12536   Optimal Space Way

It is 2180 AD, the human race has just started to abandon the earth and spread the civilization in space. ACM (Association for Cosmic Machinism) is in charge of this whole relocation process and this process is funded mostly by IBM (Inter-planet Ballistic Machines). Thousands of space cities are built in the outer space and to maintain universal business hour (Sunrises and sets at the same time in all space cities) they are built on the same imaginary plane. So the location of the cities can be expressed as two-dimensional Cartesian coordinates.

The cities are located far away from one another and the only way of going from one city to another is by using shuttle rockets. But from previous experience ACM knows that traveling by roads is a lot cheaper than traveling by rockets. So they have decided to build some roads. But building roads between every pair of cities is very expensive. So they are planning to build a super space-way (SSW) that will be a straight road. When someone needs to travel from a city c1 to city c2, he first travels from c1 to the nearest point of SSW with a rocket, and moves along the SSW to the point, which is nearest to c2. Then from there he goes to c2 using another rocket. The cost of moving along the road is proportional to the distance but cost of flying on rockets is proportional to the square of the distance. So you have to build the SSW in such a position so that total cost of traveling in a calendar year by rockets is minimized. You can assume that:

1. The cities are so small compared to the distance between them that they can be considered as points on a two dimensional Cartesian coordinate system.

2. The SSW is so narrow compared to the distance between the cities that it can be considered as a straight line. The length of the SSW can be increased indefinitely in both directions if total cost of traveling by rocket decreases by doing so.

3. For simplicity you can assume that total number of incoming and outgoing rockets from each city is the same.

4. Sometimes exactly one city can be marked as the super city, which is the center of all activities. In that case the total of incoming and outgoing flights of that city is M times higher than an ordinary city.

5. Assume that all rockets travel in straight line.

Given the location of all the cities, you will have to find a location for the SSW for which the total cost of traveling by rockets is minimum. And you have to produce the minimum average cost per rocket flight assuming that cost of flying v unit distance is v^2.

Input

The input file contains less than 50 test cases. The description of each test case is given below:

Each test case starts with two integers N $(0 < N \le 10000)$ and Q $(0 < Q \le 100)$. Here N is the total number cities built in outer space and Q is the total number of query. Each of the next N lines contains two floating-point numbers, x_i, y_i $(0.0 \le x_i, y_i \le 1000.0)$ which denotes the coordinate of the i-th city. Cities are numbered from 0 to $N - 1$ in the order they appear in the input file. Each of the next Q lines contains two integers S $(0 \le S \le N - 1)$ and M $(1 < M \le 10000)$. Here S denotes the id of the super city and M denotes that its total number of incoming and outgoing flights in the super city is M times higher than an ordinary city.

A line containing two zeroes terminates the input. This line should not be processed.

Output

For each test case produce $Q + 2$ lines of output. The first line contains the case number followed by a colon. The second line contains a floating-point number which denotes the minimum average cost per rocket flight for the optimal super space way (Assuming all cities are ordinary). This line is followed by Q lines, one line for each query. This line contains the serial of query followed by a floating-point number. This number denotes the minimum average cost for the optimal super space way assuming that S-th city is a super city and all other cities are ordinary. All floating-point numbers in the output should have five digits after the decimal point. Look at the output for sample input for details. You can assume that for all floating-point outputs an absolute error less than 10^{-5} will be ignored.

Sample Input

```
5 2
464.9900 243.2652
463.9409 772.4632
201.9822 561.6255
695.8948 933.4567
226.0628 93.1435
3 2
4 3
4 2
27.1679 304.2512
27.7639 16.2479
921.9150 863.0064
167.6203 929.5471
2 2
2 3
0 0
```

Sample Output

```
Case 1:
16172.49971
1: 14289.23473
2: 11558.37654
Case 2:
53198.72595
1: 47995.33546
2: 41543.27604
```