Task 1a

To solve this task with dfs, at first initialize an array where we will mark if that node is visited or not. Then run dfs function, and if that node is not visited then push in the queue. If there's no where to go then pop that element and append in another array. After finishing the dfs, if we reverse the last array we can get our desired array.

Task 1b

To solve this task with bfs, at first we need an array where every node's in-degree number will store. Now, the node whose in-degree is zero will push in queue. Then, start with the first elem and run bfs. Whenever go to another node, just minus the in-degree number of that node. If it is zero now then push in the queue. If there is no where to go then just append it in an array and it will be the desired array.

Task 2

For this task, everything will be same as task 1b, just there is a little change that whenever dequeueing from the queue, the element should be minimum among them and run the bfs from there.

Task 3

To solve this task, at first run dfs in the graph and store in an array in ending order. Then reverse this array, and also transverse the graph. Again run dfs in the transverse graph. But this time dfs will start in order of reverse array, that reverse array. Now we will get our scc.