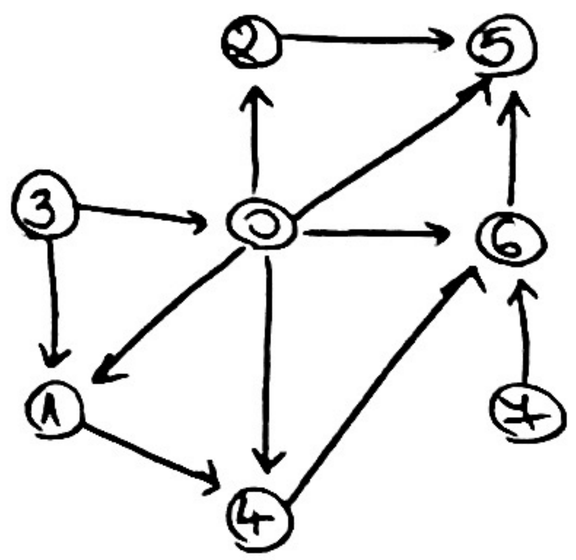


Outbound :

0	[1,2,4,5,6]
1	[4]
2	[5]
3	[0,1]
4	[6]
5	[0]
6	[5]
7	[6]

The graph :

0 1 4
0 2 7
0 4 8
0 5 50
0 6 45
1 4 37
2 5 11
3 0 2
3 1 23
4 6 99
6 5 13
7 6 78

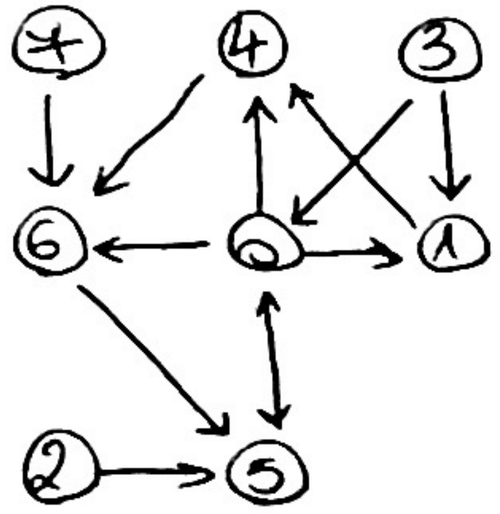
	visited	stack
initialization	0 1 2 3 4 5 6 7 F F F F F F F F	[]
i=0, iteration 1.1	0 1 2 3 4 5 6 7 T F F F F F F F	[]
iteration 1.2 neighbour = 1	0 1 2 3 4 5 6 7 T T F F F F F F	[]
iteration 2, i=1 neighbour = 4	0 1 2 3 4 5 6 7 T T F F T F F F	[5,6,4,1]
iteration 3, i=4 neighbour = 6	0 1 2 3 4 5 6 7 T T F F T F T F	[5,6,4]
iteration 4.1, i=6 neighbour = 5	0 1 2 3 4 5 6 7 T T F F T T T F	[5,6]
iteration 5, i=5 neighbour = None	0 1 2 3 4 5 6 7 T T F F T T T F	[5]
iteration 1.3, neighbour = 2	0 1 2 3 4 5 6 7 T T T F T T T F	
iteration 6, i=2 neighbour = None	0 1 2 3 4 5 6 7 T T T F T T T F	

Stack

[5]
[5,6]
[5,6,4]
[5,6,4,1]
[5,6,4,1,2]
[5,6,4,1,2,3]
[5,6,4,1,2,0]
[5,6,4,1,2,0,3]
[5,6,4,1,2,0,3,7]

Topological sort: [7,3,0,2,1,4,6,5]

iteration 1. 4+5+6 neighbour = 4+5+6	0 1 2 3 4 5 6 7 T T T F T T T F
iteration 7, i=3 neighbour = 0+1	0 1 2 3 4 5 6 7 T T T T T T T F
iteration 8, i=7	0 1 2 3 4 5 6 7 T T T T T T T T



The graph: 0 1 4 | 2 5 11 | 0 4 8 | 3 0 2 | 0 5 50 | 3 1 23 | 0 6 45 | 4 6 99 | ~~0 1 37~~ | 5 0 34 | 1 4 37 | 6 5 13 | 7 6 78

Outbound:

0	-[1,4,5,6]
1	-[4]
2	-[5]
3	-[0,1]
4	-[6]
5	[0]
6	-[5]
7	-[6]

	visited	stack
initialization	0 1 2 3 4 5 6 7 F F F F F F F F	0 1 2 3 4 5 6 7 F F F F F F F F
i=0, iteration 1 neighbour = 1	0 1 2 3 4 5 6 7 T F F F F F F F	0 1 2 3 4 5 6 7 T F F F F F F F
i=1, iteration 2 neighbour = 4	0 1 2 3 4 5 6 7 T T F F F F F F	0 1 2 3 4 5 6 7 T T F F F F F F
i=4, iteration 3 neighbour = 6	0 1 2 3 4 5 6 7 T T F F T F F F	0 1 2 3 4 5 6 7 T T F F T F F F
i=6, iteration 4 neighbour = 5	0 1 2 3 4 5 6 7 T T F F T F T F	0 1 2 3 4 5 6 7 T T F F T F T F
i=5, iteration 5 neighbour = 0	-11-	-11-

→ this is the is_dag() function

stack[0] = True => there is a cycle,
the graph is cyclic