Name: Razin Sufian

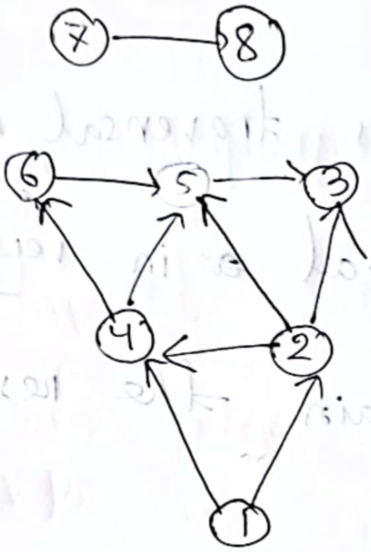
ID: 22301219

Sec: 16

LAB assignment 5

code explanation

Task 1(a)



3
5
6
4
8
2
7
1

1 7 2 8 4 6 5 3

node	indegree
1	0
2	~~1~~ 0
3	~~2~~ ~~1~~ 0
4	~~2~~ ~~1~~ 0
5	~~3~~ ~~2~~ ~~1~~ 0
6	~~1~~ 0
7	0
8	~~0~~ ~~1~~ 0

where first we have count all the indegrees of every node the if any node's indegree is 0 we add to the queue and visit all his neighbours And updated the indegree count of that visited nodes (-1). the queue is our answer.

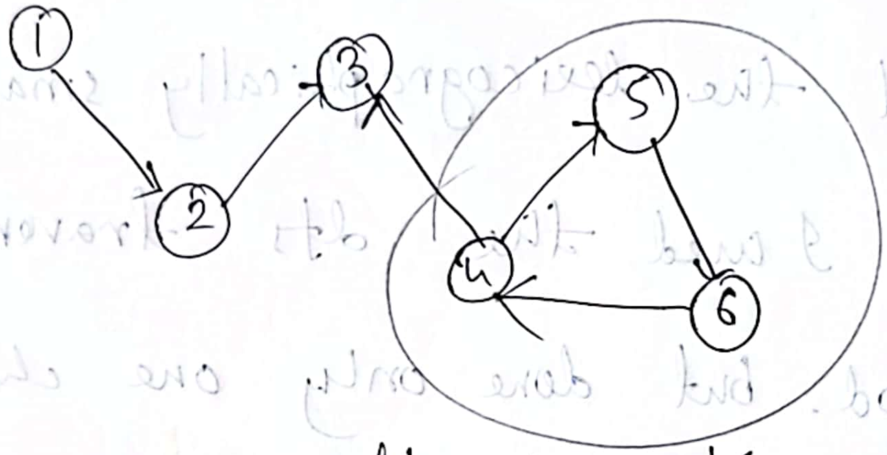
1. (a)

Here ~~we~~ for the dfs traversal we would just do normal dfs traveral ~~or~~ in very unvisited node and maintain the recursion stack.

The recursion satack in the opposite order [: : -1] will be our answer.

- But we will face problem while detectis the "Impossible" cases.

An impossible cases :



here we can, it'sa impossible case.

we can detect impossible cases if any circle is found. So, I have take a rec-stack ~~list~~ ⇒ boolian list to detect any circle. if we find any circle, we just return 'Impossible'.

Task 2

to get the lexicographically smaller path; I used the dfs traversal method. but done only one chage before deque anything for the queue we just ~~so~~ made the queue sorted; so that we can maintain lexicographically smaller path

queue = deque(sorted(queue))

Tasks:

find scc:

fun :dfs:

① it's a function give us the dfs recursion stack

② the transpose-graph give us the same graph just in the opposite direction.

③ then we run dfs -scc in the the transpose-graph in ~~the~~ according to the recursion stack. and we find-scc to detect every cycle.