Greetings From Globussoft

- Given below are 5 Programming questions, you have to solve any 3 out of 5 questions.

- These 5 questions you can attempt in any technology like C/C++, java, .Net, PHP

- To solve these 3 questions you've max. 3 hours.

- While Solving these questions you are not allowed to use any **Search Engine** like Google, Yahoo, Bing …

All the best for your test

Globussoft

QUESTION - 1

You are the owner of a railway system between n cities, numbered by integers from 1 to n. Each train travels from the start station to the end station according to a very specific timetable (always on time), not stopping anywhere between. On each station a departure timetable is available. Unfortunately each timetable contains only direct connections. A passenger that wants to travel from city p to city q is not limited to direct connections however - he or she can change trains. Each change takes zero time, but a passenger cannot change from one train to the other if it departs before the first one arrives. People would like to have a timetable of all optimal connections. A connection departing from city p at A o'clock and arriving in city q at B o'clock is called optimal if there is no connection that begins in p not sooner than at A, ends in q not later than at B, and has strictly shorter travel time than the considered connection. We are only interested in connections that can be completed during same day.

Task

Write a program that:

- reads the number n and departure timetable for each of n cities from the standard input,
- creates a timetable of optimal connections from city 1 to city n,
- writes the answer to the standard output.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first line of the input contains an integer n (2 <= n <= 100000). The following lines contain n timetables for cities 1, 2, ..., n respectively.

The first line of the timetable description contains only one integer m. Each of the following m lines corresponds to one position in the timetable and contains: departure time A, arrival time B (A < B) and destination city number t (1 <= t <= n) separated by single spaces. Departure time A and arrival time B are written in format hh : mm, where hh are two digits representing full hours (00 <= hh <= 23) and mm are two digits representing minutes (00 <= mm <= 59). Positions in the timetable are given in non-decreasing order according to the departure times. The number of all positions in all timetables does not exceed 1000000.

Output

For each test case the first line of the output contains an integer r - the number of positions in the timetable being the solution. Each of the following r lines contains a departure time A and an arrival time B separated by single space. The time format should be like in the input and positions in the timetable should be ordered increasingly according to the departure times. If there is more then one optimal connection with the same departure and arrival time, your program should output just one.

Example

Sample input:

```
1
3
3
09:00 15:00 3
10:00 12:00 2
11:00 20:00 3
2
11:30 13:00 3
12:30 14:00 3
0
```

Sample output:

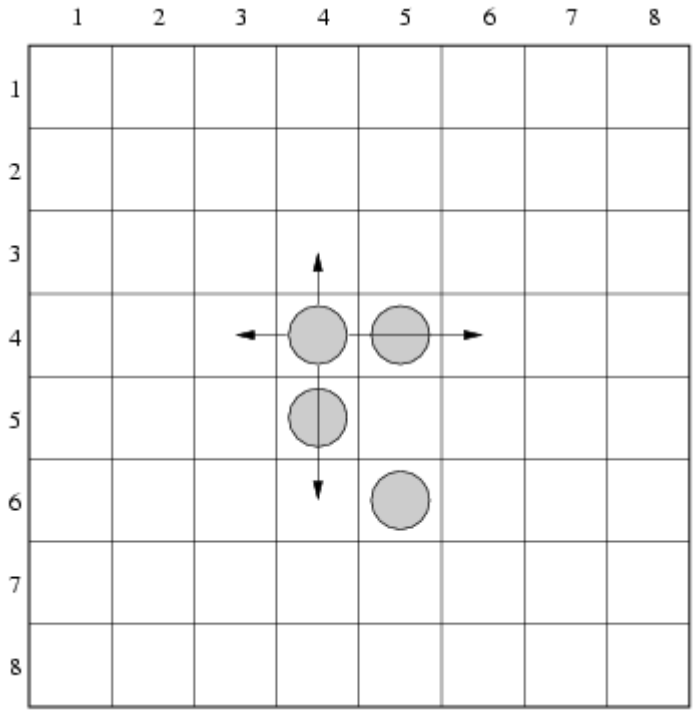
```
2
10:00 14:00
11:00 20:00
```

QUESTION – 2

Solitaire is a game played on an 8x8 chessboard. The rows and columns of the chessboard are numbered from 1 to 8, from the top to the bottom and from left to right respectively.

There are four identical pieces on the board. In one move it is allowed to:

- move a piece to an empty neighboring field (up, down, left or right),
- jump over one neighboring piece to an empty field (up, down, left or right).



There are 4 moves allowed for each piece in the configuration shown above. As an example let's consider a piece placed in the row 4, column 4. It can be moved one row up, two rows down, one column left or two columns right.

Task

Write a program that:

- reads two chessboard configurations from the standard input,
- verifies whether the second one is reachable from the first one in at most 8 moves,
- writes the result to the standard output.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case, each of two input lines contains 8 integers a_1, a_2, ..., a_8 separated by single spaces and describes one configuration of pieces on the chessboard. Integers a_{2j-1} and a_{2j} ($1 <= j <= 4$) describe the position of one piece - the row number and the column number respectively.

Output

For each test case the output should contain one word for each test case - NO' otherwise.

Example

Sample input:

```
1
4 4 4 5 5 4 6 5
2 4 3 3 3 6 4 6
```

Sample output:

```
YES
```

QUESTION – 3

The Kingdom of Byteland decided to develop a large computer network of servers offering various services.

The network is built of n servers connected by bidirectional wires. Two servers can be directly connected by at most one wire. Each server can be directly connected to at most 10 other servers and every two servers are connected with some path in the network. Each wire has a fixed positive data transmission time measured in milliseconds. The distance (in milliseconds) D(V, W) between two servers V and W is defined as the length of the shortest (transmission time-wise) path connecting V and W in the network. For convenience we let D(V, V) = 0 for all V.

Some servers offer more services than others. Therefore each server V is marked with a natural number r(V), called a rank. The bigger the rank the more powerful a server is.

At each server, data about nearby servers should be stored. However, not all servers are interesting. The data about distant servers with low ranks do not have to be stored. More specifically, a server W is interesting for a server V if for every server U such that D(V, U) <= D(V, W) we have r(U) <= r(W).

For example, all servers of the maximal rank are interesting to all servers. If a server V has the maximal rank, then exactly the servers of the maximal rank are interesting for V . Let B(V) denote the set of servers interesting for a server V.

We want to compute the total amount of data about servers that need to be stored in the network being the total sum of sizes of all sets B(V). The Kingdom of Byteland wanted the data to be quite small so it built the network in such a way that this sum does not exceed 30*n.

Task

Write a program that:

- reads the description of a server network from the standard input,
- computes the total amount of data about servers that need to be stored in the network,
- writes the result to the standard output.

Input

The input begins with the integer z, the number of test cases. Then z test cases follow.

For each test case, in the first line there are two natural numbers n, m, where n is the number of servers in the network ($1 <= n <= 30000$) and m is the number of wires ($1 <= m <= 5n$). The numbers are separated by single space.

In the next n lines the ranks of the servers are given. Line i contains one integer r_i ($1 <= r_i <= 10$) - the rank of i-th server.

In the following m lines the wires are described. Each wire is described by three numbers a, b, t ($1 <= t <= 1000$, $1 <= a, b <= n$, $a <> b$), where a and b are numbers of the servers connected by the wire and t is the transmission time of the wire in milliseconds.

Output

For each test case the output consists of a single integer equal to the total amount of data about servers that need to be stored in the network.

Example

Sample input:

```
1
4 3
2
3
1
1
1 4 30
2 3 20
3 4 20
```

Sample output:

```
9
```

(because B(1) = {1, 2}, B(2) = {2}, B(3) = {2, 3}, B(4) = {1, 2, 3, 4})

QUESTION – 4

We want to find out how much related are the members of a family of monsters. Each monster has the same number of genes but the genes themselves may differ from monster to monster. It would be nice to know how many genes any two given monsters have in common. This is impossible, however, since the number of genes is very large. Still, we do know the family tree

(well, not actually a tree, but you cannot really blame them, these are monsters, right?) and we do know how the genes are inherited so we can estimate the number of common genes quite well.

The inheritance rule is very simple: if a monster C is a child of monsters A and B then each gene of C is identical to the corresponding gene of either A or B, each with probability 50%. Every gene of every monster is inherited independently.

Let us define the degree of relationship of monsters X and Y as the expected number of common genes. For example consider a family consisting of two completely unrelated (i.e. having no common genes) monsters A and B and their two children C and D. How much are C and D related? Well, each of C's genes comes either from A or from B, both with probability 50%. The same is true for D. Thus, the probability of a given gene of C being the same as the corresponding gene of D is 50%. Therefore the degree of relationship of C and D (the expected number of common genes) is equal to 50% of all the genes. Note that the answer would be different if A and B were related. For if A and B had common genes, these would be necessarily inherited by both C and D.

Your task is to write a program that, given a family graph and a list of pairs of monsters, computes the degree of relationship for each of these pairs.

Task

Write a program that:

- reads the description of a family and a list of pairs of its members from the standard input,
- computes the degree of relationship (in percentages) for each pair on the list,
- writes the result to the standard output.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first line of the input contains two integers n and k separated by a single space. Integer n ($2 <= n <= 300$) is the number of members in a family. Family members are numbered arbitrarily from 1 to n. Integer k ($0 <= k <= n - 2$) is the number of monsters that do have parents (all the other monsters were created by gods and are completely unrelated to each other).

Each of the next k lines contains three different integers a, b, c separated by single spaces. The triple a, b, c means that the monster a is a child of monsters b and c.

The next input line contains an integer m ($1 <= m <= n^2$) - the number of pairs of monsters on the list. Each of the next m lines contains two integers separated by a single space - these are the numbers of two monsters.

You may assume that no monster is its own ancestor. You should not make any additional assumptions on the input data. In particular, you should not assume that there exists any valid sex assignment.

Output

For each test case the output consists of m lines. The i-th line corresponds to the i-th pair on the list and should contain single number followed by the percentage sign. The number should be the exact degree of relationship (in percentages) of the monsters in the i-th pair. Unsignificant zeroes are not allowed in the output (please note however that there must be at least one digit before the period sign so for example the leading zero in number 0.1 is significant and you cannot print it as .1). Confront the example output for the details of the output format.

Example

Sample input:

```
1
7 4
4 1 2
5 2 3
6 4 5
7 5 6
4
1 2
2 6
7 5
3 3
```

Sample output:

```
0%
50%
81.25%
100%
```

QUESTION – 5

There are two separate, *n*-element sets of points of a two dimensional map: **R** and **W**. None triple of points from the set **R**∪**W** is collinear. Rockets earth-to-earth are located on points from the set **R**. Enemy objects, which should be destroyed, are located on points from the set **W**. The rockets may fly only in the straight line and their trajectories cannot intersect. We are about to find for each rocket a target to destroy.

Task

Write a program which:

- reads from the standard input coordinates of the points from the sets **R** and **W**,
- finds the set of *n* pairwise not-intersecting segments, so that one end of each segment belongs to the set **R**, while the other belongs to the set **W**,
- writes the result into the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. In the first line of each test case there is written one integer *n*, *1<=n<=10000*, equal to the number of elements of the sets **R** and **W**.

In each of the following *2n* lines of the input one pair of integer numbers from the interval [*-10000, 10000*] is written. Numbers in each pair are separated by a single space. They are coordinates of the point on a map (first coordinate *x*, then *y*). The first *n* lines comprise coordinates of the points from the set **R**, the last *n* lines comprise the points from the set **W**. In the (*i+1*)-th line there are coordinates of the point r_i, in the (*i+n+1*)-th line there are coordinates of the point w_i, *1<= i<= n*.

Output

The output for each test case should consist of *n* lines. In the *i*-th line there should be one integer *k(i)*, such that the segment $r_i\ w_{k(i)}$ belongs to the set of segments which your program found. (This means that the rocket from the point r_i destroys an object in the point $w_{k(i)}$).

Example

Sample input:

```
1

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

Sample output:

```
2
1
4
3
```