Problem A. Bus Routes

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

In order to develop local tourism, Major Fu plans to introduce several bus routes between attractions in his city. There are M types of buses, namely the Green Bus, the Red Bus, the Blue Bus, etc., and N attractions in his city. He wants the arrangement to satisfy the following requirements:

- There is at most one type of buses between any two attractions.
- Visitors can travel between any two attractions by bus.
- Many visitors likes circle trip, in which the visitors start from an attraction and get back to the attraction via a sequence of other attractions. At least one circle trip should be made available via buses. The circle trips do not have to contain all attractions.

Can you help him to calculate the number of ways he can arrange the bus routes? Please note that you do not need to use all types of buses in each arrangement and bus routes are two-way.

Input

The first line of input contains a number T indicating the number of test cases ($1 \leq T \leq 200$). For each case, there is only one single line containing two non-negative integers N and M ($1 < N \leq 10^4$, $1 \leq M < 2^{31}$).

Output

For each test case, output a single line consisting of "`Case #X: Y`". X is the test case number starting from 1. Y is the answer modulo 152076289.

Example

standard input	standard output
3 3 1 3 2 4 2	Case #1: 1 Case #2: 8 Case #3: 496

Problem B. The Robot on the Plane

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

The robot moves along the plane step-by-step by the following rules:

1. The length of move at the first step is 1.
2. The length of move at each subsequent step is exactly three times longer than at previous step.
3. At each step robot may either to rest ('S'), i.e. do not move, or select one of the four directions — up ('U'), down ('D'), left ('L') or right ('R') and then go in the selected direction by a length equal to the length of the move.

Given two points (x_0, y_0) and (x_1, y_1) on the plane, determine, whether the robot can start his travel in the first point and reach second point. If second point can be reached, print the sequence of letters, corresponding to robot's moves. If there is more than one answer, print any of them.

Input

First line of the input contains four integers — coordinates of starting point x_0 and y_0 and coordinates of ending point x_1 and y_1. All coordinates does not exceed 10^{17} by absolute value.

Output

If it is impossible for robot to reach the ending point, printf 'NO' in the first line. Otherwise print 'YES', and in second line print the sequence of steps — not more than 10^5 characters 'U', 'D', 'R', 'L', 'S'.

Note that 'S' may be last move only in case, when no more steps exist in the route (i.e. only when points coincide).

Examples

standard input	standard output
0 0 1 1	NO
1 1 4 2	YES UR
-732 -732 -732 -732	YES S

Problem C. Autopilot System

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

Farmer John has just purchased a car equipped with an autopilot system. He is so excited about this high-tech product and can't wait to try it. John plans to travel to another city this weekend, so he decides to let the autopilot system lead the way.

There are N cities numbered from 1 to N inclusively. John lives in city 1 and he plans to travel to city N. The cities are connected by $N-1$ highways. Any two cities are reachable to each other via the highways. Every time John passes through a high way, he needs to pay 1 unit of money as the toll fee.

Obviously, such a road system is very busy, since there is only one path between two cities. To solve this problem, M cities are selected as transport hubs. For each city i (including the hubs), new one-way roads are constructed, each of which allows travelling from city i to one of the hubs, but travelling from the hub back to the city is not allowed. More importantly, they are free of charge!

Now Farmer John has started the autopilot system, waiting for it to lead him from city 1 to city N. Unfortunately, something goes wrong with its software, so it would pilot at random. When the car is located in a city, the system will pick the next destination as the following. For the i-th transport hub, the system will select it as the next destination with probability q_i, and travel to the hub via the free one-way road. If none of the transport hubs is selected, it drives the car to one of the neighboring city through a highway with uniform probability. In other words, denoting the sum of q_i as Q and the city has K neighbors, the probability of traveling to one of the neighboring cities is $(1-Q)/K$. The only good news is that the system will stop piloting immediately when the car arrives in city N.

Farmer John is aware of this malfunctioning now. He wants to know the expected amount of money that he needs to pay for the toll fee, if he lets the car travel at random as described above.

Input

The first line of input contains a number T indicating the number of test cases ($1 \le T \le 150$). Each test case starts with a line containing two N integers and M ($2 \le N \le 2 \cdot 10^4$, $0 \le M \le 200$). The i-th of the next M lines contains two numbers u_i and q_i, indicating that u_i is a transport hub and the probability of travelling to it from any city is q_i. It is guaranteed that all transport hubs are distinct and the sum of q_i is not greater than 1. Each of the next $N-1$ lines contains two integers x and y, indicating that there is a highway between city x and city y.

Output

For each test case, output a single line consisting of "`Case #X: Y`". X is the test case number starting from 1. Y is the expected units of money that is payed as toll fee. Your answer is considered correct if either of its relative error or absolute error is less than 10^{-6}. If it is impossible to travel to city N, you should output -1 instead.

Example

standard input	standard output
3 6 3 2 0.2 3 0.3 4 0.1 1 2 1 3 2 4 2 5 2 6 6 3 1 0.2 3 0.1 6 0.1 1 2 1 3 2 4 2 5 2 6 3 2 1 0.5 2 0.5 1 2 1 3	Sample Output Case #1: 13.176471 Case #2: 4.694250 Case #3: -1

Problem D. Immortality of Frog

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	512 mebibytes

N frogs are attempting to prolong their life-span. They live in the bottom of a well which can be described as a two-dimensional $N \times N$ grid. Grid is located in the i-th row and the j-th column. At the beginning, the i-th frog lives in the bottom of i-th column, i.e. the place below grid $(1, i)$.

The frogs are so devout that God decides to give them a chance. In each row i, a horizontal membrane ranging from (i, L_i) to (i, R_i) inclusively is created. A capsule of elixir is placed at one of the grids of the membrane with uniform probability.

Now the frogs are jumping upwards to pursue immortality. The i-th frog would be in grid (j, i) after j jumps. When a frog arrives at a grid that contains a capsule of elixir, it will eat the capsule and gain immortality. After that, it continues jumping upwards until it gets out of the well.

A membrane is considered *bad* if it covers less than N grids. The frogs are very sensitive, so they can only endure passing through 10 bad membrane. When a frog reaches the 11^{th} bad membrane, it thinks that there is no hope to get out of the well, so it will go back to the bottom of well and live there until death, even though it has eaten a capsule of elixir already.

The frogs are friends, so they want all of them gain immortality and live a happy life out of the well. They want to know the probability P that every frog eats exactly one capsule of elixir and gets out of the well.

Input

The first line of input contains a number T indicating the number of test cases ($1 \le T \le 1000$).

Each test case starts with a line containing an integer N as described above ($1 \le N \le 1000$). The second line contains N space separated integers $L_1, L_2, \ldots, L_N$. The third line contains N space separated integers $R_1, R_2, \ldots, R_N$ ($1 \le L_i \le R_i \le N$).

Output

For each test case, output a single line consisting of "`Case #X: Y`". X is the test case number starting from 1. Y is an integer defined as the probability P multiplies $\prod_{i=1}^{N}(R_i - L_i + 1)$. As the answer could be huge, you only need to output it module 105225319.

Example

standard input	standard output
2 2 1 2 1 2 2 1 1 2 2	Case #1: 1 Case #2: 2

Problem E. Land of Farms

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Farmer John and his brothers have found a new land. They are so excited and decide to build new farms on the land. The land is a rectangle and consists of $N \times M$ grids. A farm consists of one or more connected grids. Two grids are adjacent if they share a common border, i.e. their Manhattan distance is exactly 1. In a farm, two grids are considered connected if there exist a series of adjacent grids, which also belong to that farm, between them.

Farmer John wants to build as many farms as possible on the new land. It is required that any two farms should not be adjacent. Otherwise, sheep from different farms would fight on the border. This should be an easy task until several ancient farms are discovered.

Each of the ancient farms also consists of one or more connected grids. Due to the respect to the ancient farmers, Farmer John do not want to divide any ancient farm. If a grid from an ancient farm is selected in a new farm, other grids from the ancient farm should also be selected in the new farm. Note that the ancient farms may be adjacent, because ancient sheep do not fight each other.

The problem is a little complicated now. Can you help Farmer John to find a plan with the maximum number of farms?

Input

The first line of input contains a number T indicating the number of test cases ($1 \le T \le 200$).

Each test case starts with a line containing two integers N and M, indicating the size of the land. Each of the following N lines contains M characters, describing the map of the land ($1 \le N, M \le 10$). A grid of an ancient farm is indicated by a single digit ('0'-'9'). Grids with the same digit belong to the same ancient farm. Other grids are denoted with a single character '.'. It is guaranteed that all test cases are valid.

Output

For each test case, output a single line consisting of "`Case #X: Y`". X is the test case number starting from 1. Y is the maximum number of new farms.

Example

standard input	standard output
3 3 4 ..3. 023. .211 2 3 ... ... 4 4 1111 1..1 1991 1111	Case #1: 4 Case #2: 3 Case #3: 1

Problem F. Matching Compressed String

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	512 mebibytes

You are given a long string and looking for certain patterns in the string.

The string contains only lowercase letters ('a-'z'), and it is represented in a compressed format. Denoting $S_1, S_2, \ldots$ as compressed strings, another compressed string S is defined recursively in one of the following ways:

- S can be any string consisting of only lowercase letters ('a-'z').
- S can be generated by repeating another string for any times. Specifically, S is represented as "$R(S_1)$", which means that the content of S_1 is repeated R times.
- S can also be the concatenation of other strings. Specifically, S is represented as "$S_1S_2 \ldots S_L$", which means S is the concatenation of $S_1, S_2, \ldots, S_L$.
- An empty string ("") is also a valid representation.

Formally, the Backus–Naur Form (BNF) specification of the syntax is

```
<compressed> ::= "" | <lowercase-letter> | <compressed> <compressed> | <number> "(" <compressed> ")"
```

For example, the string "baaabbaaab" can be compressed as "b3(a)2(b)3(a)b". It can also be compressed as "2(b3(a)b)". On the other hand, you find deterministic finite automaton (DFA) as powerful way to describe the patterns you are looking for. A DFA contains a finite set of states Q and a finite set of input symbols called the alphabet A. Initially, the DFA is positioned at the start state $q_0 \in Q$. Given the transition function $f(q, a)$ and an input symbol a, the DFA transit to state $f(q, a)$ if its current state is q .

Let $w = a_1a_2 \ldots a_n$ be a string over the alphabet A. According to the above definition, the DFA transits through the following sequence of states:

$q_0, q_1 = f(q_0, a_1), q_2 = f(q_1, a_2), \ldots, q_n = f(q_{n-1}, a_n)$.

The DFA also contains a set of accept states $F \subseteq Q$. If the last state q_n is an accept state, we say that the DFA accepts the string w. The set of accepted strings is referred as the language that the DFA represents.

Now you are given a compressed string S and a DFA Z. You want to know if Z accepts the decompressed content of S.

Input

The first line of input contains a number T, indicating the number of test cases ($1 \le T \le 200$).

The first line of each test case contains a non-empty compressed string S, as described above. The length of S is not greater than 10^4, and $0 \le R \le 10^9$. It is guaranteed that the representation of S is valid.

The description of the DFA follows.

The first line of the description contains three integers N, M, and K, indicating the number of states, the number of rules describing the transition function, and the number of accept states ($1 \le K \le N \le 1000$, $0 \le M \le 26N$). The states are numbered from 0 to $N - 1$. The start state is always 0.

The second line contains K integers representing the accept states. All these numbers are distinct.

Each of the next M lines consists of two states p and q, and an input symbol a, which means that the DFA transits from p to q when it receives the symbol a. The symbol a is always a lowercase letter. It is guaranteed that, given p and a, the next state q is unique.

Output

For each test case, output a single line consisting of "`Case #X: Y`". X is the test case number starting from 1. Y is "`Yes`" if the DFA accepts the string, or "`No`" otherwise.

Example

standard input	standard output
3 2(b3(a)b) 2 3 1 0 0 1 b 1 0 b 1 1 a b3(a)2(b)3(a)b 2 2 1 1 0 1 b 1 0 a b3(a)2(b)3(a)b 2 4 1 0 0 1 b 0 1 a 1 0 a 1 0 b	Case #1: Yes Case #2: No Case #3: Yes

Problem G. Alice's Classified Message

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	512 mebibytes

Alice wants to send a classified message to Bob. She tries to encrypt the message with her original encryption method. The message is a string S, which consists of N lowercase letters.

$S[a \dots b]$ means a substring of S ranging from $S[a]$ to $S[b]$ $(0 \le a \le b < N)$. If the first i letters have been encrypted, Alice will try to find a magic string P. Assuming P has K letters, P is the longest string which satisfies $P = S[T \dots T+K-1]$ $(0 \le T \le i, T+K \le N)$ and $P = S[i \dots i+K-1]$ $(i+K \le N)$. In other words, P is a substring of S, of which starting address is within $[0 \dots i-1]$, and P is also a prefix of $S[i \dots N-1]$. If P exists, Alice will append integer K and T to ciphertext. If T is not unique, Alice would select the minimal one. And then i is incremented by K. If P does not exist, Alice will append -1 and the ASCII code of letter $S[i]$ to ciphertext, and then increment i by 1.

Obviously the first letter cannot be encrypted. That is to say, P does not exist when $i = 0$. So the first integer of ciphertext must be -1, and the second integer is the ASCII code of $S[0]$.

When $i = N$, all letters are encrypted, and Alice gets the final ciphertext, which consists of many pairs of integers. Please help Alice to implement this method.

Input

The first line of input contains an integer T, which represents the number of test cases $(1 \le T \le 50)$. Each test case contains a line of string, which has no more than 10^5 lowercase letters. It is guaranteed that the total length of the strings is not greater than $2 \cdot 10^6$.

Output

For each test case, output a single line consisting of "`Case #X:`" first. X is the test case number starting from 1. Output the ciphertext in the following lines. Each line contains two integers separated by a single space.

Example

standard input	standard output
2 aaaaaa aaaaabbbbbaaabbc	Case #1: -1 97 5 0 Case #2: -1 97 4 0 -1 98 4 5 5 2 -1 99

Problem H. Frog and String

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Frog studies algorithms on strings. He finds it so interesting that he can't stop playing with his strings. These days he has just learnt about palindrome, and comes up with a problem about it.

Given two integers, N and M, he wants to construct a string of length N, whose substrings contain exactly M distinct non-empty palindromes. A palindrome is a string which is exactly the same as the reverse of itself. For example, "`ABBA`", "`ADA`", "`A`", and "`UUSSUU`" are palindromes, but "`USTC`", "`AB`", and "`ABC`" are not. A substring is a consecutive part of the original string. For example, "`US`", "`USTC`", "`STC`", and "`TC`" are substrings of "`USTC`", but "`UC`" and "`CT`" are not.

Frog finds it too hard for him to solve this problem. So he asks you for help. BTW, he won't make it too easy for you, so he decided to ask you solve this problem under his restrictions. You can only use the first K capital letters in the English alphabet ('`A`'-'`Z`'). Please write a program to solve this problem.

Input

There is an integer T in the first line indicating the number of total test cases ($1 \le T \le 2 \cdot 10^4$). Each test case contains three integers N, M, and K ($1 \le N, M \le 10^5$, $1 \le K \le 26$), separated by single spaces. We guarantee the sum of N will not exceed $2 \cdot 10^6$.

Output

For each test case, output a single line consisting of "`Case #X:`" first, where X is the test case number starting from 1. Output the string that you find in the next line. The string should contain only the first capital letters. If there are multiple solutions, you can output any of them. If there is no such string satisfying Frog's requirements, output "`Impossible`" instead. Please follow the output format exactly, and do not output any additional character or new line.

Example

standard input	standard output
4 3 3 3 4 4 4 2 2 1 2 1 1	Case #1: ABA Case #2: ABCD Case #3: AA Case #4: Impossible

Note

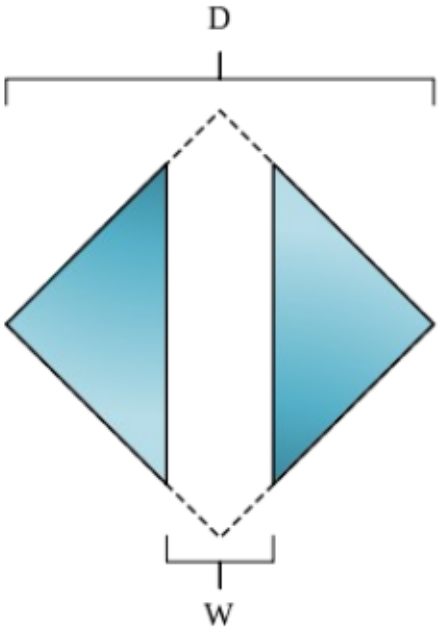
For the first test case, "`A`", "`ABA`", "`B`" are the all distinct palindrome substrings of "`ABA`". There are other possible answers, such as "`BAB`" and "`AAA`". For the second test case, "`USTC`" is not a valid answer, because it contains letters other than the first 4 capital letters.

Problem I. The Shields

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	512 mebibytes

General Zhang recently got a plasma shield to intercept incoming missiles.

The shape of the plasma shield is basically a square with a horizontal diagonal. Due to some technology issues, the plasma shield cannot protect the whole square area. One vertical stripe in the middle of the square is penetrable to the missiles. So, in fact the shield is formed by two identical isosceles right triangle. Missiles hitting the surface or the edge of the shield are considered intercepted.



Now missiles are coming to General Zhang's army. The i-th missile will strike the position (x_i, y_i) . Because he only has one plasma shield, he want to calculate when the shield is turned on in a particular way, how many of the missiles it can stop.

General Zhang will give you M queries, each query describe the shape of a plasma by 4 integers X, Y, D and W. (X, Y) denotes the center of the shield. D denotes the length of the diagonal. W denotes the width of the unprotected stripe. For each query, you need to output number of the missiles stopped by the shield.

Input

The first line of input contains an integer T $(1 \leq T \leq 20)$, which represents the number of test cases. Each test case starts with N and M in a line $(1 \leq N, M \leq 2 \cdot 10^4)$. The i-th line of the next N lines contains two integers x_i and y_i. The next M lines each line has 4 integers X, Y, D and W, which represents a query. D and W are guaranteed to be even numbers. All coordinates does not exceed $2 \cdot 10^9$ by absolute value.

Output

For each test case, output a single line consisting of "`Case #X:`" first. X is the test case number starting from 1. For each query, output the number of missiles that are intercepted in a single line. Do not output extra spaces or newlines.

Example

standard input	standard output
1 4 3 -5 0 -4 0 -3 0 0 0 0 0 0 0 0 0 8 0 0 0 8 2	Case #1: 1 3 2

Problem J. Kingdom of Tree

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	512 mebibytes

The Kingdom of Tree has cities numbered from 1 to N. The N cities are connected by $N-1$ roads, and any two cities are reachable via a sequence of roads. Obviously, the road system of the Kingdom of Tree is constructed according to a tree structure. That's how the kingdom's name comes.

To monitor and protect the roads, the general of the kingdom decides to build radar stations in the cities. We denote the radius of the radar station located in city i as R_i. R_i is a non-negative integer, and $R_i = 0$ indicates that there is no radar station in city i. Given a road connecting city i and city j, whose length is denoted as $L_{i,j}$, we say that the road is monitored by radar if $R_i + R_j \geq L_{i,j}$.

The cost of building a radar station in city i is proportional to its radius R_i. Given a construction plan of the radar stations $P = \{R_1, R_2, \ldots, R_N\}$, we denote the set of monitored roads as S. The cost per length is thus defined as the ratio between the sum of R_i and the sum of the length of monitored roads, i.e $\sum_{i=1}^{N} R_i / \sum_{(i,j)\in S} L_{i,j}$.

Due to limited budget, the general wants to make the most of the money. He wants a construction plan that minimizes the cost per length. Note that he don't have to monitor all the roads. As the most brilliant programmer of the Kingdom of Tree, can you help him to find such a construction plan?

Input

The first line of input contains a number T indicating the number of test cases ($1 \leq T \leq 200$).

Each test case starts with a line containing an integer N indicating the number of cities ($2 \leq N \leq 30$). Each of the next $N-1$ lines contains three integers i, j, and k, indicating that there is a road between city i and city j, and its length is k ($1 \leq i, j \leq N$, $i \neq j$, $1 \leq k \leq 10^9$).

It is guaranteed that any two cities are reachable via a sequence of roads.

Output

For each test case, output a single line consisting of "`Case #X: Y`". X is the test case number starting from 1. Y is the minimum cost per length. You answer is considered correct if its absolute error is less than 10^{-6}.

Example

standard input	standard output
4 7 1 2 1 2 3 2 3 4 99 4 5 3 5 6 1 6 7 4 4 1 2 3 2 3 2 2 4 1 5 1 2 2 2 3 1 3 4 2 3 5 5 2 1 2 3	Case #1: 0.66666667 Case #2: 0.50000000 Case #3: 0.62500000 Case #4: 1.00000000