```python
'''
    Time complexity: O(E * log(E))
    Space complexity: O(V + E)

    where E is the number of edges in the graph and
    V is the number of vertices in the graph
'''
class Edge:
    def __init__(self,src,dest,wt):
        self.src = src
        self.dest = dest
        self.wt = wt

def getParent(v,parent):
    if v == parent[v]:
        return v
    return getParent(parent[v],parent)
def kruskal(edges,n,E):
    edges = sorted(edges,key = lambda edge:edge.wt)
    output = []

    parent = [i for i in range(n)]
    count = 0
    i = 0
    while count < (n-1):
        currentEdge = edges[i]
        srcParent = getParent(currentEdge.src,parent)
        destParent = getParent(currentEdge.dest,parent)

        if srcParent != destParent:
            output.append(currentEdge)
            count+=1
            parent[srcParent] = destParent
        i+=1
    return output

li = [int(ele) for ele in input().split()]
n = li[0]
E = li[1]
edges = []
for i in range(E):
    curr_input = [int(ele) for ele in input().split()]
    src = curr_input[0]
    dest = curr_input[1]
    wt = curr_input[2]
    edge = Edge(src,dest,wt)
    edges.append(edge)
```

```
output = kruskal(edges,n,E)
for ele in output:
    if(ele.src < ele.dest):
        print(str(ele.src) + " " + str(ele.dest) + " " + str(ele.wt))
    else:
        print(str(ele.dest) + " " + str(ele.src) + " " + str(ele.wt))
```