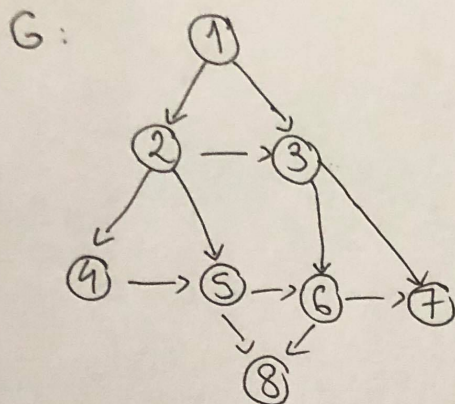


Manual execution

Topological sorting using predecessor counting



Topological sorting orders:

I 1 2 3 4 5 6 7 8

~~II 1 3 2 6 7 4 5 8~~

~~III 1 2 4 5 3 6 7 8~~

IV 1 2 3 4 5 6 8 7

III 1 2 4 5 3 6 7 8

IV 1 2 4 5 3 6 8 7

V 1 2 4 3 5 6 7 8

VI 1 2 4 3 5 6 8 7

	X, Y	count: dictionary	q: queue	sorted: list
initialization		1 2 3 4 5 6 7 8 0 1 2 1 2 2 2 2	← 1 ←	[]
iteration 1	X=1 Y=2 Y=3	1 2 3 4 5 6 7 8 0 0 1 1 2 2 2 2	← ← ← 2 3 ←	[1]
iteration 2	X=2 Y=4 Y=5 Y=3	1 2 3 4 5 6 7 8 0 0 0 0 1 2 2 2	← 3 ← ← 3 4 5 ←	[1, 2]
iteration 3	X=3 Y=6 Y=7	1 2 3 4 5 6 7 8 0 0 0 0 1 1 1 2	← 4 5 ← ← 4 5 6 7 ←	[1, 2, 3]
iteration 4	X=4 Y=5	1 2 3 4 5 6 7 8 0 0 0 0 0 1 1 2	← 5 6 7 ←	[1, 2, 3, 4]
iteration 5	X=5 Y=6 Y=8	1 2 3 4 5 6 7 8 0 0 0 0 0 0 1 1	← 6 7 ← ← 6 7 8 ←	[1, 2, 3, 4, 5]
iteration 6	X=6 Y=7 Y=8	1 2 3 4 5 6 7 8 0 0 0 0 0 0 0 0	← 7 8 ←	[1, 2, 3, 4, 5, 6]
iteration 7	X=7	— " —	← 8 ←	[1, 2, 3, 4, 5, 6, 7]
iteration 8	X=8	— " —	← ←	[1, 2, 3, 4, 5, 6, 7, 8]

STOP

G is a DAG
size of (sorted) = 8