```
n = 8
edges = [[0,1],[0,2],[1,2],[1,3],[1,4],[4,5],[6,7],[2,1],[1,0]]

adj_list = [[] for j in range(n)]
for edge in edges:
    adj_list[edge[0]].append(edge[1])





# computing MST
n = 8
edges = [[0,1,1],[0,2,5],[1,2,7],[1,3,2],[1,4,1],[4,5,3],[6,7,6]]

adj_list = [[] for j in range(n)]
for edge in edges:
    adj_list[edge[0]].append([edge[1],edge[2]])


# computing MST: naive solution
initialization
for i from 1 to n
    scan all vertices find one vertex u with the smallest distance to the tree
    mark u as visited
    for each neighbor v of u
        update distance of v if necessary
        if distance of v is updated: set u as parent of v



# computing MST

import heapq
mst = []

dist = [float('inf')] * n
visited = [False] * n
for u in visited:
    if visited[u] == True:
        continue
    dist[u] = 0
    pq = [(0,(u,-1))]
    while len(pq) > 0:
        _, (v, pre_v) = heapq.heappop(pq)
        if visited[v]:
            continue
        visited[v] = True
        mst.append((_,v,pre_v))
        for w,weight in adj_list[v]:
            if weight < dist[w]:
                dist[w] = weight
                heapq.heappush(pq,(dist[w],(w,v)))

print(mst)





# Tarjan's algorithm for SCC

def DFS(u):
    global count
    num[u] = count
    lowest[u] = num[u]
    count = count + 1
    visited[u] = True
    in_stack[u] = True
    s.append(u)

    for v in adj_list[u]:
        if visited[v] == False:
            DFS(v)
            lowest[u] = min(lowest[v],lowest[u])
        else:
            if in_stack[v] == True:
                lowest[u] = min(lowest[v],lowest[u])

    if lowest[u] == num[u]:
        scc = []
        w = s.pop()
        in_stack[w] = False
        while w != u:
            scc.append(w)
            w = s.pop()
            in_stack[w] = False
        scc.append(w)
        # process the scc
        print(scc)

visited = [False] * n
in_stack = [False] * n
lowest = [0] * n
num = [0] * n
count = 0
s = []

for u in visited:
# what happens if the graph is not connected or starting from a bad source?
# add a virtual node...
    if visited[u] == False:
        DFS(u)
```