Code Jam 2021 - Round 3

Build-A-Pair

Problem

You want to build a pair of positive integers. To do that, you are given a list of decimal digits to use. You must use every digit in the list exactly once, but you get to choose which ones to use for the first integer and which ones to use for the second integer. You also get to choose the order of the digits within each integer, except you cannot put a zero as the most significant (leftmost) digit in either integer. Note that you cannot choose just a zero for one integer either, because it would not be positive.

For example, you could be given the list $[1, 0, 2, 0, 4, 3]$. Two of the valid pairs you can build are $(200, 143)$ and $(3, 12400)$. The following pairs, on the other hand, are not valid:

- $(0102, 34)$: has a leading zero.
- $(0, 12340)$: has a non-positive integer.
- $(10, 243)$ and $(12300, 47)$: the list of digits in each of these pairs is not exactly equal to the given list of digits.

Given the list of digits to use, what is the minimum absolute difference between the two built integers that can be achieved?

Input

The first line of the input gives the number of test cases, $\mathbf{T}$. $\mathbf{T}$ lines follow. Each line describes a test case with a single string of digits $\mathbf{D}$. Each character of $\mathbf{D}$ is a digit you must use.

Output

For each test case, output one line containing `Case #`x`:` y, where x is the test case number (starting from 1) and y is the minimum possible absolute difference between the two integers built from $\mathbf{D}$ according to the rules above.

Limits

Time limit: 5 seconds.
Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 100$.
Each character of $\mathbf{D}$ is a decimal digit.
At least two characters of $\mathbf{D}$ are not `0`.

Test Set 1 (Visible Verdict)

$2 \leq$ the length of $\mathbf{D} \leq 8$.

Test Set 2 (Visible Verdict)

$2 \leq$ the length of $\mathbf{D} \leq 36$.

Sample

Sample Input

```
4
1234
0011
07080
0899
```

Sample Output

```
Case #1: 7
Case #2: 0
Case #3: 620
Case #4: 1
```

The optimal pair of integers to build are 31 and 24 for Sample Case #1, 10 and 10 for Sample Case #2, 700 and 80 for Sample Case #3, and 89 and 90 for Sample Case #4.

Code Jam 2021 - Round 3

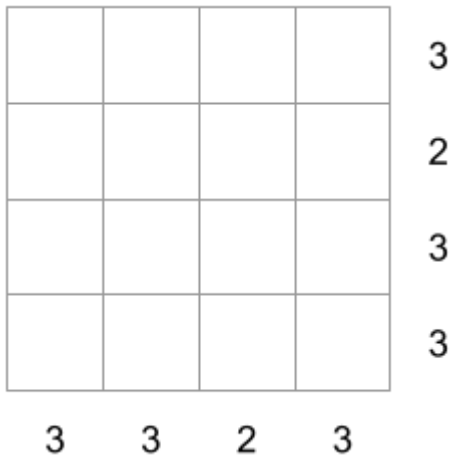
Square Free

Problem

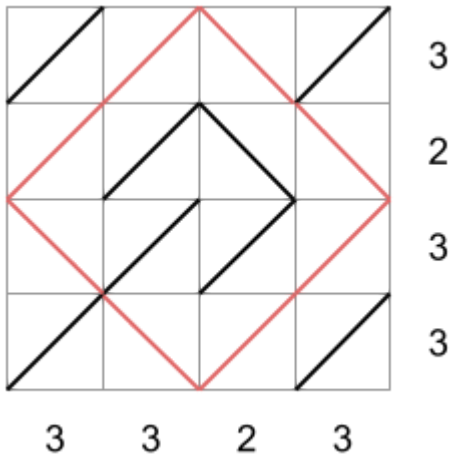
We have a matrix of square cells with $\mathbf{R}$ rows and $\mathbf{C}$ columns. We need to draw a diagonal in each cell. Exactly one of two possible diagonals must be drawn in each cell: the *forward slash* diagonal, which connects the bottom-left and the top-right corners of the cell, or the *backslash* diagonal, which connects the top-left and the bottom-right corners of the cell.

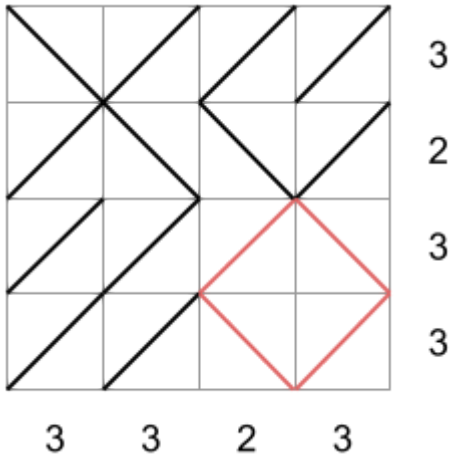
For each row and column, we want to draw a specific number of diagonals of each type. Also, after all the diagonals are drawn, the matrix should be *square free*. That is, there should be no squares formed using the diagonals we added.

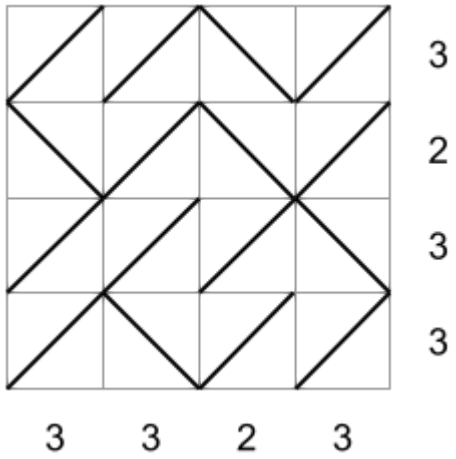
For example, suppose we have a matrix with 4 rows and 4 columns. The number next to each row is the exact number of forward slash diagonals there must be in that row. The number below each column is the exact number of forward slash diagonals there must be in that column.



There are multiple ways to fill the matrix respecting those per-row and per-column amounts. Below we depict three possibilities:







The first two matrices are not square free, while the third matrix is. In the first matrix, there is a square of side-length 2 diagonals with its vertices in the middle of each side of the matrix. In the second matrix, there is a square of side-length 1 diagonal drawn in the bottom-right corner. In the third matrix, there is no square. The third matrix would then be a valid drawing according to all the rules.

Given the size of the matrix and the exact number of forward slash diagonals that must be drawn in each row and column, produce any square free matrix that satisfies the row and column constraints, or say that one does not exist.

Input

The first line of the input gives the number of test cases, $\mathbf{T}$. $\mathbf{T}$ test cases follow. Each test case consists of exactly three lines. The first line of a test case contains $\mathbf{R}$ and $\mathbf{C}$, the number of rows and columns of the matrix. The second line of a test case contains $\mathbf{R}$ integers $\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_R}$. $\mathbf{S_i}$ is the exact number of forward slash diagonals that must be drawn in the i-th row from the top. The third line of a test case contains $\mathbf{C}$ integers $\mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_C}$. $\mathbf{D_i}$ is the exact number of forward slash diagonals that must be drawn in the i-th column from the left.

Output

For each test case, output one line containing `Case #`x`: `y, where x is the test case number (starting from 1) and y is `IMPOSSIBLE` if there is no filled matrix that follows all rules and `POSSIBLE` otherwise. If you output `POSSIBLE`, output $\mathbf{R}$ more lines with $\mathbf{C}$ characters each. The j-th character of the i-th of these lines must be `/` if the diagonal drawn in the i-th row from the top and j-th column from the left in your proposed matrix is a forward slash diagonal, and `\` otherwise. Your proposed matrix must be valid according to all rules.

Limits

Time limit: 15 seconds.
Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 100$.
$0 \leq \mathbf{S_i} \leq \mathbf{C}$, for all i.
$0 \leq \mathbf{D_i} \leq \mathbf{R}$, for all i.

Test Set 1 (Visible Verdict)

$2 \leq \mathbf{R} \leq 6$.
$2 \leq \mathbf{C} \leq 6$.

Test Set 2 (Hidden Verdict)

$2 \leq \mathbf{R} \leq 20$.
$2 \leq \mathbf{C} \leq 20$.

Sample

Sample Input

```
4
4 4
3 2 3 3
3 3 2 3
2 3
1 1
1 1 1
2 3
1 2
1 1 1
3 3
2 0 2
2 0 2
```

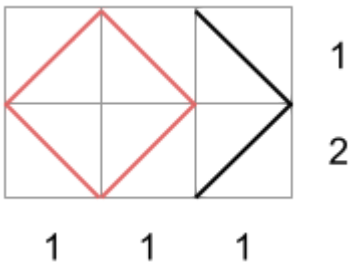
Sample Output

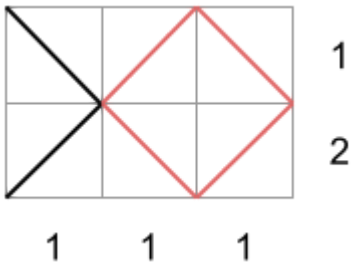
```
Case #1: POSSIBLE
//\/
\/\/
///\
/\//
Case #2: IMPOSSIBLE
Case #3: POSSIBLE
\\/
//\
Case #4: POSSIBLE
/\/
\\\
/\/
```

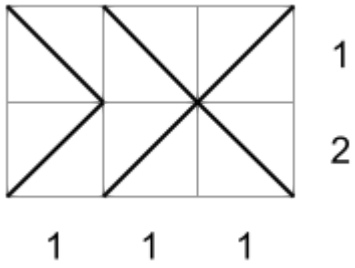
Sample Case #1 is the one explained above.

In Sample Case #2, there must be a total of 2 forward slash diagonals according to the sum of the row totals, but a total of 3 according to the sum of the column totals. It is therefore impossible to follow all rules.

In Sample Case #3 the only matrices that follow the row and column totals are the following:

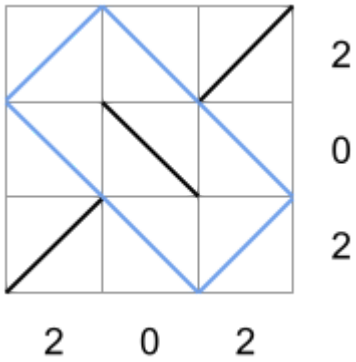






Since the first two contain a square, the third one is the only valid output for this case.

In Sample Case #4 there is only one way to fill the matrix that follows the row and column totals, shown in the picture below. Note that it produces a single rectangle, shown in blue in the picture. But, since that rectangle is not a square, the matrix is square free.



Code Jam 2021 - Round 3

Fence Design

Problem

You are hired as a temporary employee of the Fence Construction Company and have been tasked with finishing the design of the fencing for a field. Each fence must run in a straight line between two poles. Each pole occupies a single point and the location of each pole is fixed. No three poles are collinear. Fences cannot intersect each other, except possibly at their endpoints (the poles).

The design was started by someone else, but they quit the project after adding exactly two fences. You need to finish their design. To impress your bosses and clients, you want the design to have as many fences as possible, regardless of their lengths.

Given the positions of the poles and the already-built fences, please find a way to add as many fences as possible such that no pair of fences (new or existing) intersect each other, except possibly at their endpoints (the poles).

Input

The first line of the input gives the number of test cases, $\mathbf{T}$. $\mathbf{T}$ test cases follow. Each test case starts with a single line containing an integer $\mathbf{N}$, indicating the number of poles. Then, $\mathbf{N}$ lines follow. The i-th of these lines contains two integers $\mathbf{X_i}$ and $\mathbf{Y_i}$, representing the X and Y coordinates of the i-th pole's position. The last two lines for each test case represent the two existing fences. These two lines contain two integers each: $\mathbf{P_k}$ and $\mathbf{Q_k}$, representing that the k-th existing fence runs between the $\mathbf{P_k}$-th and the $\mathbf{Q_k}$-th pole (poles are numbered starting from 1).

Output

For each test case, output one line containing `Case #`x`:` y, where x is the test case number (starting from 1) and y is the maximum number of fences that can be added to the design (not including the existing ones). Then, output y more lines. Each line must contain two distinct integers i and j (both between 1 and $\mathbf{N}$, inclusive), representing a different fence that connects the i-th and j-th poles. No pair of the $y + 2$ fences (the existing fences as well as the ones you have added) may overlap, except possibly at their endpoints.

Limits

Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 50$.
$-10^9 \leq \mathbf{X_i} \leq 10^9$, for all i.
$-10^9 \leq \mathbf{Y_i} \leq 10^9$, for all i.
$(\mathbf{X_i}, \mathbf{Y_i}) \neq (\mathbf{X_j}, \mathbf{Y_j})$, for all $i \neq j$.
$1 \leq \mathbf{P_k} < \mathbf{Q_k} \leq \mathbf{N}$, for all k.
The existing fences do not intersect, except possibly at their endpoints.
No three poles are collinear.

Test Set 1 (Visible Verdict)

Time limit: 60 seconds.
$4 \leq \mathbf{N} \leq 100$.

Test Set 2 (Hidden Verdict)

Time limit: 90 seconds.
$4 \leq \mathbf{N} \leq 10^5$.

Sample

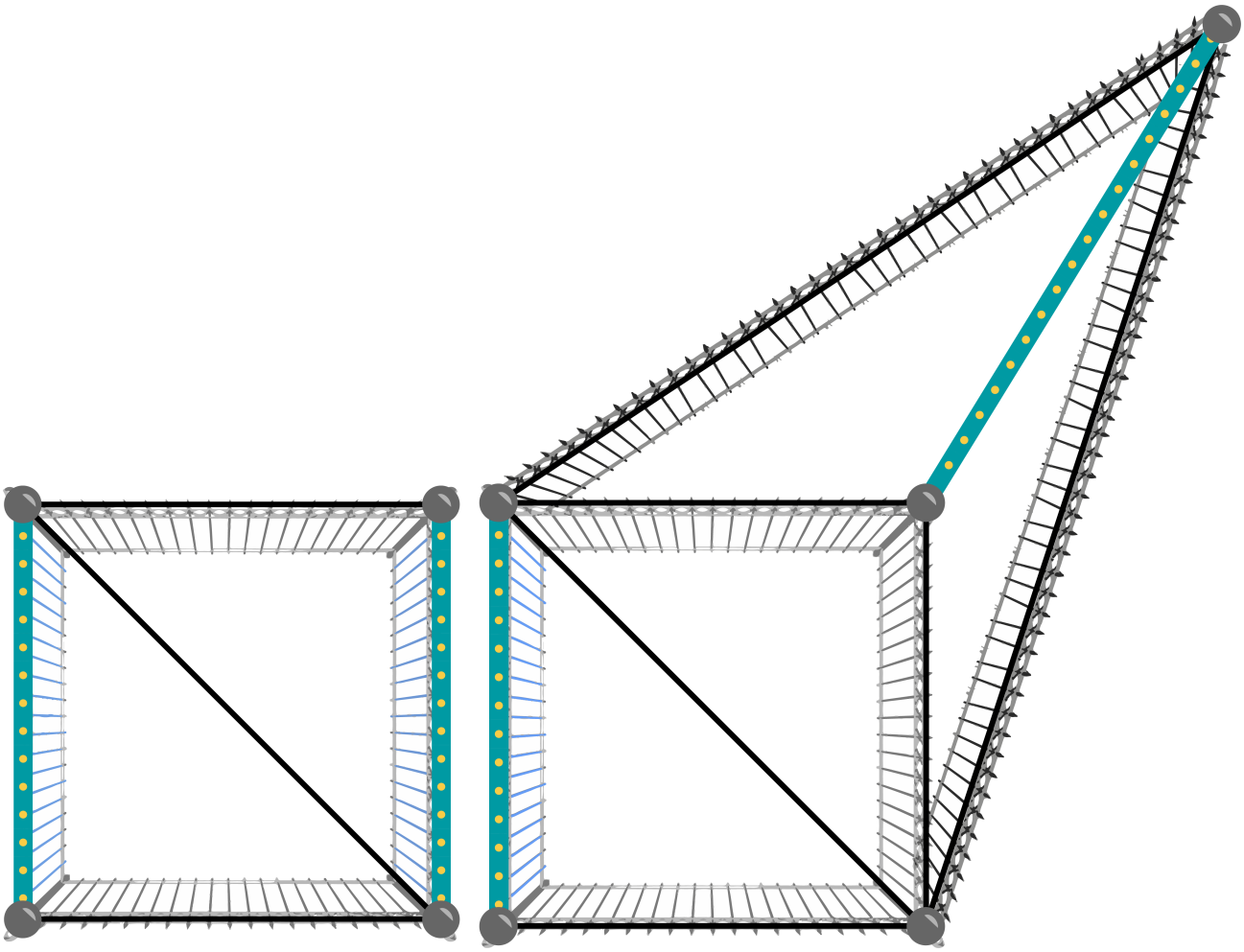
Sample Input

```
2
4
0 0
0 1
1 1
1 0
1 2
3 4
5
0 0
0 1
1 1
1 0
2 3
1 2
3 5
```

Sample Output

```
Case #1: 3
1 4
2 3
4 2
Case #2: 6
5 4
2 4
5 2
1 4
4 3
3 2
```

The following pictures show the poles and fences in the given samples. The fences with the wider blue line on them are the existing ones, and the rest show the way of adding a maximum number of fences shown in the sample output.



Code Jam 2021 - Round 3

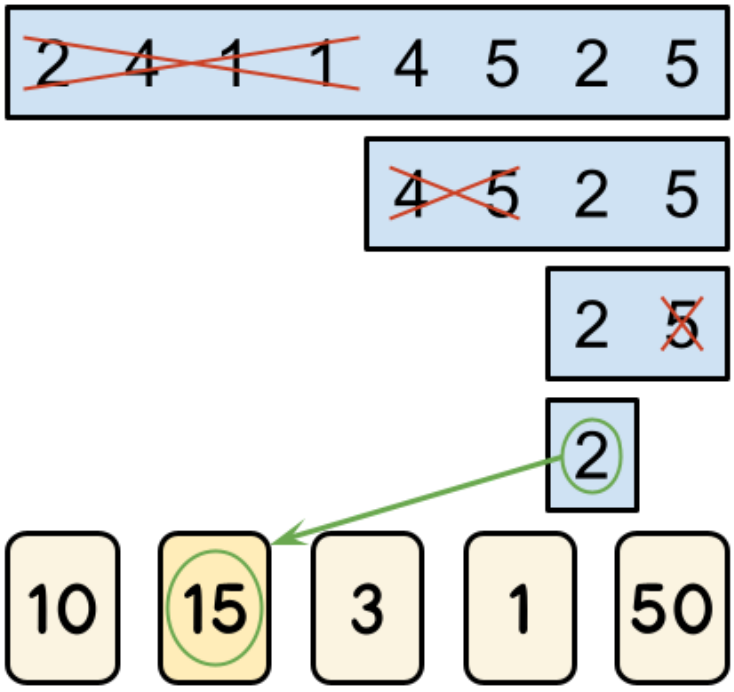
Binary Search Game

Problem

Alice and Bob are going to play the Binary Search game. The game is played on a board consisting of a single row of $2^{\mathbf{L}}$ cells. Each cell contains an integer between 1 and $\mathbf{N}$, inclusive. There are also $\mathbf{N}$ cards numbered 1 through $\mathbf{N}$. Before the game starts, the referee writes an integer between 1 and $\mathbf{M}$, inclusive, on each card, in one of the $\mathbf{M}^{\mathbf{N}}$ ways in which that can be done. Alice and Bob know the integers in the cells and on each card before they start playing.

The game proceeds alternating turns, with Alice having the first turn. There are $\mathbf{L}$ turns in total, which means Alice plays $\lceil \mathbf{L}/2 \rceil$ turns and Bob plays $\lfloor \mathbf{L}/2 \rfloor$ turns. During a turn, a player can eliminate either the leftmost half or the rightmost half of the remaining cells. For example, let us consider a board that contains the numbers $[2, 4, 1, 1, 4, 5, 2, 5]$. In her first turn, Alice must choose to eliminate one half, leaving either $[2, 4, 1, 1]$ or $[4, 5, 2, 5]$. If she eliminates the leftmost half and leaves $[4, 5, 2, 5]$, then Bob must choose between leaving $[4, 5]$ and $[2, 5]$. If he were to leave $[2, 5]$, the game's final turn would have Alice choosing between $[2]$ and $[5]$.

When the game is over, they look at the number X in the only remaining cell. The *score* of the game is the integer written on card number X. In the example above, if Alice were to eliminate $[5]$ and leave $[2]$ in her final turn, the score of the game would be the number the referee wrote on card number 2.



Alice plays optimally to maximize the score of the game, while Bob plays optimally to minimize it. They are given a fixed board with integers $\mathbf{A_1}, \mathbf{A_2}, \ldots \mathbf{A_{2^L}}$ in its cells. For maximal fairness, they will play $\mathbf{M}^{\mathbf{N}}$ games, and the referee will choose a different way to write integers on the cards for each one. That means that for any given way of writing integers on the cards, Alice and Bob will play exactly one game with it. Given the game parameters and the fixed board contents, please determine the sum of the scores of all those games. Since the output can be a really big number, we only ask you to output the remainder of dividing the result by the prime $10^9 + 7$ (1000000007).

Input

The first line of the input gives the number of test cases, $\mathbf{T}$. $\mathbf{T}$ test cases follow. Each test case consists of exactly two lines. The first line of each test case contains the three integers $\mathbf{N}$, $\mathbf{M}$, and $\mathbf{L}$. The second line contains $2^{\mathbf{L}}$ integers $\mathbf{A_1}, \mathbf{A_2}, \ldots, \mathbf{A_{2^L}}$, where $\mathbf{A_i}$ is the integer contained in the i-th cell from the left of the board.

Output

For each test case, output one line containing `Case #`x`:` y, where x is the test case number (starting from 1) and y is the sum of scores of all $\mathbf{M^N}$ games, modulo the prime $10^9 + 7$ (1000000007).

Limits

Time limit: 30 seconds.
Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 12$.
$1 \leq \mathbf{L} \leq 5$.
$1 \leq \mathbf{A_i} \leq \mathbf{N}$, for all i.

Test Set 1 (Visible Verdict)

$1 \leq \mathbf{N} \leq 8$.
$1 \leq \mathbf{M} \leq 100$.

Test Set 2 (Hidden Verdict)

$1 \leq \mathbf{N} \leq 32$.
$1 \leq \mathbf{M} \leq 10^9$.

Sample

Sample Input

```
3
2 2 2
2 1 1 1
4 3 2
3 1 1 4
5 100 3
2 4 1 1 4 5 2 5
```

Sample Output

```
Case #1: 6
Case #2: 144
Case #3: 991661422
```

In Sample Case #1, there are 4 ways to write the integers on the blank cards: $[1, 1]$, $[1, 2]$, $[2, 1]$, and $[2, 2]$. In the first two ways, no matter what Alice chooses in her first turn, Bob can always make the number in the last remaining cell be a 1, and card 1 contains a 1, which means those two games have a score of 1. In the last two ways, Alice can start by eliminating the leftmost half of the board, leaving $[1, 1]$ for Bob, who then has no choice but to leave $[1]$ at the end. Since card 1 has a 2 on it in these ways, the score of both of these games is 2. The sum of all scores is therefore $1 + 1 + 2 + 2 = 6$.