Winter Camp

This year Ivancho and his class desided to spent the winter vacation together and go to a mountain resort. To make their stay more fun, the students desided to split into several teams and make a competition. As the best student of the class, Ivancho earned the right to first choose his team members.

Ivancho chose a small field where his team will stay. There are **V** bungalows and **R** paths on the field. Every path connects two bungalows, and the paths are chosen such that you can travel between every two bungalows by just following the paths. Ivancho will have to chose in which bungalow every member of his team will stay. He is not limited in the amount of people he can have in his team, nor is he required to fill all bungalows.

During the time they stay at the camp, a heavy snowfall is expected. Becouse of this, the paths between the bungalows will have to be shoveled of snow every morning. Ivancho knows that there should be enought clean paths at all times, so that one can travel between every two bungalows where students are staying following the paths. But he also knows that his classmates are lasy and the **i**-th student will shovel no more than **Di** paths starting from his bungalow.

The contest will contain several games, both for physical strength and logical thinking. Ivancho knows the abbilities of his classmates in the different games and has made an assesment of their overall pereformance in the form of the number **Wi**.

Ivancho calls neighbours two students, which are staying in bungalows connected directly by a cleaned path. He wants all neighbours to be friends. Not all students are friends with each other, but if he puts friends in neighbouring bungalows that will improve his team's spirit. And everyone knows that a stong team spirit is the key to success. That is why he desided for every two friends **i** and **j** to rate with **Cij** how much would they boost the team spirit if they are put in neighbouring bungalows. (**Cij** = **Cji**).

To find out how good is a given team compossition, Ivancho uses the following formula:
F = (the sum of **Cij** for all neighbours) + (the sum of the products (**Wi** * (number of neighbours of **i**), for every student **i** from his team).

Ivancho wants to choose a team with as high result as possible, but he can't do that on his own. That's why he turns towards you, good programers. Help him by writing a program **camp**, which by a given information about Ivancho's classmates and the bungalows on the field, find a good composition for Ivancho's team.

Input:
The first row in the input file (**camp.in**) will contain two integers - **N** and **M**, where **N** is the number of students, and **M** is the number of pairs of friends.

The next **M** rows will contain tree integers each - **i**, **j** and **Cij** - the two friends **i** and **j** and their respectible grade **Cij**.

$i \neq j$;
$0 \leq i, j < N$;
no pair of students (i, j) will be repeated;
the input will meet the criteria that you can not create two groups of students such that no one from the first group is friend with no one from the second.

The next row will contain **N** integers - the corresponding **Wi** for each student.
The next row will contain **N** integers - the corresponding **Di** for each student.

The next row will contain two integers - **V** and **R** - respectively the number of bungalows and the number of paths between them.

The next **R** rows will contain two integers each - **p** and **q**, which show that a path exists between bungalows **p** and **q**.

$p \neq q$;

$0 \leq p, q < V$;

Between every two bungalows exitst no more than one path.

Output:

The first row in the output file (**camp.out**) should contain a single integer **K** - the number of students that will stay on the lawn.

The next **K** rows should contain two integers each - **Xi** and **Yi** - the student with number **Xi** is a member of Ivancho's team an will stay in bungalow **Yi**.

$0 \leq Xi < N$;

$0 \leq Yi < V$;

The same pair (Xi, Yi) should not be repeated in the output.

The next row should contain a single integer **T** - the number of path that must be kept clean.

The next T rows should contain two integers each - **Pi** and **Qi** - the numbers of the students, that should shovel the path between their bungalows.

$0 \leq Pi, Qi < N$;

A path must exist between Pi's and Qi's bungalows.

The same path should not be repeated in the output.

Grading:

If the output doesn't meet any of the conditions the program recieves 0 points for the test case.

Else we define **Fyour** = **F** by the formula given earlier. Let **Fmax** is the biggest score, recieved by a contestant's solution for this test. Then your program recieves $\left(\frac{Fyour+1}{Fmax+1}\right)^2 \times 100$ percent of the points for the given test case.

Limits:

In 15% of test cases N <= 1000
In another 30% of test cases N <= 5000
In the remaining tests N <= 10 000

In 10% of testcases M <= 10 000
In another 30% of testcases M <= 50 000.
In the remaining tests M <= 100 000

In 15% of test cases V <= 1000.
In another 20% of test cases V <= 2000.
In another 35% of test cases V <= 5000.
In the remaining tests V <= 10 000;

In 30% of testcases R <= 10 000.
In another 30% of testcases R <= 20 000.
In another 25% of test cases R <= 50 000.
In the remaining tests R <= 100 000;

0 <= Cij <= 1000
0 <= Wi <= 100

Let G1 is the graph, build with the students as vertices and the friendship relationship between them as edges.
Let G2 is the graph, build with the bungalows as vertices and the paths betwwen them as edges.

In 15% of test cases, G1 and G2 are trees.
In another 25% of test cases, only G1 is a tree.

In all test cases G1 and G2 are generated in such a way, that between 20% and

60% of their structure is identical. The remaining part of the graph is generated randomly.

Time limit: 5 sec.

Example 1:

camp.in	camp.out
6 10	6
0 1 2	2 0
0 2 4	5 1
0 5 7	3 2
1 3 8	1 3
1 5 11	4 4
2 4 12	0 5
2 5 13	6
3 4 16	0 2
3 5 17	2 4
4 5 19	4 3
10 5 2 1 3 0	3 1
3 3 3 3 3 3	1 5
6 6	5 0
0 4	
0 5	
1 3	
1 5	
2 3	
2 46	
2 0	
5 1	
3 2	
1 3	
4 4	
0 5	
6	
0 2	
2 4	
4 3	
3 1	
1 5	
5 0	
Fyour = 100	

Explanation:
F = (4 + 12 + 16 + 8 + 11 + 7) + (2*3 + 2*0 + 2*2 + 2*5 + 2*1 + 2*10) = 100
It should be noted that this is not the optimal solution.

Example 2:

camp.in	camp.out
6 10	5
0 1 2	1 0
0 2 4	0 1
0 5 7	2 2
1 3 8	4 3
1 5 11	5 5
2 4 12	4
2 5 13	1 0
3 4 16	0 2
3 5 17	0 5
4 5 19	5 4
10 5 2 1 3 0	

```
3 1 1 0 1 2
6 7
0 1
0 5
1 2
1 5
2 3
3 4
3 5
```

```
F = 72
```