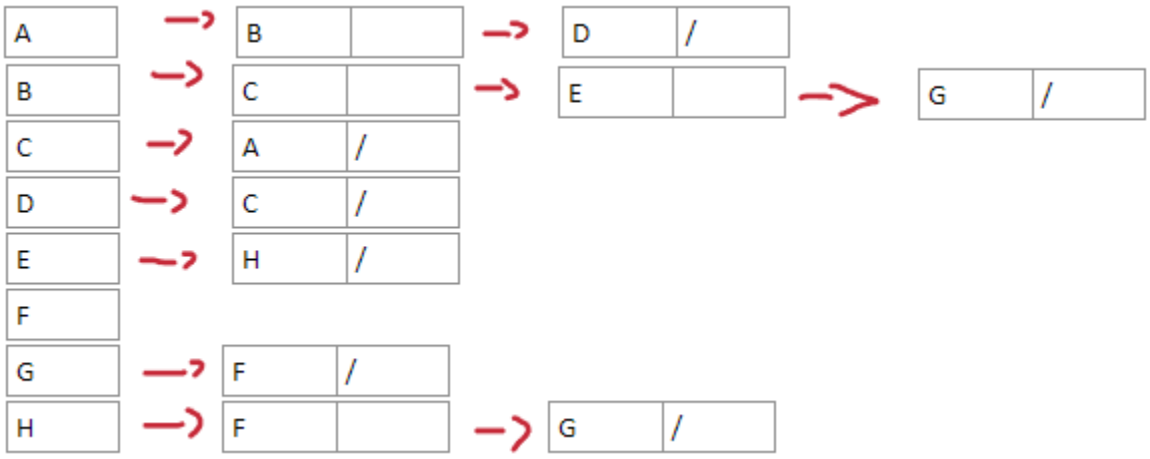
1

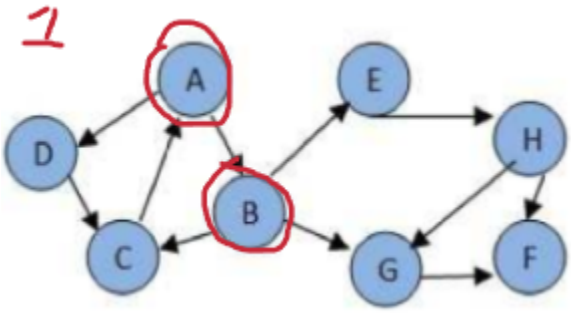
Matrix:

	A	B	C	D	E	F	G	H
A	0	1	0	1	0	0	0	0
B	0	0	1	0	1	0	1	0
C	1	0	0	0	0	0	0	0
D	0	0	1	0	0	0	0	0
E	0	0	0	0	0	0	0	1
F	0	0	0	0	0	0	0	0
G	0	0	0	0	0	1	0	0
H	0	0	0	0	0	1	1	0

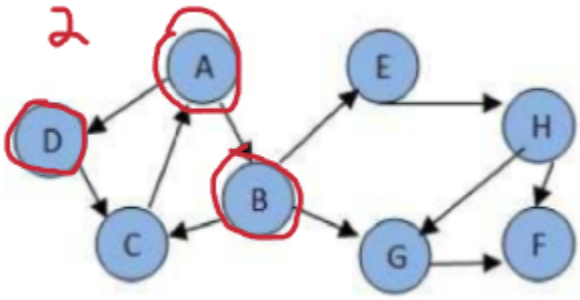
List:



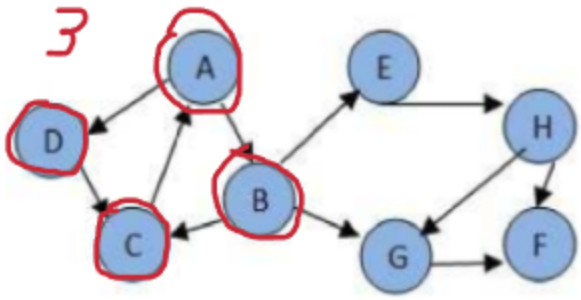
BFS:



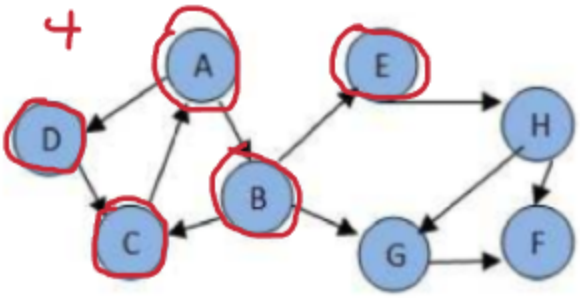
Output: A B



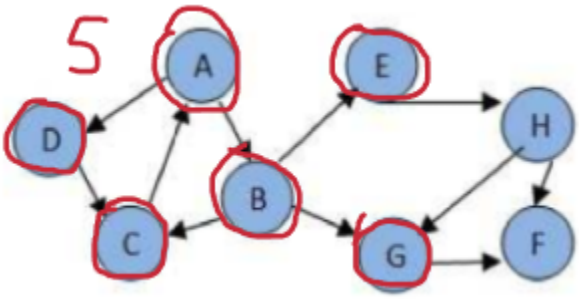
Output: A B D



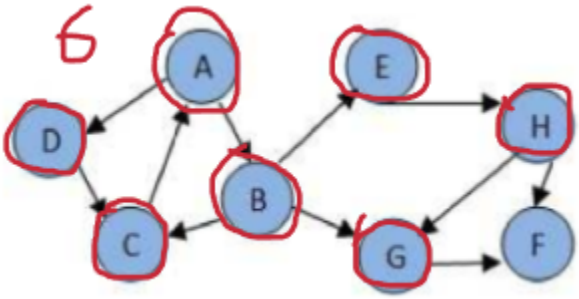
Output: A B D C



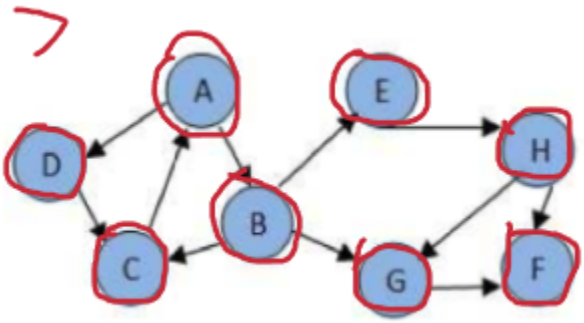
Output: A B D C E



Output: A B D C E G

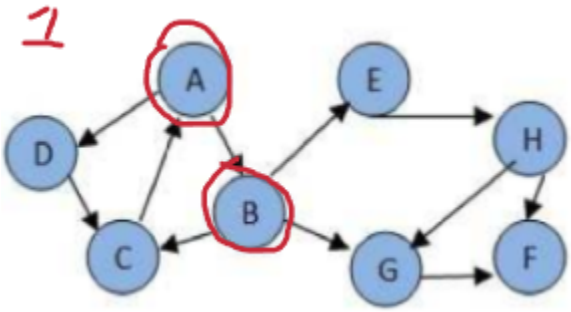


Output: A B D C E G H

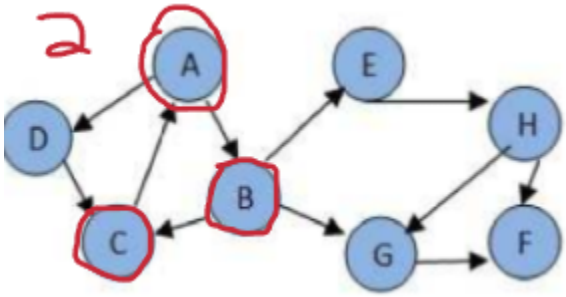


Output: A B D C E G H F

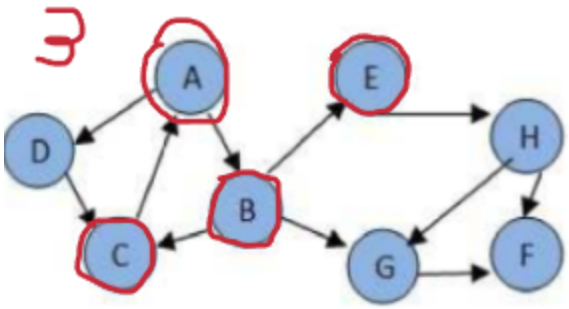
DFS:



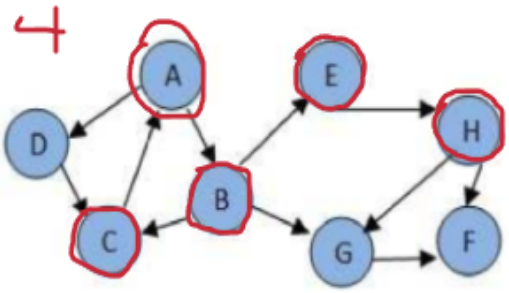
Output: A B



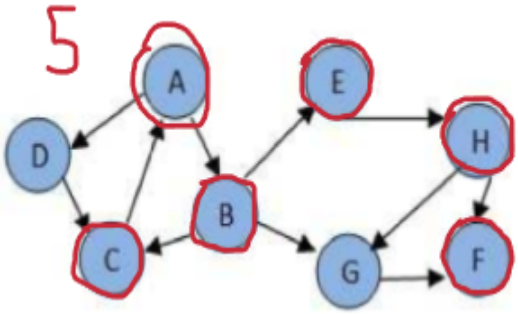
Output: A B C



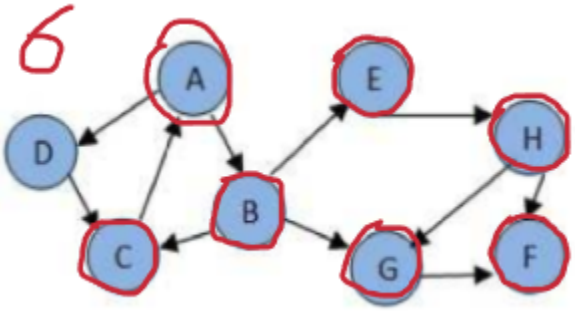
Output: A B C E



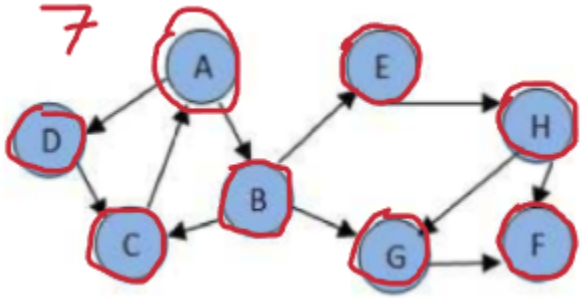
Output: A B C E H



Output: A B C E H F



Output: A B C E H F G



Output: A B C E H F G D

Discovery and finish times:

Vertex	A	B	C	D	E	F	G	H
Discovery	1	2	3	14	5	7	11	6
Finish	16	13	4	15	10	8	12	9

#2

Apply DFS() on each component. In each DFS() call, a component or a sub-graph will be visited. Call DFS on the next un-visited component. The number of calls to DFS() will be the number of connected components

#3

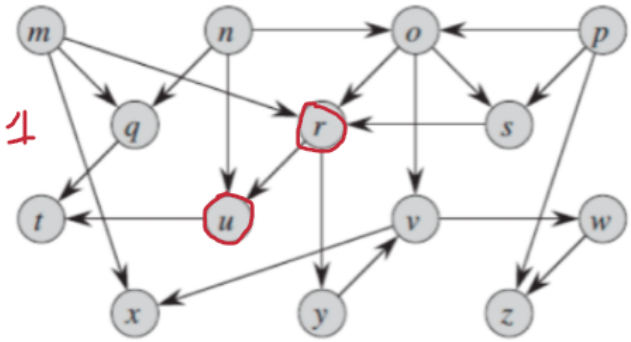
Directed:

(i , j)	White	Grey	Black
White	Yes Tree, Cross, Forward, Back	Yes Cross, Back	Yes Cross
Grey	Yes Tree, Forward	Yes Tree, Back, Forward	Yes Tree, Cross, Forward
Black	No	Yes Back, Cross	Yes Tree, Cross, Forward, Back

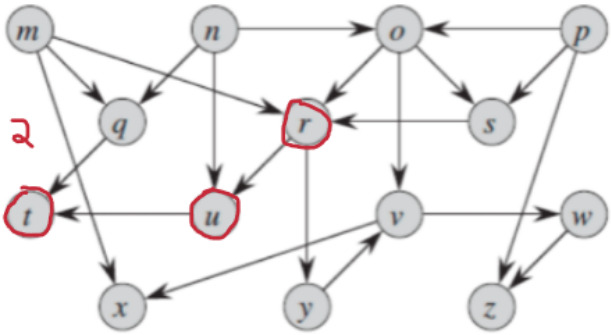
Undirected:

(i , j)	White	Grey	Black
White	Yes Tree, Back	Yes Tree, Back	No
Grey	Yes Tree, Back	Yes Tree, Back	Yes Tree, Back
Black	No	Yes Back, Tree	Yes Tree, Back

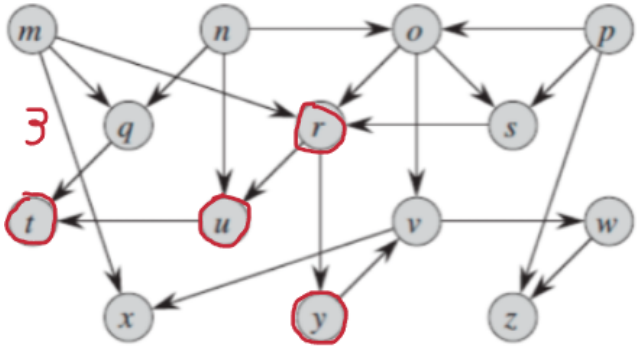
#4



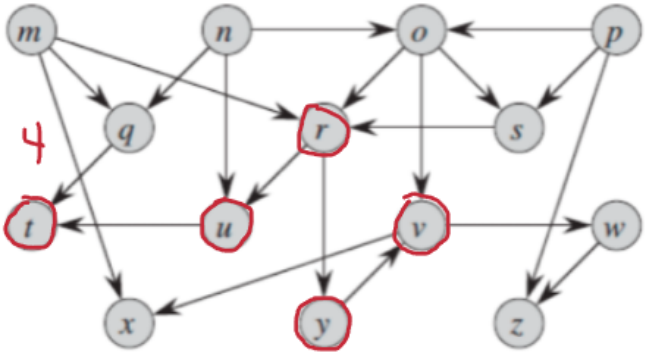
Output: r u



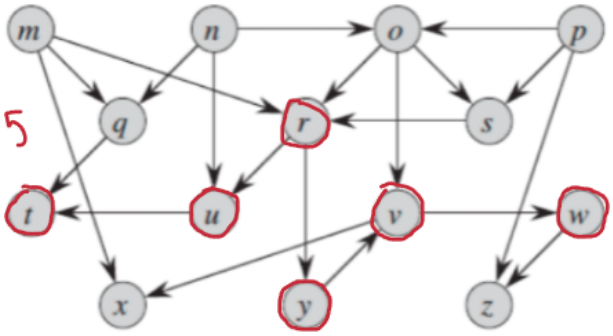
Output: r u t



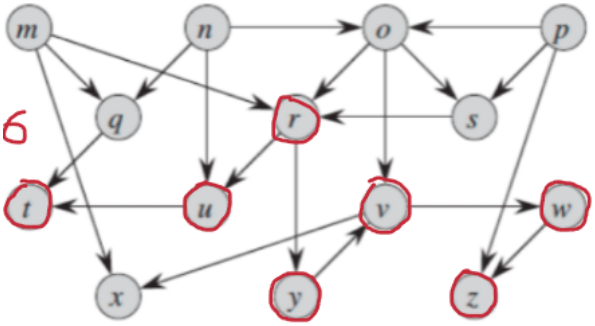
Output: r u t y



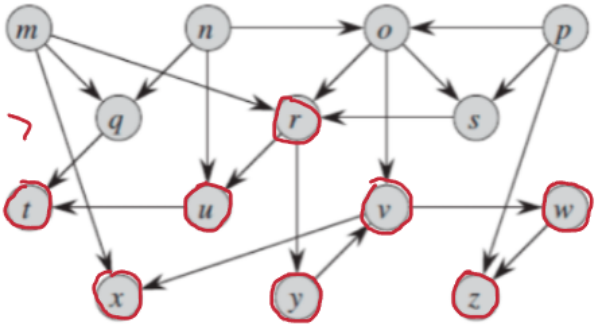
Output: r u t y v



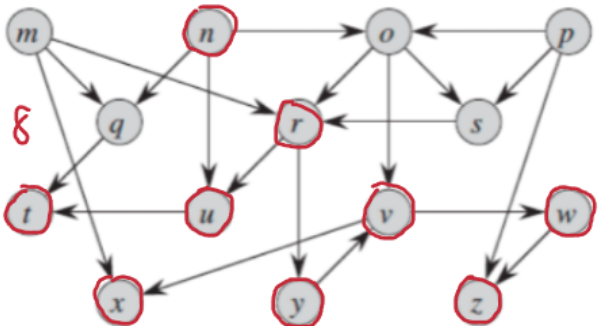
Output: r u t y v w



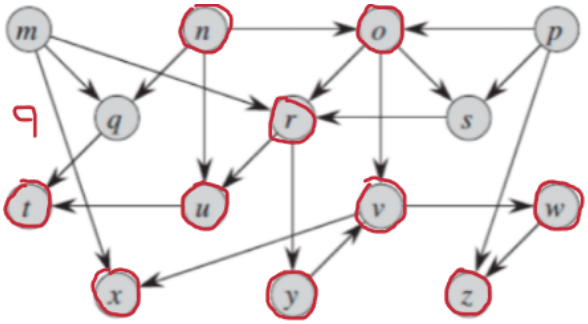
Output: r u t y v w z



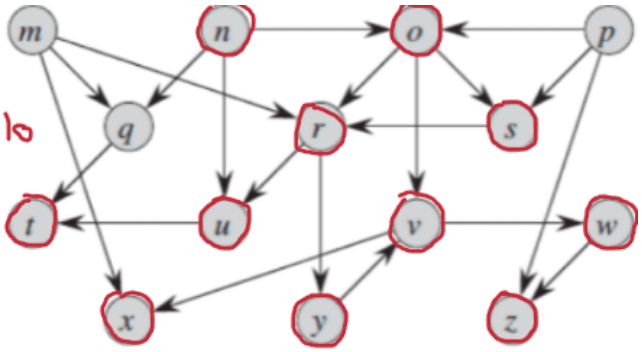
Output: r u t y v w z x



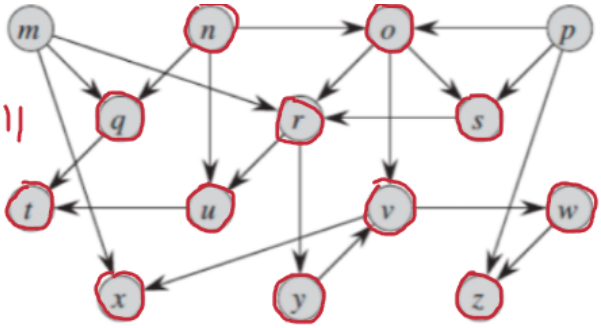
Output: r u t y v w z x n



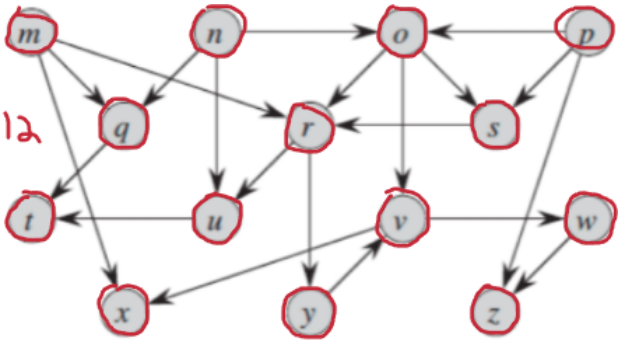
Output: r u t y v w z x n o



Output: r u t y v w z x n o s



Output: r u t y v w z x n o s q



Output: r u t y v w z x n o s q p m

#5