上海大学
SHANGHAI UNIVERSITY

2019 ICPC

The 44th ICPC Asia Regional Programming Contest Shanghai Site

第44届ICPC国际大学生程序设计竞赛亚洲区域赛（上海）

2019年11月23-24日

现场赛试题册

This page is intentionally left blank.

Problem A. Mr. Panda and Dominoes

Input file: `standard input`
Output file: `standard output`

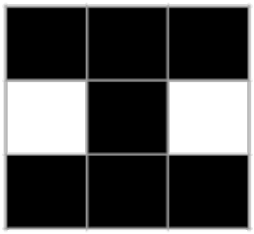
Mr. Panda likes creating and solving mathematical puzzles. One day, Mr. Panda came up with a puzzle when he was playing dominoes.

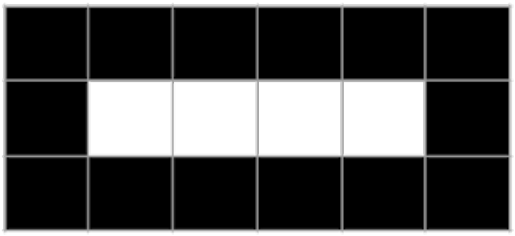
In a grid that consists of an infinite number of rows and columns, all cells are white except **N** given cells which are colored black. Mr. Panda wants to know how many dominoes there are in the gird.

A domino is a rectangle that meets the following requirements.

- The domino is formed of a continuous subset of the rows and columns of the grid.
- The domino has at least 1 row and 1 column.
- The cells on the edge of the domino are black. It means the topmost row, the bottommost row, the leftmost column and the rightmost column only consist of black cells.
- The aspect ratio of the domino is $2:1$ or $1:2$. It means, if the domino has k columns, it should have either $2k$ rows or $\frac{k}{2}$ rows (k is an even number in this case).

For example, in the chart below, the 3×3 grid on the left contains 6 dominos (4 dominos of 1×2 and 2 dominos of 2×1), and the 3×6 grid on the right contains 15 dominos (10 dominos of 1×2, 4 dominos of 2×1 and a domino of 3×6).





Because the grid is huge, Mr. Panda is too lazy to count the number of dominoes. Could you please help Mr. Panda find how many dominoes there are in the gird?

Input

The first line of input gives the number of test cases, **T**. **T** test cases follow.

Each test case starts with a line consists of an integer - **N**, the number of black cells in the gird.

Then, **N** lines follow. Each line consists of 2 integers $\mathbf{R_i}$ and $\mathbf{C_i}$, indicating row and column of a black cell in the grid.

$1 \leq \mathbf{T} \leq 100$

$1 \leq \mathbf{N} \leq 1,000,000$

$1 \leq \mathbf{R_i}, \mathbf{C_i} \leq 1,000,000,000$

$\sum \mathbf{N} \leq 5,000,000$ over all test cases

For each test, no two black cells are in the same position

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 number of dominos that Mr. Panda wants to know for the i-th input data set.

Sample input and output

Sample Input	Sample Output
2 7 1 1 1 2 1 3 2 2 3 1 3 2 3 3 14 1 1 1 2 1 3 1 4 1 5 1 6 2 1 2 6 3 1 3 2 3 3 3 4 3 5 3 6	Case #1: 6 Case #2: 15

Problem B. Prefix Code

Input file: `standard input`
Output file: `standard output`

A prefix code is a type of code system distinguished by its possession of the "prefix property", which requires that there is no whole code word in the system that is a prefix (initial segment) of any other code word in the system. It is trivially true for fixed-length code, so only a point of consideration in variable-length code.

For example, a code with code words {9, 55} has the prefix property; a code consisting of {9, 5, 59, 55} does not, because "5" is a prefix of "59" and also of "55". A prefix code is a uniquely decodable code: given a complete and accurate sequence, a receiver can identify each word without requiring a special marker between words. However, there are uniquely decodable codes that are not prefix codes; for instance, the reverse of a prefix code is still uniquely decodable (it is a suffix code), but it is not necessarily a prefix code.

Prefix codes are also known as prefix-free codes , prefix condition codes and instantaneous codes. Although Huffman coding is just one of many algorithms for deriving prefix codes, prefix codes are also widely referred to as "Huffman codes", even when the code was not produced by a Huffman algorithm. The term comma-free code is sometimes also applied as a synonym for prefix-free codes but in most mathematical books and articles a comma-free code is used to mean a self-synchronizing code, a subclass of prefix codes.

Using prefix codes, a message can be transmitted as a sequence of concatenated code words, without any out-of-band markers or (alternatively) special markers between words to frame the words in the message. The recipient can decode the message unambiguously, by repeatedly finding and removing sequences that form valid code words. This is not generally possible with codes that lack the prefix property, for example {0, 1, 10, 11}: a receiver reading a "1" at the start of a code word would not know whether that was the complete code word "1", or merely the prefix of the code word "10" or "11"; so the string "10" could be interpreted either as a single codeword or as the concatenation of the words "1" then "0".

The variable-length Huffman codes, country calling codes, the country and publisher parts of ISBNs, the Secondary Synchronization Codes used in the UMTS W-CDMA 3G Wireless Standard, and the instruction sets (machine language) of most computer microarchitectures are prefix codes.

Prefix codes are not error-correcting codes. In practice, a message might first be compressed with a prefix code, and then encoded again with channel coding (including error correction) before transmission.

For any uniquely decodable code there is a prefix code that has the same code word lengths. Kraft's inequality characterizes the sets of code word lengths that are possible in a uniquely decodable code.

In this problem, you can give a code with $\mathbf{N}$ code words. Each word contains only numbers {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. You need to check whether the code can meet the prefix property: there is no whole code word in the system that is a prefix of any other code word.

Input

The first line of the input gives the numbers of test cases, $\mathbf{T}$ ($1 \leq \mathbf{T} \leq 100$). $\mathbf{T}$ test cases follow.

Each test case consists of one line with one integer $\mathbf{N}$ ($1 \leq \mathbf{N} \leq 10,000$), the number of code words.

Then follows **N** lines with one code word on each line. A code word is a sequence of at most ten digits.

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 "Yes" if the code is a prefix code, or "No" if not.

Sample input and output

Sample Input	Sample Output
3 2 9 55 4 9 5 59 55 4 01 01 123 321	Case #1: Yes Case #2: No Case #3: No

Problem C. Maze

Input file: standard input
Output file: standard output

As the title implies, your task in this problem is related to maze, specifically a 2D maze of $\mathbf{N}$ rows and $\mathbf{M}$ columns ($\mathbf{N} \leq \mathbf{M}$), where rows are numbered from 1 to $\mathbf{N}$ from top to bottom, and columns are numbered from 1 to $\mathbf{M}$ from left to right. The cell at the i-th row and the j-th column is denoted by (i, j).

The maze has only one entry which is at $(1, 1)$ and only one exit which is at $(\mathbf{N}, \mathbf{M})$. To simplify things, you are only allowed to move right or down at any step. There might be cells with obstacle which cannot be visited, and there might be cells with treasure which must be visited.

In this problem, you are requested to count the number of valid paths from the starting location to the ending location.

There are $\mathbf{S}$ cells with treasure. The valid path should visit all cells with treasure.

The following cells with obstacle, the valid path cannot visit any cell with obstacle.

- all cells (x, y) meet $x > y$,
- all cells (x, y) meet $\mathbf{M} + x < \mathbf{N} + y$,
- and there are $\mathbf{K}$ additional cells with obstacle.

Input

The first line of the input gives the number of test case, $\mathbf{T}$ ($1 \leq \mathbf{T} \leq 10$). $\mathbf{T}$ test cases follow.

Each test case consists of one line with four integers $\mathbf{N}$, $\mathbf{M}$, $\mathbf{S}$, and $\mathbf{K}$ as described above. ($1 \leq \mathbf{N} \leq \mathbf{M} \leq 100000$, $0 \leq \mathbf{S} \leq 10$, $0 \leq \mathbf{K} \leq 20$)

Then, $\mathbf{S}$ lines follow. Each line consists of 2 integers x and y, indicating the cells with treasure in the maze. ($1 \leq x \leq \mathbf{N}$, $1 \leq y \leq \mathbf{M}$, $x \leq y$, $\mathbf{M} + x \geq \mathbf{N} + y$) (no two cells are in the same position)

After that, there are $\mathbf{K}$ lines. Each line consists of 2 integers x and y, indicating the cells with obstacle in the maze. ($1 \leq x \leq \mathbf{N}$, $1 \leq y \leq \mathbf{M}$, $x \leq y$, $\mathbf{M} + x \geq \mathbf{N} + y$) (no two cells are in the same position)

Output

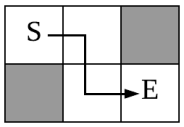
For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the number of valid paths module 1,000,000,007 ($10^9 + 7$).

Sample input and output

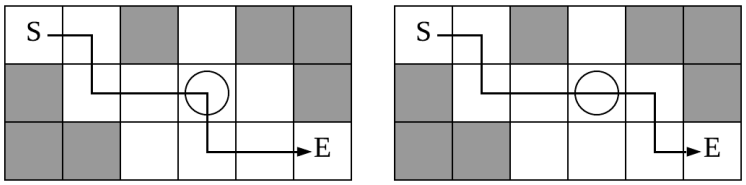
Sample Input	Sample Output
2 2 3 0 0 3 6 1 1 2 4 1 3	Case #1: 1 Case #2: 2

Note

In Sample Case #1, one valid path are:



In Sample Case #2, two valid paths are:



Problem D. Spanning Tree Removal

Input file: standard input
Output file: standard output

Bob has recently learned algorithms on finding spanning trees and wanted to give you a quiz.

To recap, a spanning tree is a subset of graph G, which has all the vertices covered with minimum possible number of edges. Hence, a spanning tree does not have cycles and it cannot be disconnected. Given an arbitrary undirected simple graph (without multiple edges and loops), a spanning tree removal is defined as: retrieve one spanning tree of the graph, and remove all the edges selected by the spanning tree from the original graph, obtaining a new graph.

Bob found "spanning tree removal" so fun that he wanted to do it over and over again. In particular, he began with a complete graph, i.e., a graph with every pair of distinct vertices connected by a unique edge. Your goal, is to smartly choose spanning trees to remove so that you can repeat as many as times as possible until there is no longer a spanning tree in the graph.

Input

The input file starts with an integer $\mathbf{T}$ ($1 \leq \mathbf{T} \leq 500$), denoting the number of test cases.

Each test case is one line: $\mathbf{N}$ ($2 \leq \mathbf{N} \leq 1000$), which the number of vertices of the graph to begin with.

The sum of $\mathbf{N}$ over all test cases in a single input does not exceed 1000.

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 how many times at most you can do the removal.

Then follows $y \times (\mathbf{N} - 1)$ lines. From line $(\mathbf{N} - 1) \times (i - 1) + 1$ to line $(\mathbf{N} - 1) \times i$, you should print a spanning tree you decided to remove at i-th time, in the format that everyone should be familiar with. Namely, each line contains two numbers u and v ($1 \leq u, v \leq \mathbf{N}$, $u \neq v$). (u, v) should be valid tree edge, and does not coincide with edges that have been removed before. If there are several solutions, output any of them.

Sample input and output

Sample Input	Sample Output
3 2 3 4	Case #1: 1 1 2 Case #2: 1 3 1 1 2 Case #3: 2 1 3 3 4 2 4 3 2 1 4 2 1

Problem E. Cave Escape

Input file: standard input
Output file: standard output

Bob is stuck in a cave represented by a matrix of $\mathbf{N}$ rows and $\mathbf{M}$ columns, where rows are numbered from $\mathbf{1}$ to $\mathbf{N}$ from top to buttom, and columns are numbered from $\mathbf{1}$ to $\mathbf{M}$ from left to right. The cell at the i-th row and the j-th column is denoted by (i, j).

Bob is currently at the cell $(\mathbf{S_R}, \mathbf{S_C})$, and the exit of the save is located at the cell $(\mathbf{T_R}, \mathbf{T_C})$.

Each cell in this cave contains a number, which is called the magic value. The magic value of cell (i, j) is $\mathbf{V_{ij}}$.

When Bob moves from one cell to an unvisited cell, he gains energy points equals to the product of two magic values. It means, if Bob moves from cell (i, j) to cell (x, y), and cell (x, y) is unvisited, he will gain $\mathbf{V_{ij}} \times \mathbf{V_{xy}}$ energy points.

Bob can move between cells that share an edge (not just a corner). On the exit cell, Bob can choose not to exit the cave and continue to explore the cave if he want to. Can you help him find the maximum number of energy points he can gain when he exits the cave.

Input

The first line of the input gives the numbers of test cases, $\mathbf{T}$. $\mathbf{T}$ test cases follow. Each test case contains two lines. The first line contains six integers $\mathbf{N}$, $\mathbf{M}$, $\mathbf{S_R}$, $\mathbf{S_C}$, $\mathbf{T_R}$ and $\mathbf{T_C}$ as described above.

The next line contains fix integers in the following format, respectively:

$\mathbf{X_1}$ $\mathbf{X_2}$ $\mathbf{A}$ $\mathbf{B}$ $\mathbf{C}$ $\mathbf{P}$

These values are used to generate $\mathbf{V_{ij}}$ as follows:

We define:

$\mathbf{X_i} = (\mathbf{A} \times \mathbf{X_{i-1}} + \mathbf{B} \times \mathbf{X_{i-2}} + \mathbf{C})$ *module* $\mathbf{P}$, for $i = 3$ to $\mathbf{N} \times \mathbf{M}$.

We also define:

$\mathbf{V_{ij}} = \mathbf{X_{(i-1)*M+j}}$, for $i = 1$ to $\mathbf{N}$, and $j = 1$ to $\mathbf{M}$.

$1 \leq \mathbf{T} \leq 10$.

$1 \leq \mathbf{N}, \mathbf{M} \leq 1000$.

$1 \leq \mathbf{S_R}, \mathbf{T_R} \leq \mathbf{N}$.

$1 \leq \mathbf{S_C}, \mathbf{T_C} \leq \mathbf{M}$.

$0 \leq \mathbf{X_1}, \mathbf{X_2}, \mathbf{A}, \mathbf{B}, \mathbf{C} \leq \mathbf{P}$.

$1 \leq \mathbf{P} \leq 100$.

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 energy points that Bob can gain.

Sample input and output

Sample Input	Sample Output
2 1 4 1 2 1 3 1 2 2 3 1 5 6 1 2 1 4 1 0 1 2 3 0 4	Case #1: 17 Case #2: 8

Note

In Sample Case #1, The matrix is:

$$\begin{bmatrix} 1 & 2 & 3 & 3 \end{bmatrix}$$

one way to get 17 energy points is:

- $(1, 2) \to (1, 1)$, get 1×2 energy points.
- $(1, 1) \to (1, 2)$, get 0 energy points.
- $(1, 2) \to (1, 3)$, get 2×3 energy points.
- $(1, 3) \to (1, 4)$, get 3×3 energy points.
- $(1, 4) \to (1, 3)$, get 0 energy points.

In Sample Case #2, The matrix is:

$$\begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 0 \\ 1 \end{bmatrix}$$

one way to get 8 energy points is:

- $(2, 1) \to (3, 1)$, get 1×2 energy points.
- $(3, 1) \to (4, 1)$, get 2×3 energy points.

Problem F. A Simple Problem On A Tree

Input file: standard input
Output file: standard output

We have met so many problems on the tree, so today we will have a simple problem on a tree.

You are given a tree (an acyclic undirected connected graph) with $\mathbf{N}$ nodes. The tree nodes are numbered from $\mathbf{1}$ to $\mathbf{N}$. Each node has a weight $\mathbf{W_i}$. We will have four kinds of operations on it and you should solve them efficiently. Wish you have fun!

Input

The first line of the input gives the number of test case, $\mathbf{T}$ ($1 \leq \mathbf{T} \leq 10$). $\mathbf{T}$ test cases follow.

For each case, the first line contains only one integer $\mathbf{N}$.($1 \leq \mathbf{N} \leq 100,000$) The next $\mathbf{N-1}$ lines each contains two integers x, y which means there is an edge between them. It also means we will give you one tree initially.

The next line will contains $\mathbf{N}$ integers which means the initial weight $\mathbf{W_i}$ of each node. ($0 \leq \mathbf{W_i} \leq 1,000,000,000$)

The next line will contains an integer $\mathbf{Q}$. ($1 \leq \mathbf{Q} \leq 10,000$) The next $\mathbf{Q}$ lines will start with an integer 1, 2, 3 or 4 means the kind of this operation.

1. Given three integers u, v, w, for the u, v and all nodes between the path from u to v inclusive, you should update their weight to w. ($1 \leq u, v \leq \mathbf{N}$, $0 \leq w \leq 1,000,000,000$)
2. Given three integers u, v, w, for the u, v and all nodes between the path from u to v inclusive, you should increase their weight by w. ($1 \leq u, v \leq \mathbf{N}$, $0 \leq w \leq 1,000,000,000$)
3. Given three integers u, v, w, for the u, v and all nodes between the path from u to v inclusive, you should multiply their weight by w. ($1 \leq u, v \leq \mathbf{N}$, $0 \leq w \leq 1,000,000,000$)
4. Given two integers u, v, you should check the node weights on the path between u and v, and you should output cubic sum of them. It means, output $\sum_x \mathbf{W}_x^3$, x is node on the path from u to v (inclusive u and v). ($1 \leq u, v \leq \mathbf{N}$)

Output

For each test case, output one line containing "Case #x:", where x is the test case number (starting from 1). For operation 4, output a single integer in one line representing the result. The result could be huge, print it module 1,000,000,007($10^9 + 7$).

Sample input and output

Sample Input	Sample Output
1 5 2 1 1 3 5 3 4 3 1 2 3 4 5 6 4 2 4 1 5 4 2 2 2 4 3 3 2 3 4 4 5 4 4 2 4	Case #1: 100 8133 20221

Problem G. Play the game SET

Input file: standard input
Output file: standard output

SET is a real-time card game designed by Marsha Falco in 1974 and published by Set Enterprises in 1991. The deck consists of 81 unique cards that vary in four features across three possibilities for each kind of feature: number of shapes (one, two, or three), shape (diamond, squiggle, oval), shading (solid, striped, or open), and color (red, green, or purple). Each possible combination of features (e.g. a card with [three][striped][green][diamonds]) appears as a card precisely once in the deck.

In the game, certain combinations of three cards are said to make up a **set**. For each one of the four categories of features — color, number, shape, and shading — the three cards must display that feature as a) either all the same, or b) all different. Put another way: For each feature the three cards must avoid having two cards showing one version of the feature and the remaining card showing a different version.

For example,

[three][solid][red][diamonds]

[two][solid][green][squiggles]

[one][solid][purple][oval]

form a **set**, because the shadings of the three cards are all the same, while the numbers, the colors, and the shapes among the three cards are all different.

You're given **N** unique cards, you need to form them into sets. You need to output the maximum number of sets with the given cards, and each card is used at most once.

Input

The first line of the input gives the numbers of test cases, **T**. **T** test cases follow. Each test case consists of one line with one integer **N** as described above. The i-th of the next **N** lines describes the i-th card. It contains four categories features, in the format of "[*][*][*][*]"

There are no identical cards.

$1 \leq \mathbf{T} \leq 100$.

$1 \leq \mathbf{N} \leq 38$.

There are at most 10 test cases have $\mathbf{N} > 35$.

Output

For each test case, the first line print "Case #x: y", where x is the test case number (starting from 1) and y is the maximum number of sets. Then print y triples $id1$, $id2$, $id3$, defining the indexes of cards (starting from 1) that can make up a **set**. If there are several solutions, output any of them.

Sample input and output

Sample Input	Sample Output
1 7 [one][diamond][solid][red] [two][diamond][solid][red] [three][diamond][solid][red] [one][diamond][solid][green] [one][diamond][solid][purple] [two][diamond][solid][purple] [purple][three][diamond][solid]	Case #1: 2 1 2 3 5 6 7

Problem H. Tree Partition

Input file: standard input
Output file: standard output

Bob has learned about all fancy algorithms on trees recently, so he decided to give you a little quiz.

You are given an undirected tree with $\mathbf{N}$ weighted nodes. You are to cut the tree into $\mathbf{K}$ partitions (subtrees). In other words, you are to cut $\mathbf{K}-\mathbf{1}$ edges from the tree so that the rest form a forest consisting of $\mathbf{K}$ trees.

The weight of a tree is defined as the sum of weights of all nodes in the tree. Your final score is the maximum weight of all partitions. Please find the best partition so that your final score is as small as possible.

Input

The first line is a positive integer $\mathbf{T}$ ($1 \leq \mathbf{T} \leq 10^5$), which is the number of test cases followed.

For each test case, the first line contains two integers $\mathbf{N}$ and $\mathbf{K}$ ($1 \leq \mathbf{K} \leq \mathbf{N} \leq 10^5$), the size of tree and the number of partitions respectively. Each of the following $n-1$ lines contains of two space-separated integers $\mathbf{U_i}$ and $\mathbf{V_i}$ ($1 \leq \mathbf{U_i}, \mathbf{V_i} \leq \mathbf{N}$; $\mathbf{U_i} \neq \mathbf{V_i}$), denoting the endpoints of the i-th edge. Finally, there is a line of $\mathbf{N}$ space-separated integers $\mathbf{W_i}$ ($1 \leq \mathbf{W_i} \leq 10^9$), which are the weights of the nodes.

The sum of values of $\mathbf{N}$ for all test cases does not exceed 2×10^5.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number and y is the answer.

Sample input and output

Sample Input	Sample Output
2 5 3 1 2 1 3 1 4 1 5 1 2 3 4 5 5 4 1 2 1 3 1 4 1 5 1 2 3 4 5	Case #1: 6 Case #2: 5

Note

For example 1, three partitions are $\{1,2,3\}, \{4\}, \{5\}$.

For example 2, four partitions are $\{1,2\}, \{3\}, \{4\}, \{5\}$.

Problem I. Portal

Input file: `standard input`
Output file: `standard output`

Alice and Bob are living in "Rectangle Kingdom". There are two portals in the world. Standing at the portal, people can jump into it, and instantly travel to the other portal. Portals do not disappear after you do that. Of course you can walk past a portal and do not jump into it, but why waste such fun?

It's known that Bob dates Alice so frequently that almost everyone in Rectangle Kingdom becomes jealous and wants to do something evil to stop them from seeing each other. You, for instance, want to separate them by moving their apartments to be as far as possible. Hopefully after your plan, Bob should spend much more effort walking to Alice's apartment, making him feel reluctant to do that, until eventually of course, they stop seeing each other and everyone is happy again.

You may assume "Rectangle Kingdom" is a 2D rectangle and Alice and Bob's apartments should be in the rectangle (the border included). The distances are Euclidean and walking one distance unit takes one time unit. The distance between two portals is 0. Furthermore, Bob always chooses the shortest path from his apartment to Alice's apartment.

Input

The first line is an integer $\mathbf{T}$, which is the number of test cases.

The next T lines each have six space-separated integers $\mathbf{A}, \mathbf{B}, \mathbf{X_1}, \mathbf{Y_1}, \mathbf{X_2}, \mathbf{Y_2}$. $\mathbf{A}, \mathbf{B}$ respectively denote the width and height of the kingdom and the kingdom is defined by 4 points: $(\mathbf{0}, \mathbf{0}), (\mathbf{0}, \mathbf{B}), (\mathbf{A}, \mathbf{B}), (\mathbf{A}, \mathbf{0})$. $(\mathbf{X_1}, \mathbf{Y_1}), (\mathbf{X_2}, \mathbf{Y_2})$ denote the position of two portals.

$1 \leq \mathbf{T} \leq 2000$

$1 \leq \mathbf{A}, \mathbf{B} \leq 10^3$

$0 \leq \mathbf{X_1}, \mathbf{X_2} \leq \mathbf{A}$

$0 \leq \mathbf{Y_1}, \mathbf{Y_2} \leq \mathbf{B}$

Output

For each test case, output one line containing "Case #x:", where x is the test case number, and next two lines "ax ay" and "bx by", denote the position of Alice's apartment and Bob's apartment after your evil moving of their apartments. The walking time of the shortest path from Bob's apartment to Alice's apartment should be maximum.

You answer is considered correct, if its absolute or relative error does not exceed 10^{-6}. Namely, let the walking time of your answer be a, and the walking time of jury's answer be b. Your answer is considered correct, if $\frac{|a-b|}{\max(1,|b|)} \leq 10^{-6}$.

Sample input and output

Sample Input	Sample Output
2 2 2 0 0 2 2 2 2 0 0 0 2	Case #1: 2.00000000 0.00000000 0.00000000 2.00000000 Case #2: 2.00000000 0.00000000 0.50000000 2.00000000

Problem J. Bob's Poor Math

Input file: `standard input`
Output file: `standard output`

Bob's math is really poor. He even doesn't know how does 'carry' works in plus operation. In his world, the 'plus' operation performed like this : $9 + 5 = 4$, $99 + 99 = 88$, $5876 + 5576 = 342$, $5555 + 5555 = 0$, $1503 + 2503 = 3006$, $512 + 1314 = 1826$. Oh how poor his math is and he could even be the problem setter. Tragedy. In his world, there is a 'Wish' data structure. Let's see how it works:

*. Initially, there will be a list of numbers to initialize the data structure. After initialization, game begins:

1. You may add a number to his data structure.

   Add x

2. You may divide all the numbers in the data structure by 10. (Note: it is integer division, for example, $9/10 = 0$, $99/10 = 9$, $5876/10 = 587$)

   $Shift$

3. You may query the largest result by given x 'plus' any number from his data structure.

   Query x

4. You may query 'sum' (by his poor math) of numbers between L and R inclusive. It means, you need to find all numbers x in his data structure that can meet $L \leq x \leq R$, and conduct the 'plus' one by one to get the 'sum'.

   Sum L R

Can you help naive Bob to implement data struct 'Wish'?

Input

At very beginning, there would be a number $\mathbf{T}$ indicates the number of tests. ($1 \leq \mathbf{T} \leq 100$)

For each test case:

The first line contains two integers N and M indicates the number of initial numbers and number of operstions. ($1 \leq \mathbf{N} \leq 10,000$ and $1 \leq \mathbf{M} \leq 10,000$)

The second line contains N integers for the initial numbers a_i. ($0 \leq a_i \leq 2 \times 10^9$)

The following M lines would follow the operation format #1 to #4.

For the operations, $0 \leq L \leq R \leq 2 \times 10^9$, $0 \leq x \leq 2 \times 10^9$

Output

For each test case, output one line containing "Case #x:", where x is the test case number (starting from 1). For each operation *Query* and *Sum*, output the answer

Sample input and output

Sample Input	Sample Output
1 4 5 1 2 88 29 Add 44 Query 71 Shift Sum 0 10 Query 91	Case #1: 90 4 99

Note

For Sample Case #1:

- Initially: [1 , 2 , 29 , 88]
- Add 44: [1 , 2 , 29 , 88 , 44]
- Query 71: $29 + 71 = 90$; $88 + 71 = 59$; $44 + 71 = 15$
- Shift: [0 , 0 , 2 , 8 , 4]
- Sum 0 10: $0 + 0 + 2 + 8 + 4 = 4$
- Query 91: $91 + 8 = 99$

Problem K. Color Graph

Input file: standard input
Output file: standard output

You are given a simple graph with $\mathbf{N}$ vertices and $\mathbf{M}$ edges. A simple graph is an undirected graph that has no self-loops (edges connected at both ends to the same vertex) and no more than one edge between any two different vertices. Initially, each edge is colored white. Each round you can pick a white edge and paint it in red. You are not allowed to generate an odd-length cycle that consists only of red edges. What's the maximum number of edges that can be painted to red?

Input

The first line of the input gives the number of test case, $\mathbf{T}$ ($1 \leq \mathbf{T} \leq 30$). $\mathbf{T}$ test cases follow.

For each case, the first line contains only two integers $\mathbf{N}$ and $\mathbf{M}$ ($2 \leq \mathbf{N} \leq 16$, $1 \leq \mathbf{M} \leq \mathbf{N}(\mathbf{N}-1)/2$). The i-th line of the next $\mathbf{M}$ lines describes the i-th edge: two integers u, v denotes an edge between u and v. It's guaranteed that there are no self-loops and multiple edges.

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 edges that can be painted to red.

Sample input and output

Sample Input	Sample Output
2 4 5 1 2 1 3 1 4 2 4 3 4 5 6 1 2 2 3 3 1 1 4 4 5 3 5	Case #1: 4 Case #2: 5

Problem L. Light It Down

Input file: standard input
Output file: standard output

You are given a tree with $\mathbf{N}$ nodes. The tree's nodes are numbered 1 through $\mathbf{N}$ and its edges are numbered 1 through $\mathbf{N-1}$. Each edge is associated with a distance.

There are $\mathbf{M}$ lightbulbs, numbered from $\mathbf{1}$ to $\mathbf{M}$. For each lightbulb i you are given an integer p_i - the node number where the lightbulb is, and no two lightbulbs in the same node. Each lightbulb can either be ON or OFF. Each input case will contain the initial state.

You're starting from node $\mathbf{S}$, and need to turn off all lightbulbs. And we can guarantee that there is no lightbulb at the starting point $\mathbf{S}$, and there is no more than one lightbulb in one node.

You can repeat the following operation until all lightbulbs are OFF: If you are currently located in node u, you will randomly select a new node v with equal probability from $\{v\colon v \neq u$, v should have no lightbulb if u has a lightbulb$\}$. Then you will move to node v by the shortest path and flip (ON $\rightarrow$ OFF, OFF $\rightarrow$ ON) the lightbulb in v if node v has lightbulb.

Your task is to compute the expected total distance to light down all lightbulbs.

Input

The first line of the input gives the number of test case, $\mathbf{T}$ ($1 \leq \mathbf{T} \leq 10$). $\mathbf{T}$ test cases follow.

For each case, the first line contains three integers $\mathbf{N}$, $\mathbf{M}$, and $\mathbf{S}$ ($1 \leq \mathbf{N} \leq 100,000$; $1 \leq \mathbf{M} < \mathbf{N}$; $1 \leq \mathbf{S} \leq \mathbf{N}$) – the number of nodes, the number of lightbulbs, and the start position.

The i-th line of the next $\mathbf{N-1}$ lines describes the i-th edge: three integers u, v, w ($1 \leq u, v \leq \mathbf{N}$; $u \neq v$; $1 \leq w \leq 100,000$) denotes an edge between u and v with distance w.

The i-th line of the next $\mathbf{M}$ lines contains two integers p_i ($1 \leq p_i \leq \mathbf{N}$; $p_i \neq \mathbf{S}$) and s_i ($0 \leq s_i \leq 1$) – the positions of the i-th lightbulb and the initial states of the i-th lightbulb, where 1 means ON and 0 mean OFF.

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 answer modulo (10^9+7). More specifically, if the answer can be formed as an irreducible fraction $\frac{A}{B}$, then y will be $(A \cdot B^{-1})$ mod (10^9+7).

Sample input and output

Sample Input	Sample Output
3 3 2 1 1 2 1 1 3 1 2 1 3 1 3 1 1 1 2 1 1 3 1 2 1 3 1 3 1 2 1 1 3 1 2 1	Case #1: 7 Case #2: 333333338 Case #3: 666666674

Problem M. Blood Pressure Game

Input file: `standard input`
Output file: `standard output`

Gugulu, a JBer, who was an ACMer one year ago, comes to Shanghai University taking part in the International Collegiate Programming Contest again. However, every time when Gugulu came to Shanghai University, he always got an Iron Medal and would consider this competition JB-like. To release his pain, Gugulu would go to the Shanghai Disneyland Park to have fun in taking the Roller Coaster. Gugulu loves the feeling with high-level blood pressure so much which makes him feel as a happy flappy bird who forgets all the Wrong Answers and Time Limit Exceededs etc.

We can regard the path of the Roller Coaster as a list of turning points with different heights which can be represented as an array $\{a_1, a_2, a_3, \ldots, a_n\}$ of size n, and Gugulu's final blood pressure after the game of the Roller Coaster is counted as the sum of all absolute values of the differences between the $n-1$ pairs of adjacent array numbers, i.e. $\sum_{i=1}^{n-1} |a_i - a_{i+1}|$.

Gugulu always got Iron Medals and is always getting Iron Medals, which makes him keep taking the Roller Coaster over and over again. However, as playing more games, his threshold on the value of blood pressure which can make himself happy is keeping increasing. As a result, the Roller Coaster of Shanghai Disneyland Park can hardly meet Gugulu's need anymore.

Therefore, Gugulu decides to add a set of m extra turning points in any order into this path, however, to consider about the reality on the distance of the original path, he can add at most one turning point into any original position between any two original elements in the array (and at most one in the head, at most one in the tail). Gugulu wants to make his blood pressure as high as possible, and he wants to know how much his blood can reach at most when he has added $\{1, 2, 3, \ldots, m\}$ extra Roller Coaster turning points into the path.

You, another JBer, are sure that Gugulu is clever enough to get the highest blood pressure as he can. It is very important for you to calculate the exact numbers to make an appointment with a proper cardiologist in advance to save Gugulu's life. You must solve this problem! Gugulu's blood pressure is becoming out of the control!

Input

The first line of the input gives the number of test cases, $\mathbf{T}$ ($1 \le \mathbf{T} \le 1000$). $\mathbf{T}$ test cases follow.

For each test case, the first line contains two integers, n ($1 \le n \le 600$) and m ($1 \le m \le n+1$)

Then, in second line, there are n integers $\{a_1, a_2, a_3, \ldots, a_n\}$ $(1 \leq a_i \leq 10^9)$, denoting the n original Roller Coaster turning points. Then, in the third line, there are m integers $\{b_1, b_2, b_3, \ldots, b_m\}$ $(1 \leq b_i \leq 10^9)$, denoting the m addition Roller Coaster turning points to be added. As it is an exciting Roller Coaster, it is guaranteed that all $n + m$ integers in the two arrays are pairwise different.

There are at most 10 test cases whose n is more than 100.

Output

For each test case, output three lines. The first line contains "Case #x:", where x is the test case number (starting from 1). In the second line, there should be m integers which represent how much Gugulu's blood pressure could reach if he has added $\{1, 2, 3, \ldots, m\}$ extra Roller Coaster turning points into the path. In the third line, there should be $n+m$ integers which represent the heights of the final Roller Coaster path if he has added all m extra turning points. If there are several solutions, output any of them.

Sample input and output

Sample Input	Sample Output
6 2 3 5 11 10 3 1 4 1 1 2 3 4 5 4 2 1 2 3 4 5 6 4 5 1 2 3 4 5 6 7 8 9 4 4 10 50 3 6 1 9 23 5 4 2 10 50 3 6 9 23	Case #1: 16 22 27 10 5 1 11 3 Case #2: 9 1 5 2 3 4 Case #3: 11 15 1 6 2 5 3 4 Case #4: 17 27 33 38 39 6 1 9 2 8 3 7 4 5 Case #5: 124 142 147 150 5 10 1 50 3 23 6 9 Case #6: 124 127 10 50 3 23 6 9