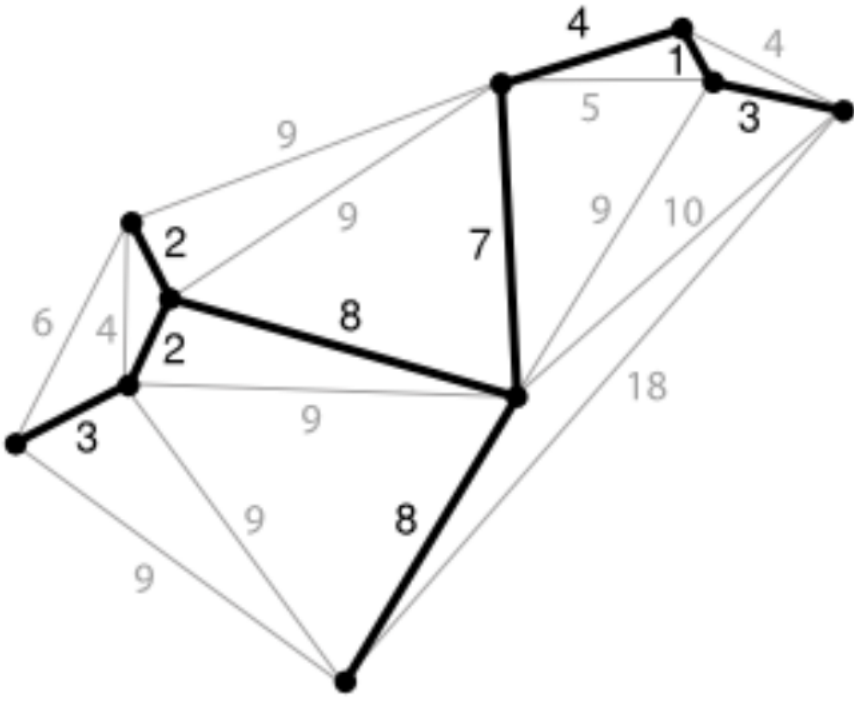
Given an undirected graph with edge weights, a minimum spanning tree is a subset of edges of minimum total weight such that any two nodes are connected by some path containing only these edges. A popular algorithm for finding the minimum spanning tree T in a graph proceeds as follows:



- let T be initially empty
- consider the edges $e_1, \ldots, e_m$ in increasing order of weight
  - add e_i to T if the endpoints of e_i are not connected by a path in T

An alternative algorithm is the following:

- let T be initially the set of all edges
- while there is some cycle C in T
  - remove edge e from T where e has the heaviest weight in C

Your task is to implement a function related to this algorithm. Given an undirected graph G wi edge weights, your task is to output all edges that are the heaviest edge in some cycle of G.

Input

The first input of each case begins with integers n and m with $1 \leq n \leq 1,000$ and $0 \leq m \leq 25,0$ where n is the number of nodes and m is the number of edges in the graph. Following this are lines containing three integers u, v, and w describing a weight w edge connecting nodes u and v whe $0 \leq u, v < n$ and $0 \leq w < 2^{31}$. Input is terminated with a line containing $n = m = 0$; this case shou not be processed. You may assume no two edges have the same weight and no two nodes are direct connected by more than one edge.

Output

Output for an input case consists of a single line containing the weights of all edges that are the heavie edge in some cycle of the input graph. These weights should appear in increasing order and consecuti weights should be separated by a space. If there are no cycles in the graph then output the text ‘fores instead of numbers.

Sample Input

```
3 3
```

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

Sample Output

```
3
2 4
forest
```