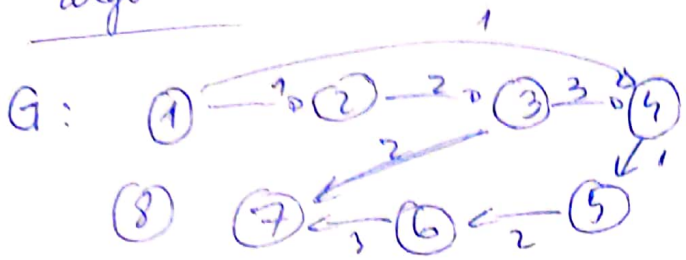
Topological sorting using predecessor counting algorithm

G:



topological sorting orders:

8, 1, 2, 3, 4, 5, 6, 7

1, 8, 2, 3, 4, 5, 6, 7

	x, y	count - dictionary	q: queue	sorted : list
initialization		1 2 3 4 5 6 7 8 0 1 1 2 1 1 2 0	← 1 8 ←	[ ]
iteration 1	x = 1 y = 2 y = 4	1 2 3 4 5 6 7 8 0 0 1 1 1 1 2 0	← 8 ← ← 8 2 4 ←	[1]
iteration 2	x = 8	1 2 3 4 5 6 7 8 0 0 1 1 1 1 2 0	← 2 4 ←	[1, 8]
iteration 3	x = 2 y = 3	1 2 3 4 5 6 7 8 0 0 0 1 1 1 2 0	4 3 ←	[1, 8, 2]
iteration 4	x = 3 y = 4 y = 7	1 2 3 4 5 6 7 8 0 0 0 0 1 1 1 0	← 4 ←	[1, 8, 2, 3]
iteration 5	x = 4 y = 5	1 2 3 4 5 6 7 8 0 0 0 0 0 1 1 0	← 5 ←	[1, 8, 2, 3, 4]
iteration 6	x = 5 y = 6	1 2 3 4 5 6 7 8 0 0 0 0 0 0 1 0	← 6 ←	[1, 8, 2, 3, 4, 5]
iteration 7	x = 6 y = 7	1 2 3 4 5 6 7 8 0 0 0 0 0 0 0 0	← 7 ←	[1, 8, 2, 3, 4, 5, 6]
iteration 8	x = 7	1 2 3 4 5 6 7 8 0 0 0 0 0 0 0 0	stop.	[1, 8, 2, 3, 4, 5, 6, 7] sizeof(sorted) = 8 G is a DAG.