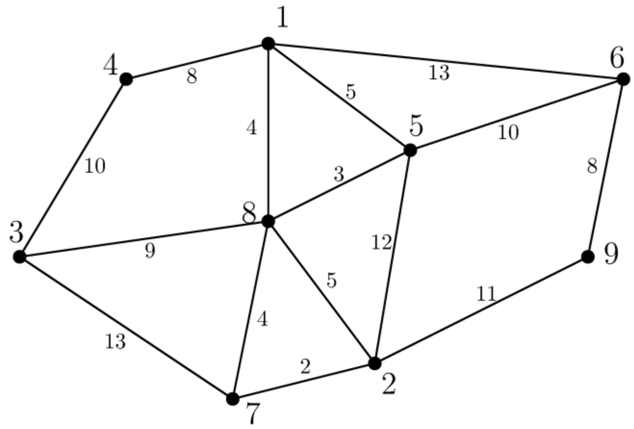
Contact

Two secret agents are living in a region which is represented by an edge weighted graph. They have to contact secretly at a location determined as follows. The time to the contact point should be minimized not to be observed from others. One requirement in this contact is that they are not allowed to stay temporally on a location. So they have move continually without any stop until they meet at a prearranged location. That is, one agent cannot stay a location in order to wait for another agent. The initial positions of two agents are at some vertices. We assume that the moving velocity of those agents are same. So the contact point can be on the vertex or the middle of the edge. Let us explain this using the following weighted graph.

If two agents are on vertices 1 and 3 in the beginning, the optimal contact point is in the edge (3,8), to be precise, at distance 6.5 from node 3, to guarantee the minimal time for contact. If two agents are on vertex 5 and 2 in the beginning, then the location for the fastest contact is on the edge (2,8). The specific point on (2,8) is the point from distance 4 from vertex 2. You are given the edge weighted graph of the city region and the initial position (vertex number) of two agents. The task is to compute the location which guarantees the fastest contact time without any intermediate staying for wait.



[Input] The first line of input file gives the number of test cases T, $T \leq 90$. The first line of each test case gives three integers N, M, K where N is the number of vertices, M is the number of edges, and K is the number of queries. Each variable is bounded $2 \leq N \leq 500$, $1 \leq M \leq 10{,}000$ and $1 \leq K \leq 3$. A query consists of two integers denoting the initial locations (vertices) of two agents. Vertex is numbered from 1 to N sequentially. And the following M lines, each line gives three integers 'i j w' to represent the edge (i, j) and its length w. The length of edge, w, is an integer between 1 and 10,000, inclusive. There is at most one edge between two vertices. After showing M lines, K lines follow to give the initial position of two agents. In each query line, two different vertex numbers are given in a single line as 'a b'. Note that you do not consider the disconnected graph where two agents can never make contacts.

Input data consists of three groups.

- Set 1: $2 \leq N \leq 20,\ 1 \leq M \leq 200, K = 1$.
- Set 2: $2 \leq N \leq 100,\ 1 \leq M \leq 1{,}000, K = 2$.
- Set 3: $2 \leq N \leq 500,\ 1 \leq M \leq 10{,}000, K = 3$.

[Output] Your task is to compute the edge (i, j) which contains the optimal contact location. Then print two integers 'i j' to represent the edge (i, j) in output file. In this procedure you should satisfy the following three rules to break ties. From now on, any edge (i, j) will be represented in the form satisfying $i \leq j$.

- **Rule (1)** If the contact point is on the edge (i,j), then print 'i j' in a single line. (i < j)
- **Rule (2)** If the contact point is exactly a vertex *i* itself which is not on the edge, then print an integer twice as 'i i' since the vertex can be considered a loop edge (i,i).
- **Rule (3)** If there are more than one contact points satisfying rule(1),(2), then we prefer the edge whose first coordinate vertex number is smaller. And if the first coordinate x in (x,y) is same, then we prefer the edge with smaller y value.

Let me explain these rule with examples. If the number of optimal edges are 4 and those (8,8), (3,6), (2,7), (5,6), then you should take the edge (2,7) and print '2 7'. If the candidate contact locations are on 4 edges such as (5,6), (3,3), (2,9), (2,6), then we should take the edge (2,6).

[Example] The weighted graph above is given in the first testing case.

Input

```
1              // T=1, 1 testing case
9 15 3    //   case 1. no. of vertices = 9, no. of edges= 15,   no. of queries=3
1 4 8       // i j w
1 5 5
1 8 4
1 6 13
2 8 5
2 7 2
9 2 11
2 5 12
3 4 10
3 8 9
3 7 13
5 8 3
5 6 10
6 9 8       // 14-th edge (6,9), the length is 8
7 8 4       // 15-th edge info. Next lines give the initial positions of agents.
1 3          // Initial positions of two agent. Vertex 1 and 3.   Query 01
5 2          // Initial positions of two agent. Vertex 5 and 2.   Query 02
3 6          // Initial positions of two agent. Vertex 3 and 6.   Query 03
```

Output

```
3 8              // contact point is in the edge (3,8)
2 8              // contact point is in the edge (2,8)
5 8              // contact point is in the edge (5,8)
```