Problem A. Minimum's Revenge

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

There is a graph of n vertices which are indexed from 1 to n. For any pair of different vertices, the weight of the edge between them is the least common multiple of their indexes.

Mr. Frog is wondering about the total weight of the minimum spanning tree. Can you help him?

Input

The first line contains only one integer T ($T \leq 100$), which indicates the number of test cases.

For each test case, the first line contains only one integer n ($2 \leq n \leq 10^9$), indicating the number of vertices.

Output

For each test case, output one line "`Case #x: y`", where x is the case number (starting from 1) and y is the total weight of the minimum spanning tree.

Example

standard input	standard output
2 2 3	Case #1: 2 Case #2: 5

Problem B. Prediction

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

There is a graph $G = \langle V_G, E_G \rangle$ with $|V_G| = n$ and $|E_G| = m$, and a magic tree $T = \langle V_T, E_T \rangle$, rooted at 1, which contains m vertices.

Each vertex of the magic tree corresponds to an edge in the original graph G and each edge occurs in the magic tree exactly once.

Each query includes a set S ($S \subseteq V_T$), and you should tell Mr. Frog the number of components in the modified graph $G^{`} = (V_G, E^{`}_G)$, where $E^{`}_G$ is a set of edges in which every edge corresponds to a vertex v in magic tree T satisfying at least one of the following two conditions:

- $v \in S$.
- v is an ancestor of some vertices in S.

Note that the queries are independent, and namely one query will not influence another.

Input

The first line of the input contains two integers n and m ($1 \le n \le 500$, $1 \le m \le 10^4$), where n is the number of vertices and m is the number of edges.

The second line contains $m - 1$ integers describing the magic tree, i-th integer represents the parent of the $(i + 1)$-th vertex.

Then the following m lines describe the edges of the graph G. Each line contains two integers u and v indicating the two ends of the edge.

The next line contains only one integer q ($1 \le q \le 5 \cdot 10^4$), indicating the number of queries.

Then the following q lines represent queries, i-th line represents the i-th query, which contains an integer k_i followed by k_i integers representing the set S_i.

It is guaranteed that $\sum_{i=1}^{q} k_i \le 3 \cdot 10^5$.

Output

For each query, output a single line containing only one integer representing the answer, namely the number of components.

Example

standard input	standard output

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

Problem C. Mr. Frog's Problem

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

One day, you, a clever boy, feel bored in your math class, and then fall asleep without your control. In your dream, you meet Mr. Frog, an elder man. He has a problem for you.

He gives you two positive integers A and B, and your task is to find all pairs of integers (C, D), such that $A \le C \le B$, $A \le D \le B$ and $\frac{A}{B} + \frac{B}{A} \le \frac{C}{D} + \frac{D}{C}$

Input

First line contains only one integer T ($T \le 125$), which indicates the number of test cases. Each test case contains two integers A and B ($1 \le A \le B \le 10^{18}$).

Output

For each test case, first output one line "`Case #x:`", where x is the case number (starting from 1).

Then in a new line, print an integer s indicating the number of pairs you find.

In each of the following s lines, print a pair of integers C and D. Pairs should be sorted by C, and then by D in ascending order.

Example

standard input	standard output

2
10 10
9 27
	Case #1:
	1
	10 10
	Case #2:
	2
	9 27
	27 9

Problem D. Coconuts

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

TanBig, a friend of Mr. Frog, likes eating very much, so he always has dreams about eating. One day, TanBig dreams of a field of coconuts, and the field looks like a large chessboard which has R rows and C columns. In every cell of the field, there is one coconut. Unfortunately, some of the coconuts have gone bad. For sake of his health, TanBig will eat the coconuts following the rule that he can only eat good coconuts and can only eat a connected component of good coconuts one time (you can consider the bad coconuts as barriers, and the good coconuts are 4-connected, which means one coconut in cell (x, y) is connected to $(x-1, y)$, $(x+1, y)$, $(x, y+1)$, $(x, y-1)$.

Now TanBig wants to know how many times he needs to eat all the good coconuts in the field, and how many coconuts he would eat each time(the area of each 4-connected component).

Input

The first line contains two integers R and C, $0 < R, C \leq 10^9$, the second line contains an integer n, the number of bad coconuts, $0 \leq n \leq 200$ from the third line, there comes n lines, each line contains two integers, x_i and y_i, which means in cell(x_i, y_i), there is a bad coconut.

It is guaranteed that in the input data, the first row and the last row will not have bad coconuts at the same time, the first column and the last column will not have bad coconuts at the same time.

Output

Print integer k, denoting the number of times TanBig needs, and in the second line, k integers denoting the number of coconuts he would eat each time; you should output them in increasing order.

Example

standard input	standard output

3 3
2
1 2
2 1
	2
	1 6

3 312 218	

Problem E. Mr. Frog's Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

One day, Mr. Frog is playing Link Game (Lian Lian Kan in Chinese).

In this game, if you can draw at most three horizontal or vertical head-and-tail-connected lines over the empty grids (the lines can be out of the whole board) to connect two non-empty grids with the same symbol or the two non-empty grids with the same symbol are adjacent, then you can change these two grids into empty and get several more seconds to continue the game.

Now, Mr. Frog starts a new game (that means there is no empty grid in the board). If there are no pair of grids that can be removed together, Mr. Frog will say "Im angry" and criticize you.

Mr. Frog is battle-scarred and has seen many things, so he can check the board in a very short time, maybe one second. As a Hong Kong Journalist, what you should do is to check the board more quickly than him, and then you can get out of the room before Mr. Frog being angry.

Input

The first line contains only one integer T ($T \leq 500$), which indicates the number of test cases.

For each test case, the first line contains two integers n and m ($1 \leq n, m \leq 30$).

In the next n lines, each line contains m integers, j-th number in the i-th line means the symbol on the grid(the same number means the same symbol on the grid).

Output

For each test case, there should be one line in the output.

You should output "`Case #x:  y`, where x is the case number(starting from 1), and y is a string representing the answer of the question. If there are at least one pair of grids that can be removed together, the y is "`Yes`" (without quote), else y is "`No`.

Example

standard input	standard output

2
3 3
1 2 1
2 1 2
1 2 1
3 3
1 2 3
2 1 2
3 2 1
	Case #1: Yes
	Case #2: No

Problem F. Auxilary Set

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

Given a rooted tree with n vertices, some of the vertices are important.

An auxiliary set is a set containing vertices satisfying at least one of the two conditions:

- It is an important vertex.
- It is the least common ancestor of two different important vertices.

You are given a tree with n vertices (1 is the root) and q queries. Each query is a set of nodes which indicates the unimportant vertices in the tree. Answer the size (i.e. number of vertices) of the auxiliary set for each query.

Input

The first line contains only one integer T ($T \leq 1000$), which indicates the number of test cases.

For each test case, the first line contains two integers n ($1 \leq n \leq 10^5$) and q ($0 \leq q \leq 10^5$).

In the following $n-1$ lines, the i-th line contains two integers u_i, v_i ($1 \leq u_i, v_i \leq n$) indicating there is an edge between u_ii and v_i in the tree.

In the next q lines, the i-th line first comes with an integer m_i ($1 \leq m_i \leq 10^5$) indicating the number of vertices in the query set.Then comes with m_i different integers, indicating the nodes in the query set.

It is guaranteed that $\sum_{i=1}^{q} m_i \leq 10^5$.

It is also guaranteed that the number of test cases in which $n \geq 1000$ or $\sum_{i=1}^{q} m_i \geq 1000$ is no more than 10.

Output

For each test case, first output one line "`Case #x:`", where x is the case number (starting from 1).

Then q lines follow, i-th line contains an integer indicating the size of the auxiliary set for each query.

Example

standard input	standard output

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

Problem G. Birthday Gift

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

Both Mr. Frog and Wallice love running. Wallice missed Mr. Frogs birthday recently, so he decided to give a belated birthday gift. He quickly found out an idea — he decided to run a closed curve to contain those meaningful buildings. Unfortunately, he only gets a little time left since he is going to attend an important press conference.

Wallice wants to know the maximal number of buildings can be contained in the closed curve. Note that his speed is 1.

Input

The first line of the input contains an integer N ($1 \le N \le 80$), and a double t ($0 \le l \le 5000$) indicating the numbers of buildings Wallice cares about and the time he has.

In the following n lines, the i-th line contains two doubles x_i, y_i ($-600 \le x_i, y_i \le 600$) indicating the position of the buildings.

It is guaranteed that the answer would not change even if l changes up to 10^{-5}, and there would not be any 3 points on one line even if any point changes its position up to 10^{-5}.

Output

Print the maximum number of buildings Wallice can circled in in limited time.

Example

standard input	standard output

`4 4.1`
`0 0`
`0 1`
`1 0`
`1 1`
	`4`

`4 3.5`
`0 0`
`0 1`
`1 0`
`1 1`
	`3`

Note

For the second sample, Wallice does not have enough time to circle all the four buildings so he circles three of them instead.

Problem H. Basic Data Structure

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

Mr. Frog learned a basic data structure recently, which is called stack.There are some basic operations of stack:

- `PUSH x` — put x on the top of the stack, x must be 0 or 1.
- `POP` — throw the element which is on the top of the stack.

Since it is too simple for Mr. Frog, a famous mathematician who can prove "Five points coexist with a circle" easily, he comes up with some exciting operations:

- `REVERSE` — Just reverse the stack, the bottom element becomes the top element of the stack, and the element just above the bottom element becomes the element just below the top elements... and so on.
- `QUERY` — Print the value which is obtained with such way: Take the element from top to bottom, then do NAND operation one by one from left to right, i.e. if $a_{top}, a_{top-1}, \cdots, a_1$ is corresponding to the element of the Stack from top to the bottom, $value = a_{top}$ nand a_{top-1} nand ... nand a_1. Note that the Stack will not change after QUERY operation. Specially, if the Stack is empty now, you need to print "`Invalid.`" (without quotes).

By the way, NAND is a basic binary operation:

- 0 nand 0 = 1
- 0 nand 1 = 1
- 1 nand 0 = 1
- 1 nand 1 = 0

Because Mr. Frog needs to do some tiny contributions now, you should help him finish this data structure: print the answer to each `QUERY`, or tell him that is invalid.

Input

The first line contains only one integers N ($2 \leq N \leq 200000$), indicating the number of operations.

In the following N lines, the i-th line contains one of these operations below:

- `PUSH x` (x must be 0 or 1)
- `POP`
- `REVERSE`
- `QUERY`

It is guaranteed that the current stack will not be empty while doing POP operation.

Output

Print several lines, i-th line contains an integer indicating the answer to the i-th `QUERY` operation. Specially, if the i-th `QUERY` is invalid, just print "`Invalid.`" (without quotes). Please see the sample for more details.

Examples

standard input	standard output
8 PUSH 1 QUERY PUSH 0 REVERSE QUERY POP POP QUERY	1 1 Invalid.
3 PUSH 0 REVERSE QUERY	0

Problem I. GCD

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3.5 seconds
Memory limit:	256 mebibytes

Mr. Frog likes generating numbers! He can generate many numbers from a sequence.

For a given sequence $a_1, a_2, \cdots, a_n$ Mr. Frog can choose two numbers l and r $(1 \leq l \leq r \leq n)$ and calculate the gcd between l-th and r-th number in this sequence $g = gcd(a_l, a_{l+1}, \cdots, a_r)$. As an expert in generating numbers, Mr. Frog wants to know how many distinct numbers can be generated by a sequence.

Mr. Frog likes challenges, so there may be many modifications in this sequence. In the i-th modification, Mr. Frog may change a_p to v_i. After each modification, you are asked to tell how many distinct numbers can be generated by this sequence immediately!

Input

The first line contains only one integer T, which indicates the number of test cases.

For each test case, the first line includes two numbers n, q $(1 \leq n, q \leq 5 \cdot 10^4)$, which indicate the length of sequence and the number of modifications.

The second line contains n numbers: $a_1, a_2, \cdots, a_n$.

Then q lines follow, each line contain two integers p_i, v_i $(1 \leq p_i \leq n, 1 \leq v_i \leq 10^6)$.

Test data guarantee that $1 <\leq a_i \leq 10^6$ all the time and the sum of all n and q in the input is less than or equal to $2 \cdot 10^5$.

Output

For each test case, first output one line "`Case #x:`", where x is the case number (starting from 1). Then print q lines, each line contain only one number, which is the answer to current sequence.

Example

standard input	standard output

2
3 2
1 2 3
1 3
2 3
3 2
3 3 3
1 1
2 2
	Case #1:
	3
	1
	Case #2:
	2
	3

Note

For case 1, after the first operation, 3,2,1 can be generated by the sequence 3, 2, 3. Whereas after the second operation, sequence 3, 3, 3 can generate only 3.

Problem J. Mission Possible

Input file:	`standard input`
Output file:	`standard output`
Time limit:	12 seconds
Memory limit:	256 mebibytes

Mr. Frog loves playing video games. However, it seems that he does not have enough gift in this field, so he often gets stuck in some certain levels. Now this annoying situation comes to him again.

In this mission, Mr. Frog is asked to pass a dangerous region which is currently under his enemies' control. Every second he is inside this region, his enemies would attack him with everything they have. Luckily, he is wearing a powersuit and still have time to enhance it.

Powersuit has three main attributes: health point, velocity and recover speed. In the beginning of this mission, the powersuit has H health points, and is able to move at no faster than V m/s. Once the health point of powersuit is less than 0, the powersuit would be totally destroyed and Mr. Frog would immediately be killed. At the end of every second that the powersuit still works, the suit would repair itself and recover R health points.

After precise calculation, Mr. Frog has found that, to go through this area, he needs to run at least D meters distance, and his enemies' attack would cause A health point loss per second. Note that attacks take place all the time while the recover only happens at the end of every second, so you can consider that, every whole second, the powersuit first loses A health points (precisely, if $0 \leq q \leq 1$, then for q seconds the powersuits loses $A \cdot q$ health points), and then recovers R health points at the end of it, if it still works at that moment.

Mr. Frog could enhance his powersuit at any time he wants during the mission. At the beginning all three attributes of his powersuit equals to 0. He needs to pay G_1 in order to increase H by 1, while G_2 to increase V by 1 and G_3 to increase R by 1. For any of these three values, he can only increase it by non-negative integer. For some well-known reasons, Mr. Frog does not want to finish the game too quickly, so you cannot increase the powersuit's speed to more than D m/s. Since you looks so clever, now Mr. Frog wants to know not only how he can finish this mission, but also the way to spend the least money, can you help him?

Input

The first line contains only one integer $1 \leq T \leq 25$, which indicates the number of test cases. For each test case, there are five integers D, A, G_1, G_2, G_3 ($1 \leq D, A \leq 5 \cdot 10^5$, $1 \leq G_1, G_2, G_3 \leq 200$). It is guaranteed that sum of all D in the input does not exceed $5 \cdot 10^6$ and that sum of all A in the input does not exceed $5 \cdot 10^6$.

Output

For each test case: output one line "`Case #x: Ans`', where x is the case number (starting from 1) and Ans is the minimum cost required to finish the mission.

Example

standard input	standard output

2
5 1 1 2 5
10 1 1 2 5
	Case #1: 7
	Case #2: 8

Problem K. Backpack on Tree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 mebibytes

There is a rooted tree with n nodes. For each node i, there is an item whose volume is c_i and value is v_i and if node i is not the root, it is guaranteed that $|subtreei| \leq \frac{2}{3}|subtree_{father_i}|$. Bacon wants to pick items in $subtree_s$ so that their total volume is exactly t. Help Bacon determine the maximal total value of items he can pick.

Input

The first line contains one integer T $(1 \leq T \leq 40)$ and there are exactly T test cases below.

For each test case, the first line contains one integer n $(1 \leq n \leq 2 \times 10^4)$.

The following $n - 1$ lines describe edges in the tree. Each line contains two integers a_i and b_i $(1 \leq a_i, b_i \leq n, a_i \neq b_i)$ describing an edge of the tree.

For the following n lines, the i-th line contains two integers c_i and $v_i(1 \leq c_i \leq 5, 1 \leq v_i \leq 10^9)$.

Next line contains one integer the number of queries Q and each of the following Q lines contains two integers s_i and $t_i(1 \leq s_i \leq n, 1 \leq t_i \leq 10^5)$ as a query.

Note that node 1 is the root of the tree.

There is no more than 4 test cases that n is greater than 10^4, and no more than 10 test cases that n is greater than 10^3. sum of all Q are not greater than 2×10^5.

Output

For each test case, first line contains "`Case #x:`, where x indicates the number of test cases (starting from 1).

Then print Q lines and the i-th line contains the answer of the i-th query. Print -1 for the query if there is no way to pick items in $subtree_s$ with total volume t.

Example

standard input

```
2
5
1 2
1 3
1 4
1 5
1 1
2 2
3 3
4 4
5 5
3
1 15
2 2
3 3
5
1 2
1 3
1 4
4 5
5 123
3 4543
4 21
1 1231
2 12
3
1 5
5 2
4 4
```

standard output

```
Case #1:
15
2
3
Case #2:
4555
12
-1
```