3114 All Roads Lead to Albuquerque, er, Rome

A friend of mine has an unusual method of driving around the city, which he says helps reduce the number of routes he must memorize in order to not get lost. He picks two locations as hubs (H_1 and H_2), assigns all other locations to either H_1 or H_2, and then learns the shortest path from all locations to and from their associated hub. If he then wishes to travel from A to B, he goes from A to the hub associated with A, then to the hub associated with B (if B is associated with the other hub than A), then to B. My friend always travels to the hubs, even if that means that he visits his destination two or three times.

Your program should analyze a city (a set of nodes and edge lengths) and pick the best pair of hubs and assignment of nodes to hubs. The best configuration will be the configuration that mini- mizes the average distance of the trips between all pairs of nodes. If more than one configuration yields the lowest average, your program can output any of them.

Input

The input contains several test cases.

The input for each test case starts with a single line

n m

$2 \leq n \leq 50$ and $1 \leq m \leq 1000$. n is the number of locations in the city and m is the number of road segments that directly connect two locations in the city. There may be more than one road segment between a pair of locations, and a road segment may start and end at the same location.

Each of the next m lines will describe the road segment between two locations and will contain three integers

a b d

$1 \leq a \leq n$, $1 \leq b \leq n$, and $1 \leq d \leq 1000$. a and b are locations that describe the ends of the road segment and d is the distance required to travel from a to b (or b to a) along the road segment. There are no one-way roads.

There will always exist a path between any two locations along the given road segments.

Output

For each test case, output an optimal choice of hubs and assignment of locations to hubs by outputting a line containing n integers, separated by spaces. If the i-th location is a hub, the i-th integer should be zero. If the i-th location is not a hub, the i-th integer should give the number of the i-th location's hub (1 to n inclusive).

If more than one configuration yields the lowest average, print any of them.

Note: For the first test case, '`2 0 0`' is also valid output.

Sample Input

```
3 2
1 2 40
2 3 20
7 10
```

```
1 1 1
1 2 2
2 4 2
4 3 2
3 1 2
2 3 5
3 7 10
7 6 1
5 6 1
4 5 1
16 15
1 8 1
2 8 1
3 8 1
4 9 1
5 9 1
6 9 1
7 8 1
8 9 3
9 10 1
8 11 1
8 12 1
8 13 1
9 14 1
9 15 1
9 16 1
```

Sample Output

```
0 0 2
4 4 4 0 0 5 5
8 8 8 9 9 9 8 0 0 9 8 8 8 9 9 9
```