space-separated integers **P Q** where **P** is the closest city from the capital and **Q** is the total vat. Please see the sample input-output for more clarifications.

Sample Input

```
1
5 5
1 2 3 4 5
1 2
1 3
2 4
2 5
2 4 5
1 2
2 1 5
1 5
2 1 5
```

Output for Sample Input

```
Case 1:
2 11
2 8
5 8
```

N.B.: Input file is huge. Please use faster I/O.

Problem C

Input: Standard Input
Output: Standard Output

Binary Search

Binary search is one of the most important tools that is being used in solving problems. You are given an array **S** consists of **N** elements indexed from **1** to **N**. We want to search an element **P** within the array. Using binary search we can solve this problem very easily. Binary_search function is given as follows. This function returns **true** if P is found in the array and returns **false** if P is not found in the array.

```
function Binary_search(S, N, P):
1.    L := 1
2.    R := N
3.    while L ≤ R:
4.        m := floor((L + R) / 2)
5.        if S[m] < P:
6.            L := m + 1
7.        else if S[m] > P:
8.            R := m - 1
9.        else:
10.           return true
11.   return false
```

The given array must remain in sorted order before performing a binary search. But we can still find an element in an unsorted array using binary search algorithm, depending on the array and the element we are looking for. For example, Let an array **S[ ] = {2,3,1}**, we are looking for **P = 3**, using binary search algorithm. Here, **L = 1**, **R = 3**, **m = floor((1+3)/2) = 2**. Since **S[2] = 3**, binary search will return **true**.

In this problem, you are given two integers **N** and **P**. The array **S** consists of **N** distinct integers from **1** to **N**. You have to find how many permutations of the array **S** will return **true** if we search **P** using the given binary search algorithm.

Input

Input starts with an integer **T (1 ≤ T ≤ 50000)** denoting the number of test cases. Following T test cases will contain two integers **N(1 ≤ N ≤ 10000000)** and **P(1 ≤ P ≤ 10000000)**.

Output

For each test case, print the case number followed by the number of permutations of the array which will return **true** using the given binary search algorithm. Since the answer can be very big, print the answer modulo 1000000007 ($10^9 + 7$). Follow the sample I/O for more clarity.

Sample Input

```
3
1 1
3 2
7 5
```

Output for Sample Input

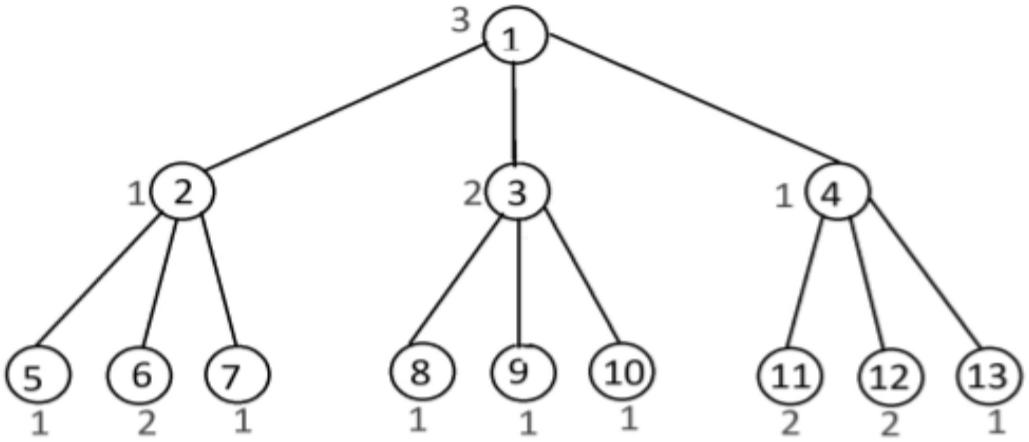
```
Case 1: 1
Case 2: 4
Case 3: 2160
```

Problem D	Input: Standard Input Output: Standard Output
Subtree Query	

You have given a tree of **N** nodes. Nodes are numbered from **1** to **N** and the tree is rooted on the node **1**. Every node of the tree has a value on it.

Subtree rooted at **U** and **V** are similar if they produce the same array after sorting all values of their subtree.

For example,



The number written inside the circle are the node number and number written outside the circle are the value of associative node. (i.e root node has value 3)

Subtree rooted at **2** and **3** are similar, because both of them produce the same array **{1, 1, 1, 2}** after sorting all the values associated with the nodes of those two subtrees. But subtree rooted at **3** and **4** are not similar because subtree rooted at **3** produces the array **{1, 1, 1, 2},** and subtree rooted at **4** produces the array **{1, 1, 2, 2},** and they are not the same array.
Along with the tree you will also given some operation to perform. There are two types of operations

1. **Update(X, V):** add **V** to all the nodes of subtree rooted at **X**.
2. **Query(X, Y):** Check If subtree rooted at **X** and **Y** are similar.

Input

First line of the input will contain a single integer, **T**, denoting the number of test cases. Each test case starts with two space separated integers **N, O**. **N** denotes the number of nodes in the tree and **O** denotes the number of operations.

Next line contains **N** space separated integers, **i-th** of which indicates the associated value of **i-th** node. Next **N-1** lines contain two space separated integer **U, V,** denoting that there is an edge between node **U** and **V**.

Each of next **O** lines contains three integers, describing an operation. First of them is the type of operation, which can be either **1** or **2.** If it is **1**, then the following two integers are **X, V**, denoting an update operation. Otherwise, the following two integers are **X, Y**, denoting a query operation.

Constraints:

- **1 ≤ T ≤ 20**
- **1 ≤ N ≤ 2 x 10^5**
- **-10^5 ≤** value of each node **10^5**
- **1 ≤ O ≤ 2 x 10^5**
- **1 ≤ U, V ≤ N**
- **1 ≤ X, Y ≤ N**
- **-10^5 ≤ V ≤ 10^5**

Summation of **N** over all test cases will be less or equal to **5 x 10^5**. Summation of **O** over all test cases will be less or equal to **5 x 10^5**.

Output

For each test case, print the line without quotation **"Case t:"** Where **t** is the number of the test case starting from **1**. For each **Query(X, Y)** operation print **1** if subtree rooted at **X, Y** are similar; otherwise print **0**, in separate line. Check sample input output for exact formatting.

Sample Input

```
2
13 4
3 1 2 1 1 2 1 1 1 1 2 2 1
1 2
1 3
1 4
2 5
2 6
2 7
3 8
3 9
3 10
4 11
4 12
4 13
2 2 3
2 3 4
1 12 -1
2 3 4
2 1
1 1
1 2
2 1 2
```

Output for Sample Input

```
Case 1:
1
0
1
Case 2:
0
```

N.B.: Input file is huge. Please use faster I/O.

Problem E	Input: Standard Input Output: Standard Output
Balloon Heads	

Have you ever heard of balloon heads? Well, they had a fever and their heads have turned to balloons. As their caretaker, Alice wants to shoot them to put some senses into their (balloon) heads.

There are **n** balloon heads floating on 2d grid. They can be considered as points on the plane. You are given the coordinates of them at time **0**. At the beginning of each second, a balloon head goes up one unit. That is, it goes from **(x, y)** to **(x, y+1)**. But if it's y-coordinate becomes a multiple of **h** then it stays still for **extra t** seconds.

For example, if **h = 5** and **t = 3**, then a balloon head starting at **(3, 8)** at second **0**, will go to **(3, 9)** at second **1, (3, 10)** at second **2**, and then it stays at **(3, 10)** for another **3** seconds (at second **3, 4** and **5**). Then at **6th** second, it will move to **(3, 11)**.

To put the balloon heads to sleep, Alice has decided to perform **q** operations. i'th of her operation can be one of the following two types:

1 T_i : Here T_i is used to generate 4 more parameters X_i , Y_i , DX_i , DY_i . Pseudocode to generate them is mentioned below:

seed := T_i
seed := (seed * 1103515245 + 1234) % 100000001 , X_i := (seed%lim)+1
seed := (seed * 1103515245 + 1234) % 100000001 , Y_i := (seed%lim)+1
seed := (seed * 1103515245 + 1234) % 100000001 , DX_i := (100000000 - (seed%lim))
seed := (seed * 1103515245 + 1234) % 100000001 , DY_i := (100000000 - (seed%lim))

Value of variable **lim** in the above pseudocode is given in input before all queries, which is the same for all queries.

This means at **T_i-th** second, Alice shoots all the points inside and on the border of the rectangle having opposite corners at points **(X_i , Y_i)** and **(X_i + DX_i , Y_i + DY_i)**. It is guaranteed that for the ith and jth query, if **i < j** then **T_i < T_j** . If a balloon head moves at T_i-th second, then its position after the move will be considered for this operation.

2 B_i : This is a query Alice makes to you. The query asks how many times the **B_i-**th balloon head has been shot before this query is made. The balloons are indexed from 1 to n in the order they are listed in input.

Input

The first line of the input contains an integer **T** which denotes number of test cases. Description of T test cases follows.
The first line of each test case has two integers **n** and **q**, which indicates the total number of balloon heads and number of operations Alice performs respectively. The second line of the test case has two integers **h** and **t**, which indicates the constants as mentioned in the statement.
Next line has four integers **sx, sy, dsx, dsy .** From these four numbers, 2n values are generated using the following pseudocode:

```
seed := sx
i := 1
WHILE i ≤ n:
        x_i := sx + (seed % (dsx-sx+1)) , seed := (seed * 1103515245 + 1234) % 100000001
        y_i := sy + (seed % (dsy-sy+1)) , seed := (seed * 1103515245 + 1234) % 100000001
                IF y_i % h > 0 THEN
                i := i+1
        ENDIF
ENDWHILE
```

Here $\mathbf{x_i}$, $\mathbf{y_i}$ indicate coordinate at time **0** of **i-th** balloon head. It is guaranteed that $\mathbf{y_i}$ is not a multiple of **h**.
Next line contains an integer **lim** which will be used to generate query parameters as specified in the statement.

Each of the next q lines indicates one of Alice's operations as mentioned in the statement. It is guaranteed that there will be at least one query of the second type.

Constraints

- **1 ≤ T ≤ 13**
- **1 ≤ n ≤ 200,000, summation of n over all test cases does not exceed 800,000**
- **1 ≤ q ≤ 200,000, summation of q over all test cases does not exceed 800,000**
- **2 ≤ h ≤ 5**
- **1 ≤ t ≤ 5**
- **1 ≤ sx, sy, dsx, dsy ≤ 100,000,000 , sx ≤ dsx, sy ≤ dsy**
- **1 ≤ x_i , y_i ≤ 100,000,000 , y_i is not divisible by h**
- **1 ≤ lim ≤ 100,000,000**
- **1 ≤ T_i , X_i , Y_i ,DX_i ,DY_i ≤ 100,000,000**
- **1 ≤ B_i ≤n**

Output

The first line of each test case should contain the case number in the format: **"Case TC:"** where **TC** is the case number (without quotes).Then, for each of the 2nd type queries, output it's answer in one line. Follow the sample output for more clarity.

Sample Input	Output for Sample Input
1 3 6 2 1 40000000 40000000 60000000 60000000 100000000 1 1 2 1 1 3 2 2 1 8 2 3	Case 1: 0 1 1

N.B.: Input file is huge. Please use faster I/O.

Problem F	Input: Standard Input Output: Standard Output
Mathematical Marathon	

The city of Alexa hosts a marathon every year. The rules of the marathon are simple, the course consists of a series of **N** connected checkpoints joined by **M** roads. Each person starts from a starting checkpoint and must reach a designated destination checkpoint. Now people need water to run. For each litre of water consumed, a runner can run 1 kilometer. Each person starts by consuming **X** litres of water at the starting-point. Water fountains are available at every checkpoint but there are no water fountains in the middle of the roads.

However this year the problem solver guild's annual olympiad was scheduled on the same day as the marathon. During this marathon most of the city remains closed, but the people from the guild are refusing to cancel their event. After much chaos from the guild the city council has agreed to combine the olympiad with the marathon. Each person running the marathon will now have to solve problems while running.

Each runner will be given a maximum capacity of water that he can drink. This amount will be given before the race starts and it will be same for every runner. Initially this capacity is **X**, but after a runner uses a road of length **K**, his new capacity becomes $\mathbf{X_{new} = gcd(X_{old}, K)}$ so he can only consume $\mathbf{X_{new}}$ amount of water at that checkpoint. A runner will not drink more than exactly what is needed to cross a road, even if the maximum capacity allows it, as the extra weight may slow him down. and as there are no checkpoints in the middle of the road, so if a road has length $\mathbf{K > X_{new}}$ he cannot use that road. Given these constraints, the runners have come to you for help. Help them find the shortest route for their race.

Input

The first line of the input is an integer **T**, denoting the number of test cases. Each test case starts with a line consisting of 2 integers **N** and **M** which indicate the number of checkpoints and connecting roads respectively. **i-th** of the following **M** lines contain 3 integers $\mathbf{U_i, V_i, W_i}$. Here $\mathbf{U_i, V_i}$ indicates the 2 ends of an undirected road, and $\mathbf{W_i}$ indicates it's length. There may be multiple roads connecting the same checkpoints. On the last line for the test case, you are given 3 integers, the starting checkpoint **ST**, destination checkpoint **EN**, and the initial capacity **X**.

Output

For each test case, print the case number followed by a single integer denoting the length of the shortest path from the source to the destination, or 'impossible' if there are no valid paths. See sample output for clarification.

Constraints:

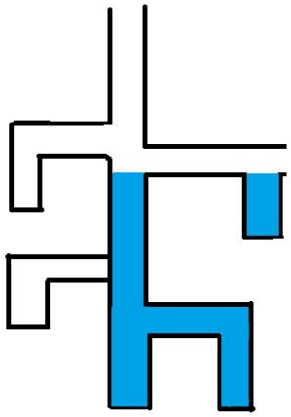
- $\mathbf{1 \le T \le 10}$
- $\mathbf{1 \le N, M, X, W_i \le 10^5}$
- $\mathbf{1 \le ST, EN, U_i, V_i \le N}$

Sample Input	Output for Sample Input
2 6 7 1 2 4 1 4 1 2 3 2 2 4 4 3 6 4 4 5 4 5 6 4 1 6 12 5 5 1 2 4 2 3 1 3 5 3 1 4 6 4 5 10 1 5 12	Case 1: 16 Case 2: impossible

N.B.: Input is huge, please use faster I/O methods.

Problem G	Input: Standard Input Output: Standard Output
Water Puzzle	

We all are familiar with a puzzle, where some pipe is connected with other. Sometimes connections are blocked and sometimes pipes have leakage. When we pour water from top, some part of pipes may have water and some part of pipes may not have any water because of leakage. Below figure is an example.



In the figure above, if we pour water from the top, after a certain amount of time water will leak from the right side. In this problem you will be given a similar structure as a grid with N rows and M columns. The input grid will consist of only '.' and '#' where '.' means empty cell and '#' means blocked cell. Each empty cell can store 1 unit of water.

To clarify more:

- At each unit time one unit water will fall from the top of M columns and due to gravity water will always try to move downwards until it finds any barrier.
- If it finds a barrier, water will be distributed evenly in both directions and will move to surface having lower height due to gravity.
- So, the unit water may fall from the structure due to leakage or it may be stored in some surface bounded by blocks and the height of water level will increase.
- You can simply assume, any nonempty cell as barrier. Also, leaks are so powerful that they can release any amount of water at a single time.
- So, it is guaranteed that after a certain period of time, the amount of water in the structure will become stable(as it is shown in the above figure), that is, no more water will be stored in the structure (water will fall due to leakage or will overflow).

You have to find out the amount of water that will be stored in the structure after being stable.

Input

Input begins with an integer **T**, denotes the number of test cases. Each case begins with two integers **N** and **M** denoting the number of rows and number of columns. Then, there will be N lines with M characters in each line. There will be only '.' and '#' in the grid.

Constraints:
1 ≤ T ≤ 10, 5 ≤ N ≤ 1000, 5 ≤ M ≤ 1000
ΣN*M over all test cases ≤ 2,000,000

Output

For each case of input print one line with format **"Case t: X"**. **t** is the case number starting from 1 and X is the maximum amount of water the structure may contain.

Sample Input

```
2
14 13
....#.#......
#####.#......
#.....#######
#.###........
#.#.#.####.##
###.#.#..#.#.
....#.#..###.
....#.#......
....#.######.
....#......#.
....#.####.#.
....#.#..#.#.
....#.#..#.#.
....###..###.
15 13
....#.#......
#.###.#......
#.....#######
#.###........
#.#.#.####.##
###.#.#..#.#.
....#.#..###.
....#.#......
#.###.######.
#...#......#.
#.###.###..#.
#.#.#.#..#.#.
###.#.#..###.
....#.#..#...
....########.
```

Output for Sample Input

```
Case 1: 19
Case 2: 22
```

Problem H	Input: Standard Input Output: Standard Output
State Cup Madness	

In Australia there are **6** states, Western Australia, Queensland, South Australia, New South Wales, Victoria and Tasmania. Regular sporting events are being arranged among these states.

Currently a cricket league is going on. Mr. X is a big fan of Tasmania. He is very eager to know what is the best possible outcome for his team optimistically. He hired you for doing this. As you are a great programmer, you solved this problem instantly.

But another football league is still going on. Some of the states had taken part in this tournament with different franchise names. Each state could have no more than one team. Some states might not even participate in this tournament. But Tasmania has a team for sure in this tournament. So, Mr. X has appointed you to find out the best possible result for the Tasmanian team.

Match Rules:

- A match is played between two teams.
- Each team score some goals. The team that scores maximum goals wins the match. If they both score the same amount of goals, then it's a draw.
- A win gives you **3** points, draw gives you **1** point and losing gives you **0** point.

Tournament Rules:

- Each team will play a match with other teams in their home.
- So, a pair of teams will face each other in home-away basis.
- Each match is independent. Result of home match will not affect the away match.
- If there are **N** teams in the tournament, a team will play **2** $\times$ **(N - 1)** matches. So, there will be total **N** $\times$ **(N - 1)** matches.
- Teams are sorted according to their total amount of points, the more the better.
- If two or more teams have the same amount of points, they are sorted by their amount of (goal scored - goal conceded), the more the better.
- Still, If two or more teams have the same place, they are sorted by their amount goal scored, the more the better.
- Even if two or more teams have the same place, they are sorted by their franchise names, lexicographically smaller is better.

Already some matches have been played. The scorecard of those match are given to you. You have to find the maximum place that the Tasmanian team can possibly achieve and suitable point difference with the top most other team. If, the tasmanian team can be champion, then they want to maximize the point difference with the second team. Otherwise they want to minimize the point difference with the top team.

Input

Input starts with an integer, **T**, denoting the number of test cases. For each test case there will be two integer **N** and **M**, denoting number of teams in that tournament and number of already played matches. The next line contains **N** space separated words denoting the team names, the first team is the Tasmanian team. The next **M** lines contain scorecard for each already played match in this given format:

team1 g1 - g2 team2

Here, **team1** denotes the home team and **team2** denotes the away team. **g1** denotes the number of goals scored by **team1** and **g2** denotes the number of goals scored by **team2**.

Output

For each test case, print a single line in this format:

Case X: P D

Here, **X** denotes the number of the test case. **P** denotes the maximum achievable place for the Tasmanian team and **D** denotes the suitable point difference.

Constraints

1 ≤ T ≤ 1000

2 ≤ N ≤ 6

0 ≤ M < N * (N - 1)

1 ≤ length_of_team_name ≤ 20 and all characters will be lowercase english letter

0 ≤ g1, g2 ≤ 20

It is guaranteed that the Tasmanian team has at least one match left, all team names are distinct. There is no limit for goals scored for upcoming matches (have to be non-negative for sure).

Sample Input	Output for Sample Input
3 2 1 arzentina brajil arzentina 2 - 3 brajil 3 4 arzentina brajil zermany brajil 0 - 7 zermany zermany 4 - 0 arzentina arzentina 2 - 3 brajil arzentina 0 - 2 zermany 3 4 arzentina brajil zermany brajil 0 - 7 zermany zermany 4 - 0 arzentina arzentina 2 - 2 brajil arzentina 0 - 2 zermany	Case 1: 1 0 Case 2: 2 9 Case 3: 2 5

Problem I	Input: Standard Input Output: Standard Output
King of Africa	

You are the king of the gridland in africa. There are **N** tribes lives in your country. To make the service management of the country you divide the whole country land to **NxN** square grid. To send proper service to every tribe you want to rearrange the country's tribe in a way that no two cell in a row or column will contain same tribe members. In order to do so,

- You can move people from one cell to another cell if they have same tribe. For example, let cell**(x1, y1)** and cell **(x2,y2)** have people of the same tribe. So, if you move people from cell **(x1,y1)** to cell**(x2,y2)** it will left cell **(x1,y1)** as empty.
- You can not move any tribe to any empty cell.

To do this he can just pick **N** cell, one of each tribe and move all the **people** of same tribes to those cells. But this will make the country looks like empty. So he wants rearrange in a way that he need to empty the minimum number of cells. If there are several such ways, he will choose the cells in a way that he needs to move the minimum number of people.

As you are the best friend and cabinet members of the king so he asked you to help him to achieve the constraints.

Input

First line, there will be an integer **T (1≤T≤100)** which is the number of test cases.

Every test case starts with a number 1**≤N ≤ 100**.

Next **N** line contain **N** integer which denoes **NxN** grid **A**. where element in **i-th** row and **j-th** column **A[i,j]** denotes the tribe number living in that cell.

Next **N** line contain **N** integer which denoes **NxN** grid **B**. where element in **i-th** row and **j-th** column **B[i,j]** denotes the number of people of tribe **A[i,j]** living in that cell.

Constraints

- **1 ≤ A[i, j] ≤ N**
- **1 ≤ B[i, j] ≤ 10^5**

Output

For each test case you need to print a line **"Case t: X Y"**. where **t** is the case number, **X** is the minimum number of cells king need to empty and **Y** is the minimum number of people king need to move.

Sample Input

```
1
2
1 1
2 2
1 3
2 3
```

Output for Sample Input

```
Case 1: 2 3
```

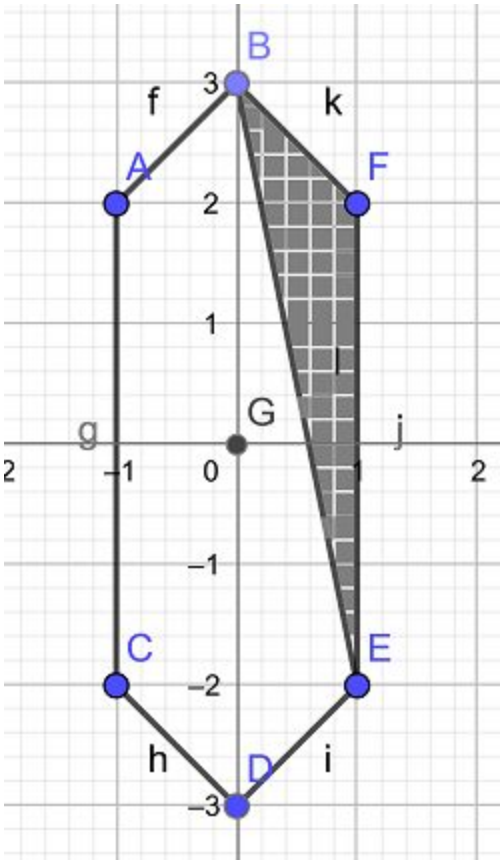
N.B.: Input is huge, please use faster I/O methods.

Problem J

Input: Standard Input
Output: Standard Output

No Points in Polygon

You will be given a convex polygon and several points which lie completely inside this given polygon. You need to make a cut connecting any two vertices of the given polygon such that area of the new polygon that contains none of the given points (*inside or on its edge*) is maximized.



Note: A point is considered to be completely inside a polygon if the point lies strictly within the enclosed area, and not on any of the edges.

Input

First line of input contains an integer **T (1 ≤ T ≤ 20)**, the number of test cases.

For each test case, first line contains two integers, **N** and **M**. Here, **N (3 ≤ N ≤ 3x10^5)** is the number of vertices of the convex polygon, and **M (1 ≤ M ≤ 3x10^5)** is the number of points given inside the polygon.

Each of the following **N** lines contains two integers **(x_i, y_i)**, describing a vertex of the given convex polygon. Vertices will be given in a counter clockwise order.

Finally, each of the following **M** lines contains two integers **(x_j, y_j)**, describing a point that lies within the given convex polygon.

For all coordinates **(x, y)**, it is given that **$|x|, |y| \leq 10^9$**. And, it is guaranteed that the given convex polygon will always have a positive area.

Output

For each test case, print a line containing the test case number starting from **1** and an integer **X**, where **X = TWICE** the maximum possible area as described in the problem statement. Check sample input and output for details.

Sample Input

```
2
5 3
0 1
3 0
4 2
2 3
0 3
2 1
3 1
3 2
6 1
0 3
-1 2
-1 -2
0 -3
1 -2
1 2
0 0
```

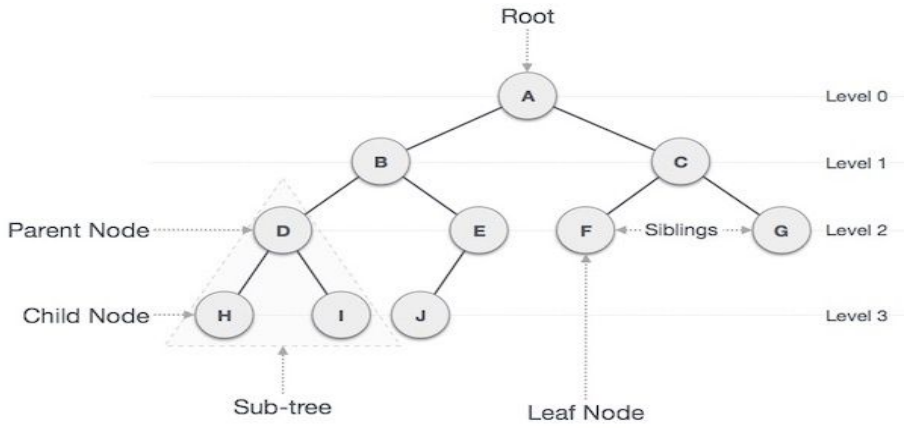
Output for Sample Input

```
Case 1: 4
Case 2: 4
```

N.B.: Input file is huge. Please use faster I/O.

Problem K	Input: Standard Input Output: Standard Output
Depth of Binary Tree	

Tree is a data structure where the nodes are connected by edges. In each tree, there will be a node called the root and every node is connected to the root directly or indirectly. All the nodes will have zero or more nodes connected to it directly as it's children. The depth of a tree is denoted by the highest level in a tree. Which is the maximum distance for any node from root of that tree. A binary tree is a tree in which each node has at most two children, which are referred to as the left child and the right child. The following picture contains a binary tree with depth = 3.



You will be given **N,** which denotes the total number of nodes in a binary tree. You have to print what is the maximum and minimum possible depth a binary tree can have if it has **N** nodes.

Input

The first line of the input contains an integer **T(1 ≤ T ≤ 20)**, denoting the number of test cases. Then in next **T** lines, **T** test cases will follow. Each line will have **N(1 ≤ N ≤ 20)**.

Output

For each test case, print the results in *"**Case I: MAX MIN**" format*, where ***I*** is the case number starting from **1** and **MAX, MIN** are maximum and minimum possible depth.

Sample Input

```
3
1
2
3
```

Output for Sample Input

```
Case 1: 0 0
Case 2: 1 1
Case 3: 2 1
```