Problem A

Numbers

Input File: standard input
Output File: standard output
Time Limit: 0.3 seconds (C/C++)
Memory Limit: 256 megabytes

A palindrome is an integer which reads the same backward as forward. For example, numbers 142241 and 102201 are palindromes, but 1023401 and 10510 — no. You want to represent a number n as the sum of two palindromes. Find the number of ways to do it.

Input

There is only one line containing the integer n $(1 \leq n \leq 10^{18})$.

Output

Output one number — the number of ways to represent the number n as the sum of two palindromes.

Sample input	Sample output
156	4
9524	4
42657	6
5735832847451	28

Note

In the first test, the following pairs of numbers are suitable: $(5, 151)$, $(55, 101)$, $(101, 55)$, $(151, 5)$.
In the second test, the following pairs of numbers are suitable: $(515, 9009)$, $(636, 8888)$, $(8888, 636)$, $(9009, 515)$.
In the third test, the following pairs of numbers are suitable: $(33, 42624)$, $(333, 42324)$, $(4884, 37773)$, $(37773, 4884)$, $(42324, 333)$, $(42624, 33)$.

Problem B

Broken Watch

Input File: standard input
Output File: standard output
Time Limit: 0.2 seconds (C/C++)
Memory Limit: 256 megabytes

A UFO crash lands on Earth. The alien captain survives the crash relatively unscathed; however his trusty watch is damaged beyond repair.

The alien watch is very similar to a human watch: It is a disc $30mm$ in diameter and it has three hands of lengths A, B and C microns $(1000 \leq A, B, C \leq 15000)$. However, the aliens measure time differently: there are N $(2 \leq N < 2^{32})$ alien seconds to an alien minute. As such, there are N markings on the edge of the disc, rather than 60.

The watch's glass cover is broken and the hands have come loose: they can rotate freely and independently of one another. By pointing each of the hands at an arbitrary marking, their tips can be made to form an imaginary triangle (as long as they are non-collinear).

Having nothing left to do but wait for help to arrive, the alien ponders the following question: What is the number M of distinct triangles that contain the watch's center? (Triangles where the center lies on one of the edges of the triangle count as well.)

Input

The input consists of A, B, C, and N on a single line, separated by whitespace.

Output

The output is M modulo 2^{64}.

Sample input	Sample output
`15000 15000 15000 2`	`0`
`5000 10000 15000 3`	`6`
`15000 15000 15000 3`	`1`
`15000 15000 15000 4`	`4`
`15000 15000 15000 5`	`5`
`15000 15000 15000 6`	`14`

Problem C

Tree

Input File: standard input
Output File: standard output
Time Limit: 0.1 seconds (C/C++)
Memory Limit: 256 megabytes

You are given a tree of n vertices, each with a unique number from 1 to n. A vertex has a color, black or white. Choose exactly m black vertices so that the length of the longest path between any of them is minimal.

Input

The first line contains two integers n and m $(1 \leq m \leq n \leq 100)$ — the number of vertices and the number of black vertices you have to choose.

The fourth line contains n integers $p_1, p_2, \ldots, p_n$ $(0 \leq p_i \leq 1)$. If the $p_i = 1$, then the i-th vertex is black; otherwise, it is white. It is guaranteed that the number of black vertices is at least m.

Each of the next $n-1$ lines contains two integers v_i and u_i $(1 \leq v_i, u_i \leq n)$ meaning that there is an edge between v_i and u_i.

It is guaranteed that the input graph is a tree.

Output

Print a single integer — the answer to the task.

Sample input	Sample output

6 3
1 1 0 1 1 1
1 2
1 3
1 4
3 5
3 6
	2

9 4
1 0 1 0 1 0 0 1 1
1 2
2 4
2 3
4 5
1 6
6 7
6 8
7 9
	5

Note

In the first example, the only option is to choose 1, 2, and 4. The maximum distance will be 2.

In the second example, you can choose 1, 3, 8, and 9. The maximum distance will be between 3 and 9.

Problem D

Space Station

Input File: standard input
Output File: standard output
Time Limit: 0.4 seconds (C/C++)
Memory Limit: 256 megabytes

Jones has achieved one of his greatest dreams: he has been accepted to join the next mission to the International Space Station (ISS). He has already been assigned his first task: to check the integrity of all the electronic components on board of the space station.

The ISS is composed of N modules, numbered from 1 to N. Jones has discovered that, for efficiency reasons, the station is designed such that there exists exactly one simple path between any two different modules. In the event of a solar flare, the bi-directional segments that link two different modules are particularly vulnerable to radiation. Inspecting a segment i takes an integral amount of time, C_i. Naturally, Jones is trying to find the fastest visitation path that starts in module 1, visits each segment at least once and returns to module 1.

In addition to using the direct segments between two modules, Jones may put on a spacesuit and seek a direct path, outside the station, between any two modules. However, he can only perform such a feat M times. Jones assumes it takes a fixed amount of time, K, to put on the spacesuit and use it to jump from one module to any other module.

Input

The first line contains the number T, representing the number of tests. The first line that describes a test contains the three variables N M K. $(1 \leq N, M \leq 1000)$ The following N–1 lines are of the form A B C $(1 \leq A \leq B \leq N; 0 \leq C, K \leq 10^6)$, meaning there is a direct segment from A to B, taking C time units.

Output

On each line i, a single number representing the answer for test i.

Sample input	Sample output

```
2
5 2 4
1 2 2
2 3 2
1 4 2
4 5 2
7 2 0
1 2 1
1 3 5
2 4 10
2 5 1
5 6 10
5 7 5
```
	```
	12
	33
	```

Problem E

Fishermen

Input File: standard input
Output File: standard output
Time Limit: 0.5 seconds (C/C++)
Memory Limit: 256 megabytes

The ocean can be represented as the first quarter of the Cartesian plane. There are n fish in the ocean. Each fish has its own coordinates. There may be several fish at one point.

There are also m fishermen. Each fisherman has its own x-coordinate. The y-coordinate of each fisherman is equal to 0.

Each fisherman has a fishing rod of length l. Therefore, he can catch a fish at a distance less than or equal to l. The distance between a fisherman in position x and a fish in position (a, b) is $|a - x| + b$.

Find for each fisherman how many fish he can catch.

Input

The first line contains three integers n, m, and l $(1 \leq n, m \leq 2 \cdot 10^5, 1 \leq l \leq 10^9)$ — the number of fish, the number of fishermen, and the length of the fishing rod, respectively.

Each of the next n lines contains two integers x_i and y_i $(1 \leq x_i, y_i, \leq 10^9)$ — the fish coordinates.

Next line contains m integers a_i $(1 \leq a_i \leq 10^9)$ — the fishermen coordinates.

Output

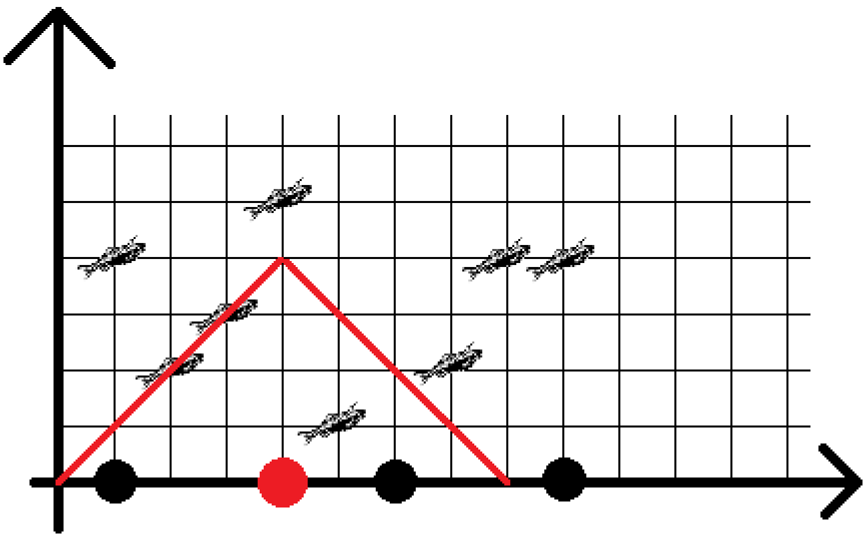
For each fisherman, output the number of fish that he can catch, on a separate line.

Sample input	Sample output

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

Note

The picture illustrates for the above example the area on which the third fisherman can catch fish.



Problem F

Min Max Convert

Input File: standard input
Output File: standard output
Time Limit: 0.4 seconds (C/C++)
Memory Limit: 256 megabytes

Let A be a sequence of N elements. You can perform two types of operations on this sequence:

1. Select an interval of positions $[a, b] (1 \leq a \leq b \leq N)$. Let x be the maximum value on this interval. Replace all the elements in the interval with x.
2. Select an interval of positions $[a, b] (1 \leq a \leq b \leq N)$. Let x be the minimum value on this interval. Replace all the elements in the interval with x.

Determine a sequence of operations such that sequence A becomes another given sequence B (of also N elements). The number of operations must be less or equal than $2 * N$.

Input

The first line of the input contains a single number N. The second line contains A, a sequence of N elements. The third line contains B, another sequence of N elements.

Output

If there is **no solution** such that sequence A becomes B, print **–1**. Otherwise, print on the first line a single number x, the minimum number of operations needed to transform sequence A in B. Each of the next x lines will contain a character (the type of the operation: m if the operation use the minimum and M for the maximum) and an interval (a, b), describing the operations needed for the process. If there are multiple solutions, print any of them.

Constraints

- $1 \leq N \leq 100.000$
- All values in A and B are integer numbers from $[1, N]$

Sample input	Sample output

5
1 5 5 3 4
1 1 4 4 4
	3
	m 1 2
	M 4 5
	m 3 5

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

Problem G

Matrix Queries

Input File: standard input
Output File: standard output
Time Limit: 1.5 seconds (C/C++)
Memory Limit: 256 megabytes

You are given a matrix of size $2^n \times 2^n$, initially painted in white color. The color of a cell can be either black or white. Let's define the *price* of a matrix as:

1. If a matrix is painted with only one color, the price will be 1 coin;
2. Otherwise, you should split the matrix into 4 size-equal matrices, and the price of a matrix will be the sum of submatrices prices plus 1 coin.

You are given q queries. Each query gives you the number of row/column x, and you have to change the color of all cells in this row/column (i.e., if a cell is white, it will be black; if a cell is black, it will be white) and find the *price* of the new matrix.

Input

The first line contains two integers n and q ($0 \le n \le 20$, $1 \le q \le 10^6$) where n means that the size of the matrix is $2^n \times 2^n$ and q means that there are going to be q queries.

Each of the next q lines contains two integers t and x ($0 \le t \le 1$, $1 \le x \le 2^n$). If $t = 0$, then the x-th row will be changed; otherwise, the x-th column.

Output

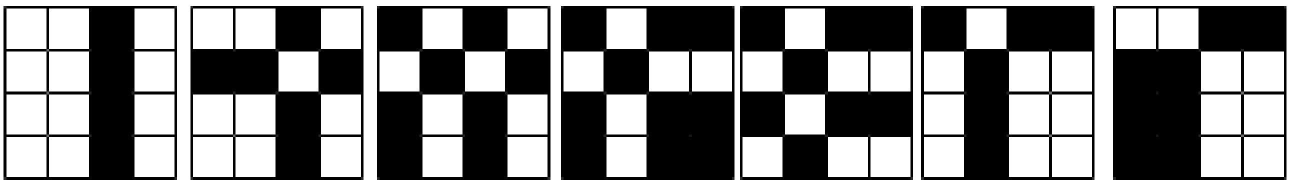
For each query, print a matrix price.

Sample input	Sample output

2 7
1 3
0 2
1 1
1 4
0 4
0 3
1 1
	13
	17
	21
	17
	21
	17
	13

Note

In the sample, after each query the matrix will be as follows:



Problem H

Modern Djinn

Input File: standard input
Output File: standard output
Time Limit: 0.4 seconds (C/C++)
Memory Limit: 256 megabytes

You are certainly no leader, but a president. Lucky for you, you own a modern djinn, a spiritual creature that makes your wishes come true. One such wish is to pretend to have democracy in your society.

Society is simple. It consists of N people numbered from 1 to N, some of which are "happy", and some are regular ("unhappy"). Human nature is very tricky. People are happy only when others aren't. You have M wishes, numbered from 1 to N, $X \rightarrow Y$ denoting that person X wants person Y to be unhappy. A person X is happy if and only if at least one of his wishes is satisfied. Democracy is also not that complex. Some may say that you need to have at least half of the people happy (or half of the wishes solved) in order to have democracy, but that it is not true at all. As I said earlier, you are a good president, not a good leader. You have access to media so you define democracy. Therefore, out of all M wishes, you decided to make at least $\lfloor M/4 \rfloor + 1$ wishes come true.

All that is left is to decide which wishes you want to grant, the djinn will handle the rest.

Input

The input will contain multiple test cases. The first number T, will represent the number of test cases. The next lines will describe the test cases in order.

The first line of a test case contains 2 positive integers N and M, the number of people and the number of wishes. Next M lines of a test case contain pairs of 2 integers X Y separated by a space denoting a wish of the form: X wants Y to be unhappy.

Output

For each test case on the first line output a number K, representing the number of wishes that were granted. On the second line output K distinct integers separated by spaces representing the indices of the wishes that were granted. These can be printed in any order.

Constraints

- $1 \leq T \leq 10.000$
- $2 \leq N \leq 100.000$
- $1 \leq M \leq 200.000$
- There is no wish of the form X wants X to be unhappy.
- There can be multiple wishes of the form X wants Y to be happy.
- For each test case it is guaranteed that a solution exists.
- Any correct solution is accepted

Sample input	Sample output

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

For the first test case we can grant at most one wish, outputting any of them is correct.

For the second test case another solution is to grant wishes 1 3 and 4. The minimum required number of wishes that needed to be granted was 2 though.

Problem I

Inversion

Input File: standard input
Output File: standard output
Time Limit: 0.1 seconds (C/C++)
Memory Limit: 256 megabytes

A sequence $p_1, p_2, \ldots, p_n$ is called a permutation of n numbers $1, 2, \ldots, n$ if any number in the range $[1,\ n]$ occurs exactly once in it. The pair (i, j) of integers in the range 1 to n is called an inversion if $i < j$ and $p_i > p_j$.

Let's call an inversion graph a graph which has exactly n vertices and there is and an edge between the pair (i, j) if and only if this pair is an inversion.

A set s of vertices of a graph is called independent if no two vertices from this set have an edge between them. A set t of vertices of a graph is called dominant if every vertice which does not belong to the set has an edge between at least one vertice which belongs to it. A set g of vertices of a graph is called independent-dominant if it is both dominant and independent.

You have an inversion graph of a particular permutation $1, 2, \ldots n$ which is defined with pairs of vertices (a_i, b_i) which have an edge between them. Find the number of independent-dominant sets of the graph.

It is guaranteed that the answer does not exceed 10^{18}.

Input

The first line contains two integers n and m ($1 \leq n \leq 100$, $0 \leq m \leq \frac{n \times (n-1)}{2}$) — the number of vertices of the graph and the number of edges in the graph.

Each of the next m lines contains two integers u_i and v_i ($1 \leq u_i, v_i \leq n$), which means that there is an edge between u_i and v_i.

It is guaranteed that there exists a permutation that gives this graph.

Output

Print out the number of independent-dominant sets of vertices of the graph.

It is guaranteed that the answer does not exceed 10^{18}.

Sample input	Sample output
4 2	2
2 3	
2 4	
5 7	3
2 5	
1 5	
3 5	
2 3	
4 1	
4 3	
4 2	
7 7	6
5 6	
2 3	
6 7	
2 7	
3 1	
7 5	
7 4	
5 6	5
1 3	
4 5	
1 4	
2 3	
1 2	
1 5	

Note

The first sample is graph for permutation $[1, 4, 2, 3]$. We can select two sets of nodes: $(1, 3, 4)$ or $(1, 2)$.
The second sample is graph for permutation $[3, 5, 4, 1, 2]$. We can select three sets of nodes: $(1, 2), (1, 3), (4, 5)$.
The third sample is a graph for permutation $[2, 4, 1, 5, 7, 6, 3]$.
The fourth sample is a graph for permutation $[5, 2, 1, 4, 3]$.

Problem J

Rabbit vs Turtle

Input File: standard input
Output File: standard output
Time Limit: 0.4 seconds (C/C++)
Memory Limit: 256 megabytes

A rabbit and a turtle decided to race each other. Since the turtle is from Craiova and the rabbit is from Ardeal, the turtle is obviously faster than the rabbit. Our goal is to help the rabbit win the race.

The race is held on a graph with N nodes and M vertices. The race starts at node 1 and ends at node N. Both the rabbit and the turtle decided beforehand to select the path which they are going to use (each one of them with his own path). Therefore, they know the graph, the paths and also the time it takes for each one of them to cross each edge of the graph.

The turtle may be faster than the rabbit, but it's still a turtle (let's call it George). George selects some nodes on his path where he decides to sleep for a while. If at any moment in time he realizes that the rabbit cheats, George will stop sleeping and will go to the finish without any further rest.

The rabbit (let's call it Stan) has only one advantage. Stan has a great-great-great-. . . ...-great grandmother who is a fox, therefore he has a sly side. Stan does not intend to keep his promise about the path (but George does). At some point, his plan is to change the path he is initially assigned and take the shortest path to node N. The only problem is that he has to be smart about it. The moment George discovers that Stan is cheating, he will stop his sleeping activities, which is not good.

Stan can only change his path when he is in a node (obviously not when he is on an edge). He does not know George's sleeping schedule, but you do!!! Compute the number of moments when Stan can change his path such that he will win the race. The moment Stan cheats, George will discover immediately as long as he is not sleeping. If he is sleeping during that time, he will find out upon waking up.

Input

The first line contains two numbers N and M. The next M lines will describe the graph by (A, B, T, R): there is an edge from node A to node B. The turtle will cross the edge in T units of time, while the rabbit will do it in R units of time. The i-th such edge is considered to be the edge with index i.

The next line contains a number P_T, the number of nodes in George's path. The next P_T lines will describe the path by (edge_index, sleep): the index of the next edge George is going to use and the number of time units he is going to sleep after using that edge. The sleeping value for the last edge is irrelevant since the turtle will already reach the finish.

The next line contains a number P_R, the number of edges in Stan's initial path. The last line will contain P_R values, the indices of edges Stan is going to use.

Output

The output contains on the first line one value: x, the number of moments (nodes on the path) when Stan can cheat. The second line contains x numbers in **ascending order**, the indices of the nodes where Stan can cheat.

Constraints

- $2 \leq N \leq 100.000$
- $1 \leq P\_R, P\_T < 100.000$
- $1 \leq M \leq 200.000$
- $1 \leq T, R \leq 1.000.000.000$

- $0 \leq sleep \leq 1.000.000.000$
- The edges from the input that describe the paths are given in the correct order.
- The nodes in the turtle's path do not repeat.
- The nodes in the rabbit's path do not repeat.
- If the rabbit and the turtle arrive at the finish in the same time, the rabbit wins.
- If the rabbit cheats in the exact moment when the turtle goes to sleep, the turtle will first fall asleep and only after waking up he will realize about the cheating.
- The rabbit is considered to cheat ONLY if he changes the direction of the path and the new path that he will take is strictly faster than the original one (otherwise, there is no point to cheat)
- The cheating path may have common nodes with the original path. The only restriction is that in the exact moment of cheating, the rabbit has to take a different node. He may even go back to the previous node if he has an edge that allows that.

Sample input	Sample output

8 12
1 2 2 10
2 3 1 10
3 8 2 10
1 4 10 3
4 5 10 2
5 6 10 4
6 8 10 2
1 7 10 5
4 7 10 2
5 7 10 2
6 7 10 1
7 8 10 1
3
1 3
2 2
3 0
4
4 5 6 7
	2
	4 5

6 6
1 4 1 3
4 6 1 1
4 2 1 6
2 6 6 6
3 4 2 3
1 3 4 5
2
1 2
2 0
4
6 5 3 4
	0

Problem K

Points and Rectangles

Input File: standard input
Output File: standard output
Time Limit: 2 seconds (C/C++)
Memory Limit: 256 megabytes

You have an empty infinite two-dimensional plane and q queries. There are two types of queries:

- $\langle\langle\ 1\ \ x\ \ y\rangle\rangle$ — add a point with the coordinates (x, y) to the plane.
- $\langle\langle 2\ \ x_1\ \ y_1\ \ x_2\ \ y_2\rangle\rangle$ — add a rectangle whose lower left corner has the coordinates (x_1, y_1) and the upper right — (x_2, y_2). The area of this rectangle can be zero and a rectangle can degenerate into a point.

Rectangles and points may overlap, that is, there is not guarantee that the figures are distinct.
In addition, to fulfill these queries, after each of them, you need to print the number of pairs of rectangles and points, in which the point lies on the border or inside the rectangle.

Input

The first line contains one integer q $(1 \le q \le 10^5)$ — the number of queries.
Each of the following q lines contains one query:

- $\langle\langle 1\ \ x\ \ y\rangle\rangle$ $(1 \le x, y \le 10^9)$ — add a point with the coordinates (x, y) to this plane.
- $\langle\langle 2\ \ x_1\ \ y_1\ \ x_2\ \ y_2\rangle\rangle$ $(1 \le x_1 \le x_2 \le 10^9,\ 1 \le y_1 \le y_2 \le 10^9)$ — add a rectangle whose left lower corner has the coordinates (x_1, y_1), and the upper right — (x_2, y_2).

Output

You need to print out q lines, the i-th line must contain one integer number — the number of pairs of rectangles and points, in which the point lies on the side or inside the rectangle.

Sample input	Sample output

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

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

7
1 5 5
1 5 5
1 5 5
2 2 2 9 9
2 1 1 5 5
2 1 1 2 2
1 2 2
	0
	0
	0
	3
	6
	6
	9

Note

Explanation of the first example:

After the first query, we have one point with the coordinates $(2, 3)$, and there are no rectangles at all, so there are no pairs of points and rectangles at all.

After the second query, we still do not have rectangles, so there are no pairs at all.

Still no rectangles after the third query.

In the fourth query, we added a rectangle with the coordinates of the left lower point $(1, 1)$, and the coordinates of the upper right corner are $(5, 5)$. All three previously added points lie inside this rectangle, so we have three pairs.

After the fifth query, we have four pairs: the points added during the first three queries lie inside the rectangle that was added in the fourth query (the first three pairs) and the pair where the point added in the second query lies inside the rectangle, which was added in the fifth query.