Using data compression technique, the long string "ruiruiruiruiruirui" can be compressed into "(ruirui)3" or "(rui)6". To simplify the technique, multiple compressions are not allowed. For example, we **don't allow** to compress the string "princessruiruiprincessruirui" into "(princess(rui)2)2".

Now for given string S and k patterns using the mentioned technique, we want to distinguish each pattern which is a prefix of S. We emphasize that a pattern as a string can be cyclic shifted. For example, a pattern "abcd" can be shifted into "bcda", "cdab" or "dabc".

Input

The first line contains an integer t ($1 \leq t \leq 13$) which is the number of test cases.

For each test case, the first line is the string S. The second line contains the integer $k(1 \leq k \leq 10)$ and following k lines list the patterns, labelled from 1 to k. The string S and all patterns only contain lower-case letters, numbers and parentheses. Numbers of replication are positive integers no more than 2×10^8. The length of S is no more than 11000. The length of each pattern is no more than 11000 as well. We guarantee that the actual length of each T (the length of the original pattern without compression) would be smaller than the actual length of S (the length of original S without compression). The length of each substring given in parentheses is no more than 10.

Output

For each test case, output the sum of labels' squares for patterns as the prefix of S.

Hint:

In the first test case, the string S is '`zrzrzrzrruiruicumt`'. The first pattern '`zrzrzrzr`' is a prefix of S. The third one '`zrzrzrzrru`' is a prefix of S as well. The fourth on can be shifted into '`zrzrzr`' and the fifth one can be shifted into '`zrzrzrzrr`'. The sum of squares is $1^2 + 3^2 + 4^2 + 5^2 = 51$.

Sample Input

```
3
z(rz)3r(rui)2cumt
5
(zr)4
zrzrrui
zr(zr)2z(r)2u
(rz)3
(zr)2z(r)2zr
(ab)2aab(aba)3(ba)2(zhang)940712
4
(babaa)2(baa)2
(aabab)2
(ab)3
(aba)2(ab)3a(ab)2(a)2b
(a)100b(a)100c(a)100d
1
(a)100d(a)100c(a)100b
```

Sample Output

```
Case #1: 51
Case #2: 21
Case #3: 0
```

Ladies and gentlemen, please sit up straight.

Don't tilt your head. I'm serious.

For n given strings $S_1, S_2, \cdots, S_n$, labelled from 1 to n, you should find the largest i $(1 \leq i \leq n)$ such that there exists an integer j $(1 \leq j < i)$ and S_j is not a substring of S_i.

A substring of a string S_i is another string that occurs **in** S_i. For example, "ruiz" is a substring of "ruizhang", and "rzhang" is not a substring of "ruizhang".

Input

The first line contains an integer t $(1 \leq t \leq 50)$ which is the number of test cases. For each test case, the first line is the positive integer n $(1 \leq n \leq 500)$ and in the following n lines list are the strings $S_1, S_2, \cdots, S_n$. All strings are given in lower-case letters and strings are no longer than 2000 letters.

Output

For each test case, output the largest label you get. If it does not exist, output '`-1`'.

Sample Input

```
4
5
ab
abc
zabc
abcd
zabcd
4
you
lovinyou
aboutlovinyou
allaboutlovinyou
5
de
def
abcd
abcde
abcdef
3
a
ba
ccc
```

Sample Output

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

Given a simple unweighted graph G (an undirected graph without self-loops or multiple edges) with n nodes and m edges. Let T be a spanning tree of G. We say that a cut in G two-respects T if and only if it **cuts just two edges of** T.

Since love needs good faith and hypocrisy return for only grief, you should find the minimum cut of graph G two-respecting given spanning tree T. To simplify the problem, we guarantee that for each edge $(u, v) \notin T$ in graph G, the unique path in T between u and v must pass through the node 1.

Input

The input contains several test cases. The first line of the input is a single integer t ($1 \leq t \leq 25$) which is the number of test cases. Then t test cases follow.

Each test case contains several lines. The first line contains the integer n ($3 \leq n \leq 20000$) and the integer m ($n - 1 \leq m \leq 100000$). The following $n - 1$ lines describe the spanning tree T and each of them contains two integers u and v corresponding to an edge. The following $m - n + 1$ lines describe the undirected graph G and each of them contains two integers u and v corresponding to an edge which is no in the spanning tree.

The sum of m for all test cases would not be larger than 500000.

Output

For each test case, you should output the minimum cut of graph G two-respecting given spanning tree T.

Sample Input

```
2

8 14
1 2
2 3
1 4
4 5
1 6
6 7
6 8
3 4
2 5
5 7
1 7
2 6
2 8
3 8

4 6
1 2
1 3
1 4
2 3
3 4
4 2
```

Sample Output

```
Case #1: 3
Case #2: 4
```

n pagodas were standing erect in Hong Jue Si between the Niushou Mountain and the Yuntai Mountain, labelled from 1 to n. However, only two of them (labelled a and b, where $1 \leq a \neq b \leq n$) withstood the test of time.

Two monks, Yuwgna and Iaka, decide to make glories great again. They take turns to build pagodas and Yuwgna takes first. For each turn, one can rebuild a new pagodas labelled i ($i \notin \{a, b\}$ *and* $1 \leq i \leq n$) if there exist two pagodas standing erect, labelled j and k respectively, such that $i = j + k$ or $i = j - k$. Each pagoda can not be rebuilt twice.

This is a game for them. The monk who can not rebuild a new pagoda will lose the game.

Input

The first line contains an integer t ($1 \leq t \leq 500$) which is the number of test cases. For each test case, the first line provides the positive integer n ($2 \leq n \leq 20000$) and two different integers a and b.

Output

For each test case, output the winner ('`Yuwgna`' or '`Iaka`'). Both of them will make the best possible decision each time.

Sample Input

```
16
2 1 2
3 1 3
67 1 2
100 1 2
8 6 8
9 6 8
10 6 8
11 6 8
12 6 8
13 6 8
14 6 8
15 6 8
16 6 8
1314 6 8
1994 1 13
1994 7 12
```

Sample Output

```
Case #1: Iaka
Case #2: Yuwgna
Case #3: Yuwgna
Case #4: Iaka
Case #5: Iaka
Case #6: Iaka
Case #7: Yuwgna
Case #8: Yuwgna
Case #9: Iaka
Case #10: Iaka
Case #11: Yuwgna
Case #12: Yuwgna
Case #13: Iaka
Case #14: Yuwgna
Case #15: Iaka
Case #16: Iaka
```

A graph G with $n \times m$ nodes forms a $n \times m$ grid with $n \times m$ vertices. $(n-1) \times m$ weighted edges connect the vertices of adjacent rows, and $n \times (m-1)$ weighted edges connect the vertices of adjacent columns.

A spanning tree of graph G is a subgraph that is a tree and connects all the vertices together. A minimum spanning tree (MST) or minimum weight spanning tree is then a spanning tree with weight less than or equal to the weight of every other spanning tree. Graph G has many different minimum spanning trees. For each MST T, the $LRdeg(u)$ of node u is defined as the number of nodes, in the previous column or the previous row connecting with u, plus one. And we define $\tau(T) = \prod_u LRdeg(u)$ as the product of $LRdeg(u)$ for all nodes.

Your mission is to find the weight of the minimum spanning tree of graph G, and count $\tau(T)$ of all minimum spanning trees. Two MST(s) are considered different if they contain different subsets of edges.

Input

The input contains several test cases. The first line of the input is a single integer t $(1 \leq t \leq 32)$ which is the number of test cases. Then t test cases follow.

For each test case, the first line contains the two integers n $(1 \leq n \leq 800)$ and m $(1 \leq m \leq 7)$. Each line of the next n lines contains $m-1$ integers, which describe the weights of edges connecting the vertices of adjacent columns. And each line of the next $n-1$ lines contains m integers, which describe the weights of edges connecting the vertices of adjacent rows. The weights of edges are no more than 10.

Output

For each test case, you should output two integers in one line. The first one is the weight of the minimum spanning tree. The second one is the sum of $\tau(T)$ for all different minimum spanning trees, modulo $10^9 + 7$.

Sample Input

```
2

2 5
9 8 5 6
4 6 2 3
1 7 8 3 8

5 5
8 10 5 4
1 7 7 7
5 4 5 5
3 2 2 2
8 7 8 3
8 5 7 8 6
10 3 2 4 3
8 7 2 8 9
9 4 8 3 9
```

Sample Output

```
Case #1: 37 288
Case #2: 96 4478976
```

There are m stones lying on a circle, and n frogs are jumping over them. The stones are numbered from 0 to $m-1$ and the frogs are numbered from 1 to n. The i-th frog can jump over exactly a_i stones in a single step, which means from stone $j \ mod \ m$ to stone $(j + a_i) \ mod \ m$ (since all stones lie on a circle).

All frogs start their jump at stone 0, then each of them can jump as many steps as he wants. A frog will occupy a stone when he reach it, and he will keep jumping to occupy as much stones as possible. A stone is still considered "occupied" after a frog jumped away. They would like to know which stones can be occupied by at least one of them. Since there may be too many stones, the frogs only want to know the sum of those stones' identifiers.

Input

There are multiple test cases (no more than 20), and the first line contains an integer t, meaning the total number of test cases.

For each test case, the first line contains two positive integer n and m - the number of frogs and stones respectively ($1 \leq n \leq 10^4, \ 1 \leq m \leq 10^9$).

The second line contains n integers $a_1, a_2, \cdots, a_n$, where a_i denotes step length of the i-th frog ($1 \leq a_i \leq 10^9$).

Output

For each test case, you should print first the identifier of the test case and then the sum of all occupied stones' identifiers.

Sample Input

```
3
2 12
9 10
3 60
22 33 66
9 96
81 40 48 32 64 16 96 42 72
```

Sample Output

```
Case #1: 42
Case #2: 1170
Case #3: 1872
```

The discovery of anti-gravitations technology changed the world. The invention of anti-gravitation shoes (Grav-shoes) enables people to fly in the sky freely. This led to the rise of a new sky sport: **"Flying Circus"**.

Utilizing Grav-shoes and personal flying suits, competitors battle it out in a special field, where they compete scoring obtain m points within a certain time limit. The field is a square with edge length 300 meters. Moreover, there are four buoys floating at each corner of the square. Four buoys are numbered as $1, 2, 3, 4$ in clockwise order.

Two players start at buoy #1. When game begin, they will try to touch four floating buoys in clockwise order. (Since buoy #1 is the start point, the first buoy they need to touch will be buoy #2, and after that, they need to touch buoy #3, #4, #1 in order) **Note that they could fly freely in the field, even fly inside the square field.**

Under two situations the player could score one point.

- **1. If you touch a buoy before your opponent, you will get one point.** For example if your opponent touch the buoy #2 before you after start, he will score one point. So when you touch the buoy #2, you won't get any point. Meanwhile, you cannot touch buoy #3 or any other buoys before touching the buoy #2.
- **2. Ignoring the buoys and relying on dogfighting to get point.** If you and your opponent meet in the same position, you can try to fight with your opponent to score one point. For the proposal of game balance, two players are not allowed to fight before buoy #2 is touched by anybody.

There are three types of players.

- **Speeder**: As a player specializing in high speed movement, he/she tries to avoid dogfighting while attempting to gain points by touching buoys.
- **Fighter**: As a player specializing in dogfighting, he/she always tries to fight with the opponent to score points. Since a fighter is slower than a speeder, it's difficult for him/her to score points by touching buoys when the opponent is a speeder.
- **All-Rounder**: A balanced player between Fighter and Speeder.

There will be a training match between Asuka (All-Rounder) and Shion (Speeder). Since the match is only a training match, the rules are simplified: **the game will end after the buoy #1 is touched by anybody.** Shion is a speed lover, and his strategy is very simple: touch buoy #2, #3, #4, #1 along the shortest path.

Asuka is good at dogfighting, so she will always score one point by dogfighting with Shion, and **the opponent will be stunned for T seconds after dogfighting.** Since Asuka is slower than Shion, she decides to fight with Shion for only one time during the match. It is also assumed that if Asuka and Shion touch the buoy in the same time, the point will be given to Asuka and Asuka could also fight with Shion at the buoy. We assume that in such scenario, the dogfighting must happen after the buoy is touched by Asuka or Shion.

The speed of Asuka is V_1 m/s. The speed of Shion is V_2 m/s. Is there any possibility for Asuka to win the match (to have higher score)?

Input

The first line contains an integer t ($0 < t \leq 1000$), followed by t lines. Each line contains three double T, V_1 and V_2 ($0 \leq V1 \leq V2 \leq 2000, 0 \leq T \leq 2000$) with no more than two decimal places, stands for one case.

Output

If there exist any strategy for Asuka to win the match, output '`Yes`', otherwise, output '`No`'.

Hint:

Asuka could fly to the mid point of the edge between buoy #2 and buoy #3, and wait there until Shion come. Asuka will fight with Shion when they meet. As a result, Shion will be stunned (It means Shion cannot move for 100 seconds.) Then Asuka could fly back to buoy #2, since the buoy #2 is touched, she will get no point by touching buoy #2. But after that, she could fly to buoy #3, #4, #1 along the edge and get three points.

Sample Input

```
2
1 10 13
100 10 13
```

Sample Output

```
Case #1: No
Case #2: Yes
```

Ruirui and Doc are playing an interesting game on a chessboard with n rows and m columns. The rows are numbered from 1 to n from top to bottom, and the columns are numbered from 1 to m from left to right. There are some broken grids on the chessboards, which a chess cannot move in. Firstly, Doc gives Ruirui a sequence of commands, each command is of one of four following forms:

- **Move Up**: moving from grid (x, y) to grid $(x-1, y)$;
- **Move Down**: moving from grid (x, y) to grid $(x+1, y)$;
- **Move Left**: moving from grid (x, y) to grid $(x, y-1)$;
- **Move Right**: moving from grid (x, y) to grid $(x, y+1)$.

Then Ruirui puts a single chess on a grid of the chessboard

Ruirui will move the chess by Doc's commands in sequence. If the chess will be out of boarder or in a broken grid after a move, she omits this command and **go on** to consider the next one until the last command. Now Ruirui wants to find the grid which the chess will be in the end.

Input

The first line contains a single integer T ($1 \le T \le 10$), which indicates the number of test cases. Then T test cases follow.

For each test case, the first line contains 4 integers n, m, o and l ($1 \le n, m, o, l \le 1000$) representing the number of rows, the number of columns, the number of broken grids and the length of Doc's command sequence.

Next o lines, each line contains two integers i and j describing the position of broken grid.

The last line contains Doc's command sequence, it's a string of length l with each character being one of {`U`,`D`,`L`,`R`} denoting Move Up, Move Down, Move Left and Move Right respectively.

Output

For each test case, for each unbroken grid (i, j), assume a chess started at (i, j) would stop at $(x(i,j), y(i,j))$, output the sum of $(i - x(i,j))^2 + (j - y(i,j))^2$ (over all unbroken (i, j)).

Sample Input

```
2
5 5 5 5
2 3
5 1
5 5
4 4
3 5
RRRLR
10 10 10 10
2 6
3 8
7 2
5 3
4 3
3 2
7 9
6 8
9 10
10 6
DLLDRRURLR
```

Sample Output

```
Case #1: 49
Case #2: 241
```

Given the finite **multi-set** A of n pairs of integers, an another finite **multi-set** B of m triples of integers, we define the product of A and B as a **multi-set**

$$\begin{aligned} C &= A * B \\ &= \{\langle a, c, d\rangle \mid \langle a, b\rangle \in A,\ \langle c, d, e\rangle \in B \text{ and } b = e\} \end{aligned}$$

For each $\langle a, b, c\rangle \in C$, its BETTER set is defined as

$$\begin{aligned} &BETTER_C(\langle a, b, c\rangle) \quad = \\ &\{\langle u, v, w\rangle \in C \quad \mid \quad \langle u, v, w\rangle \neq \langle a, b, c\rangle,\ u \geq a,\ v \geq b,\ w \geq c\} \end{aligned}$$

As a **multi-set** of triples, we define the TOP subset (as a multi-set as well) of C, denoted by $TOP(C)$, as

$$TOP(C) = \{\langle a, b, c\rangle \in C \mid BETTER_C(\langle a, b, c\rangle) = \emptyset\}$$

You need to compute the size of $TOP(C)$.

Input

The input contains several test cases. The first line of the input is a single integer t ($1 \leq t \leq 10$) which is the number of test case. Then t test cases follow.

Each test case contains three lines. The first line contains two integers n ($1 \leq n \leq 10^5$) and m ($1 \leq m \leq 10^5$) corresponding to the size of A and B respectively. The second line contains $2 \times n$ nonnegative integers

$$a_1, b_1, a_2, b_2, \cdots, a_n, b_n$$

which describe the multi-set A, where $1 \leq a_i, b_i \leq 10^5$. The third line contains $3 \times m$ nonnegative integers

$$c_1, d_1, e_1, c_2, d_2, e_3, \cdots, c_m, d_m, e_m$$

corresponding to the m triples of integers in B, where $1 \leq c_i, d_i \leq 10^3$ and $1 \leq e_i \leq 10^5$.

Output

For each test case, you should output the size of set $TOP(C)$.

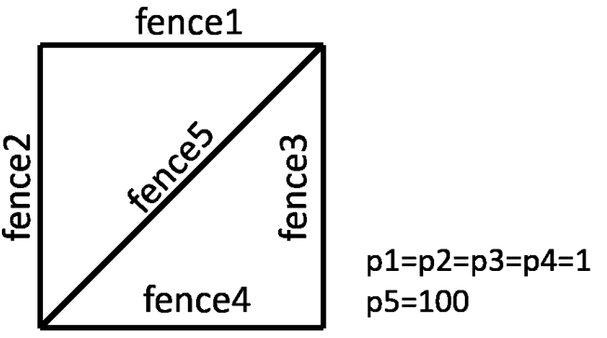
Sample Input

```
2
5 9
1 1 2 2 3 3 3 3 4 2
1 4 1 2 2 1 4 1 1 1 3 2 3 2 2 4 1 2 2 4 3 3 2 3 4 1 3
3 4
2 7 2 7 2 7
1 4 7 2 3 7 3 2 7 4 1 7
```

Sample Output

```
Case #1: 5
Case #2: 12
```

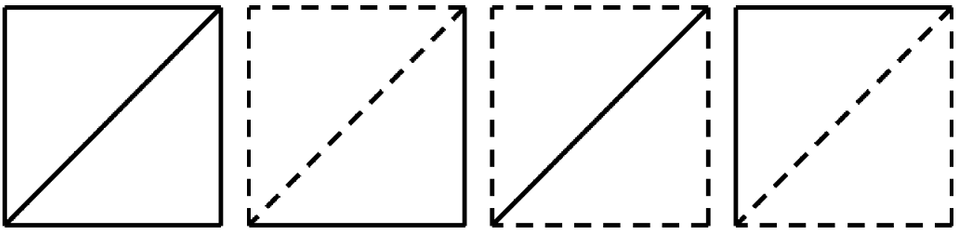
Farmer John built n fences on his farm. The i-th fence is a line segment in the plane whose two endpoints are (x_{1i}, y_{1i}) and (x_{2i}, y_{2i}). Two different fences can only intersect at their endpoints. The following picture shows an example layout of fences.



To make cows happy, John will decorate the fences with rope lights. A fence can hang several rope lights paralleling to itself. However, a light can not be used alone. John can only make a decoration by using some lights together (we call it a **string of lights**) and these lights will form a circle to decorate some fences (a light corresponds to a fence). In the example, he can use three lights decorate three fences numbered $1, 2, 5$ which form a circle.

A string of lights which decorate fences numbered $k_1, k_2, \cdots, k_m$ costs John $\sum_{i=1}^{m} p_{k_i}$ dollars. The total cost is the sum of costs of each string. Notice that the cost for more than one **strings of lights** should be accumulated. In the example, suppose that he decorate the fences with two strings of lights. One string decorate three fences numbered $1, 2, 5$ and the other string decorate three fences numbered $1, 2, 3, 4$. Then the total cost should be $2 \times p_1 + 2 \times p_2 + p_3 + p_4 + p_5 = 106$ dollars.

John can control the switch of **each string** of lights. He can switch on **all lights on that string** or switch off all of them, but he cannot switch on some of they while keeping other lights on that string switched off. For a fence, if even number of strings on it are switched on, then all the lights on that fence will not work; otherwise, all of them work. We define the shape of lights as a combination of status of each fence (considering the fences lit or not). A shape of lights might be achieved by arrangements of lights. The following picture shows all possible shape of lights in the example. (Dash lines stands for fences where lights work and the solid stands for fences where lights don't work.)



The cows love lights, but they are fickle too. They want the lights arranged to form different shapes from day to day, so John need to make arrangement for decorations so that **every possible shapes** can be formed by switching on **some of** the strings of lights and switching off the others. What is the minimum money John must spend to meet needs of cows.

In the example, he can decorate the fences with two strings of lights. One string decorate three fences numbered $1, 2, 5$ and the other string decorate three fences numbered $1, 2, 3, 4$. If he switch off all strings, then cows can get the first shape; if he switch on all strings, then cows can get the last shape; if he switch on only the first string, then then cows can get the third shape; if he switch on only the second string, then then cows can get the second shape. Thus, he can spend 106 dollars to meet need of cows, and it also turns out to be the minimum cost.

Input

The first line contains an integer t ($1 \le t \le 8$), meaning the total number of test cases. For each test case, the first line of input contains n ($1 \le n \le 1000$). The following n lines describe the fences and each line will contain five space separated integers x_{1i}, y_{1i}, x_{2i}, y_{2i} and p_i, where $|x_{1i}|, |y_{1i}|, |x_{2i}|, |y_{2i}| \le 10^9$ and $1 \le p_i \le 10^5$.

Output

For each test case, output one integer, which is the minimum money John should spend.

Sample Input

```
2
5
0 0 0 1 1
0 0 1 0 1
0 1 1 1 1
1 0 1 1 1
1 0 0 1 100
9
1 1 3 1 1
1 1 1 3 2
3 1 3 3 2
1 3 3 3 1
1 1 2 2 2
2 2 3 3 3
3 1 2 2 1
2 2 1 3 2
4 1 5 1 4
```

Sample Output

```
Case #1: 106
Case #2: 22
```

On the last day before the famous mathematician Swan's death, he left a problem to the world: Given integers n and a_i for $0 \leq i \leq 4$, calculate the number of n-digit integers which have at most a_i-digit i in its decimal representation (and have no $5, 6, 7, 8$ or 9). Leading zeros are not allowed in this problem.

Input

There is one integer T $(1 < T \leq 10)$ in the beginning of input, which means that you need to process T test cases. In each test case, there is one line containing six integers representing n and a_0 to a_4, where $2 \leq n \leq 15000$ and $0 \leq a_i \leq 30000$.

Output

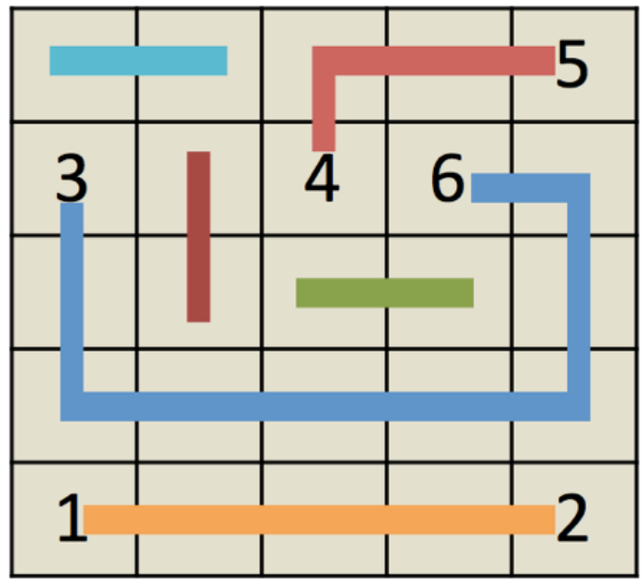
For each test case, you should print first the identifier of the test case and then the answer to the problem, module $10^9 + 7$.

Sample Input

```
10
5 0 1 2 3 4
5 1 1 1 1 1
5 2 2 2 2 2
5 3 3 3 3 3
5 3 2 1 3 2
5 3 2 0 0 0
5 0 0 0 5 0
7000 41 2467 6334 2500 3169
7000 7724 3478 5358 2962 464
7000 5705 4145 7281 827 1961
```

Sample Output

```
Case #1: 535
Case #2: 96
Case #3: 1776
Case #4: 2416
Case #5: 1460
Case #6: 4
Case #7: 1
Case #8: 459640029
Case #9: 791187801
Case #10: 526649529
```



Number Link is a famous game available in platforms including iOS and Android. Given a board with n rows and m columns, the target of the game is to connect pairs of grids with the same numbers. Once two numbers are paired, the path connecting them will occupy the corresponding grids. The path can only go vertically or horizontally. Note that, no two paths could intersect (by sharing the same grid) in any grid. In this problem, you are going to play a modified version, called Number Link ++. See the picture on the right for an example.

In this new game, you can use two types of paths. Type I is to connect two number grids with different parities (i.e., connect odd number with any other even number). It might be hard to cover the entire grid with only type I path, so we allow type II path, which is a circle path covers only the empty grids (the only **special case of type II path** is a path only connecting two adjacent empty grids; see the figure above). Since there is no free lunch, we have no free path either. When goes from grid (a, b) to an adjacent grid (c, d), you have to pay for a certain amount of tolls. The cost is the same when goes back from (c, d) to (a, b). Usually the cost of a path is the sum of tolls you paid by traveling along the grids on this path. The only exception is for the special case of type II path. In that case, you have to **pay twice** the cost (since it is a circle). The total cost of the game is the sum of costs for all the paths.

Can you help me figure out the paths so that each grid is on exactly one path? If there exists such solution, what is the minimum possible cost?

Input

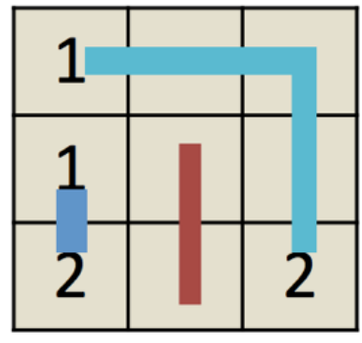
The first line of input consists of an integer T, which is the number of test cases. Each case begins with two integers, n and m, in a line ($1 \leq n, m \leq 50$). The next n lines describe the board. Each line consists of m nonnegative numbers, which describe the status of each column from left to right. If the number is zero, then the grid is empty; otherwise it indicates the number on the corresponding grid. The next $n-1$ lines each have m nonnegative numbers, which describe the cost of vertical connection. The j-th number in i-th line is the cost when travels from grid (i, j) to $(i+1, j)$. The next n lines each have $m-1$ nonnegative numbers, which describe the cost of horizontal connection. The j-th number in i-th line is the cost for a path to go from grid (i, j) to $(i, j+1)$. All the numbers, including the answer, can be represented using 32-bit signed integer.

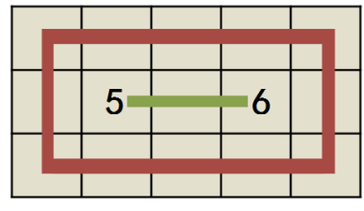
Output

For each test case, first output the case number, then output a single number, which is the minimum cost possible to finish the game. When there is no solution available, simply output '`-1`'.

Hint:

On the right are the solutions corresponding to case 1 and case 3 respectively. In case 1, you should double pay the red path, since it is a special case of type II path.





Sample Input

```
3
3 3
1 0 0
1 0 0
2 0 2
1 2 1
2 1 1
3 1
5 6
1 4
1 4
1 1 2 2
1 2 3
3 5
0 0 0 0 0
0 5 0 6 0
0 0 0 0 0
1 1000 1000 1000 1
1 1000 1000 1000 1
1 1 1 1
1000 1 1 1000
1 1 1 1
```

Sample Output

```
Case #1: 10
Case #2: -1
Case #3: 14
```

Bessie and her friend Elsie decide to have a meeting. However, after Farmer John decorated his fences they were separated into different blocks. John's farm are divided into n blocks labelled from 1 to n. Bessie lives in the first block while Elsie lives in the n-th one. They have a map of the farm which shows that it takes they t_i minutes to travel from a block in E_i to another block in E_i where E_i $(1 \leq i \leq m)$ is a set of blocks. They want to know how soon they can meet each other and which block should be chosen to have the meeting.

Input

The first line contains an integer T $(1 \leq T \leq 6)$, the number of test cases. Then T test cases follow.

The first line of input contains n and m. $2 \leq n \leq 10^5$. The following m lines describe the sets E_i $(1 \leq i \leq m)$. Each line will contain two integers $t_i (1 \leq t_i \leq 10^9)$ and S_i $(S_i > 0)$ firstly. Then S_i integer follows which are the labels of blocks in E_i. It is guaranteed that $\sum_{i=1}^{m} S_i \leq 10^6$.

Output

For each test case, if they cannot have the meeting, then output '`Evil John`' (without quotes) in one line.

Otherwise, output two lines. The first line contains an integer, the time it takes for they to meet. The second line contains the numbers of blocks where they meet. If there are multiple optional blocks, output all of them in ascending order.

Hint:

In the first case, it will take Bessie 1 minute travelling to the 3rd block, and it will take Elsie 3 minutes travelling to the 3rd block. It will take Bessie 3 minutes travelling to the 4th block, and it will take Elsie 3 minutes travelling to the 4th block. In the second case, it is impossible for them to meet.

Sample Input

```
2
5 4
1 3 1 2 3
2 2 3 4
10 2 1 5
3 3 3 4 5
3 1
1 2 1 2
```

Sample Output

```
Case #1: 3
3 4
Case #2: Evil John
```