BLG 336E Analysis of Algorithms II

Recitation 3

Alperen Kantarcı

Istanbul Technical University

15 March, 2021

PART I. How to traverse a graph?

Breadth First Search (BFS) traversing algorithm: traverse a graph layer wise.

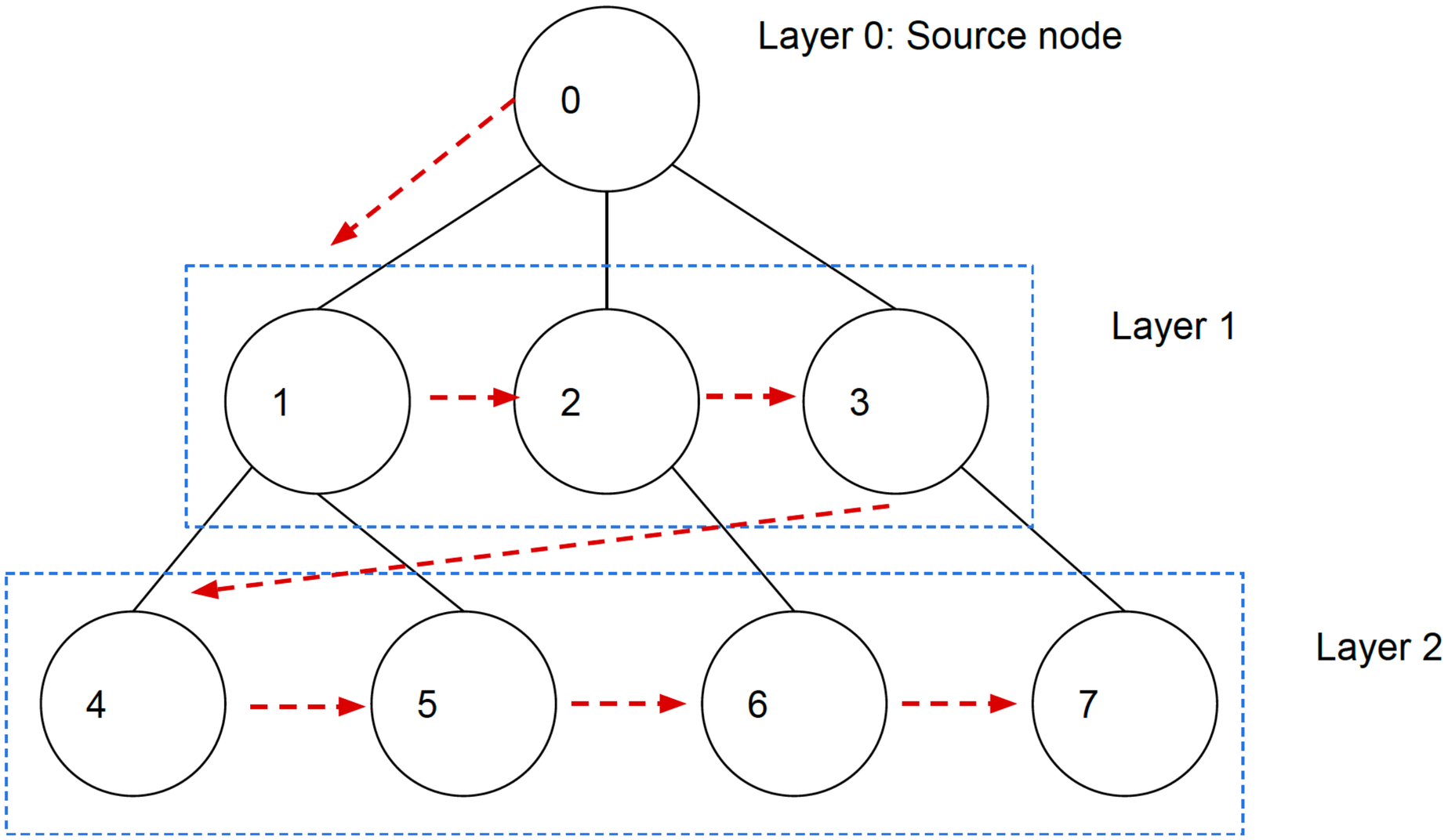


Figure 1. BFS Algorithm

Breadth First Search algorithm pseudocode

Breadth First Search (BFS) traversing algorithm: traverse a graph layer wise.

```
Create a queue Q
Mark starting vertex V as visited and put V into Queue Q

While Queue Q is not empty
      Remove the head node U of the Queue Q
      Mark and enqueue (add) all unvisited neighbors of node U into Queue Q
```

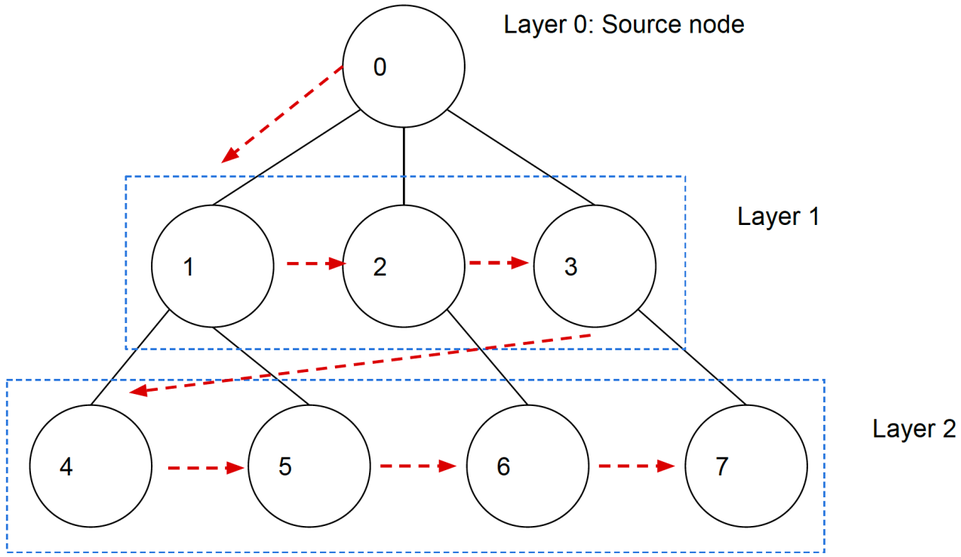
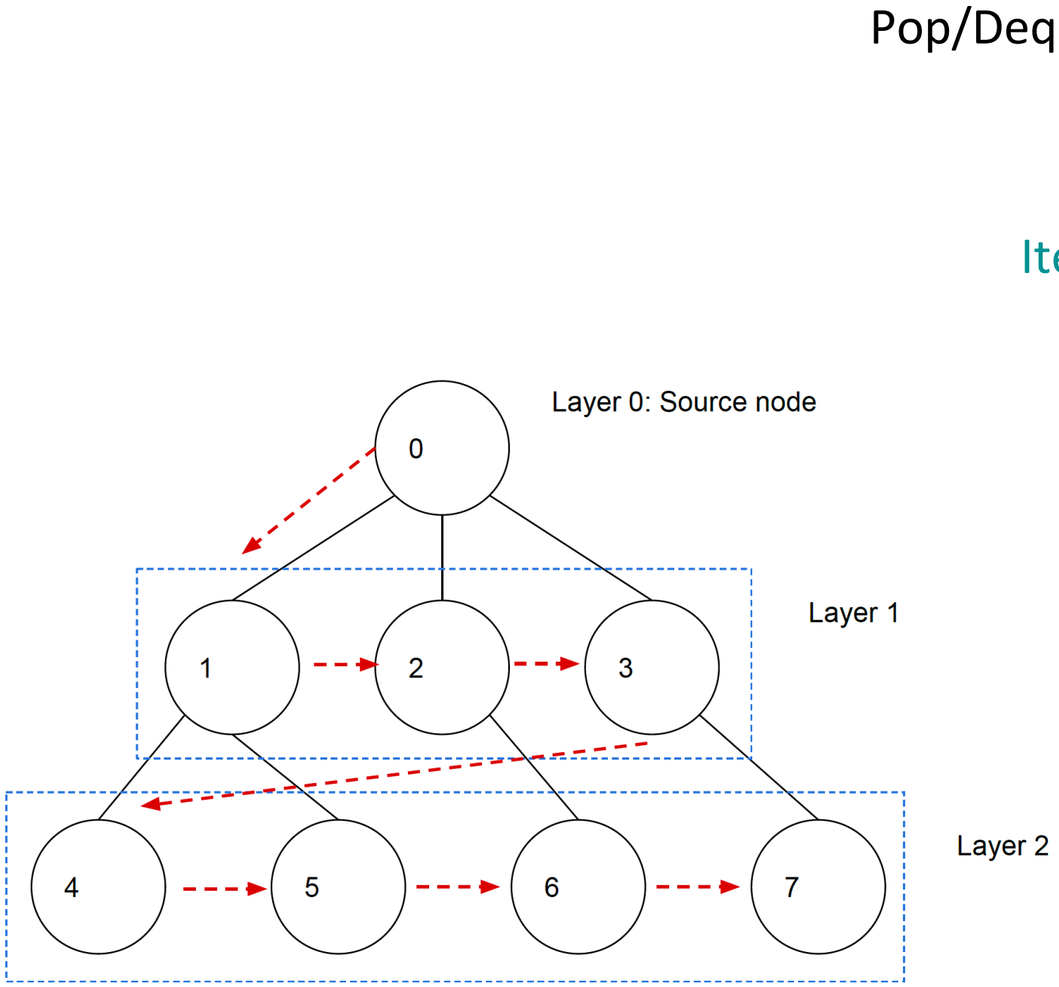


Figure 2. BFS Algorithm pseudocode

BFS algorithm traversals with illustrations



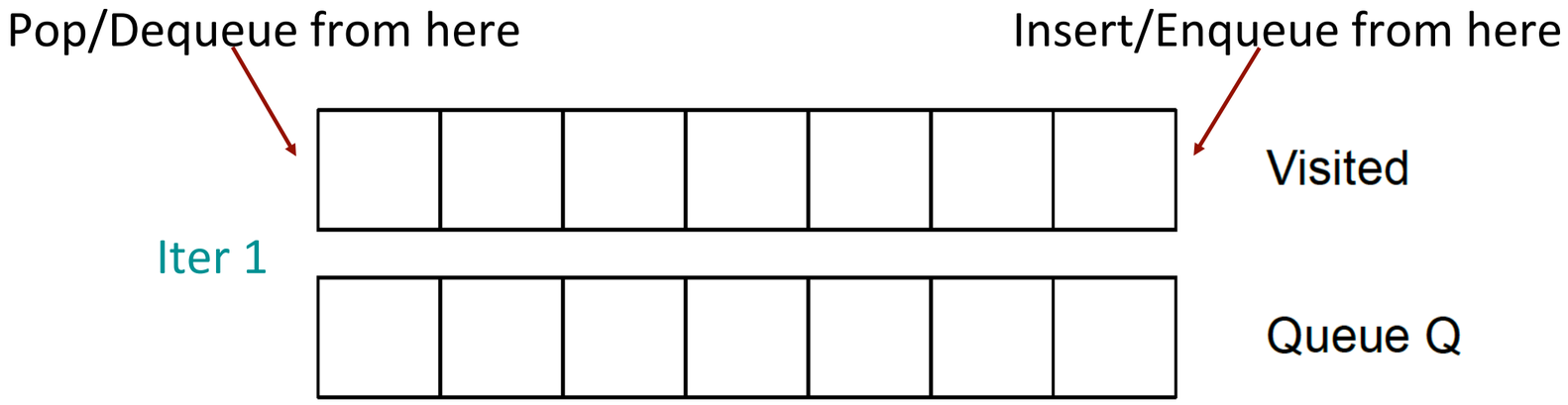
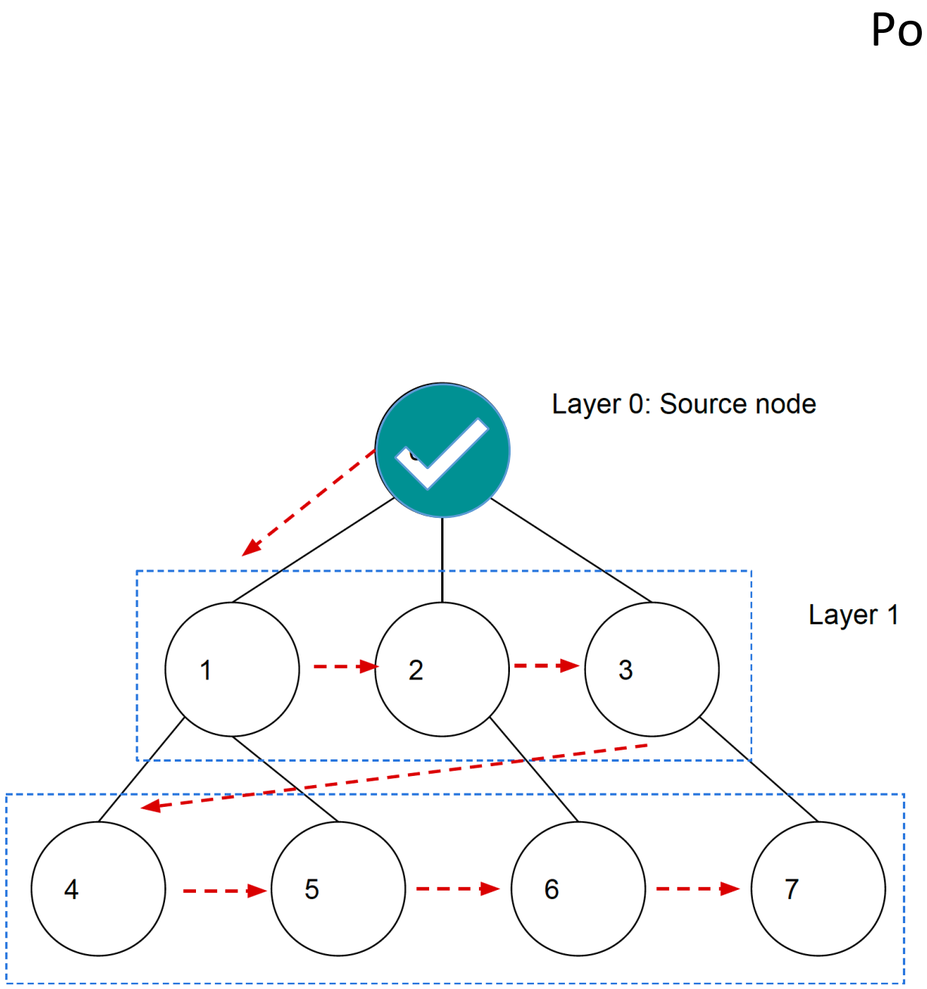


Figure 3. BFS Algorithm traversals with illustrations

BFS algorithm traversals with illustrations



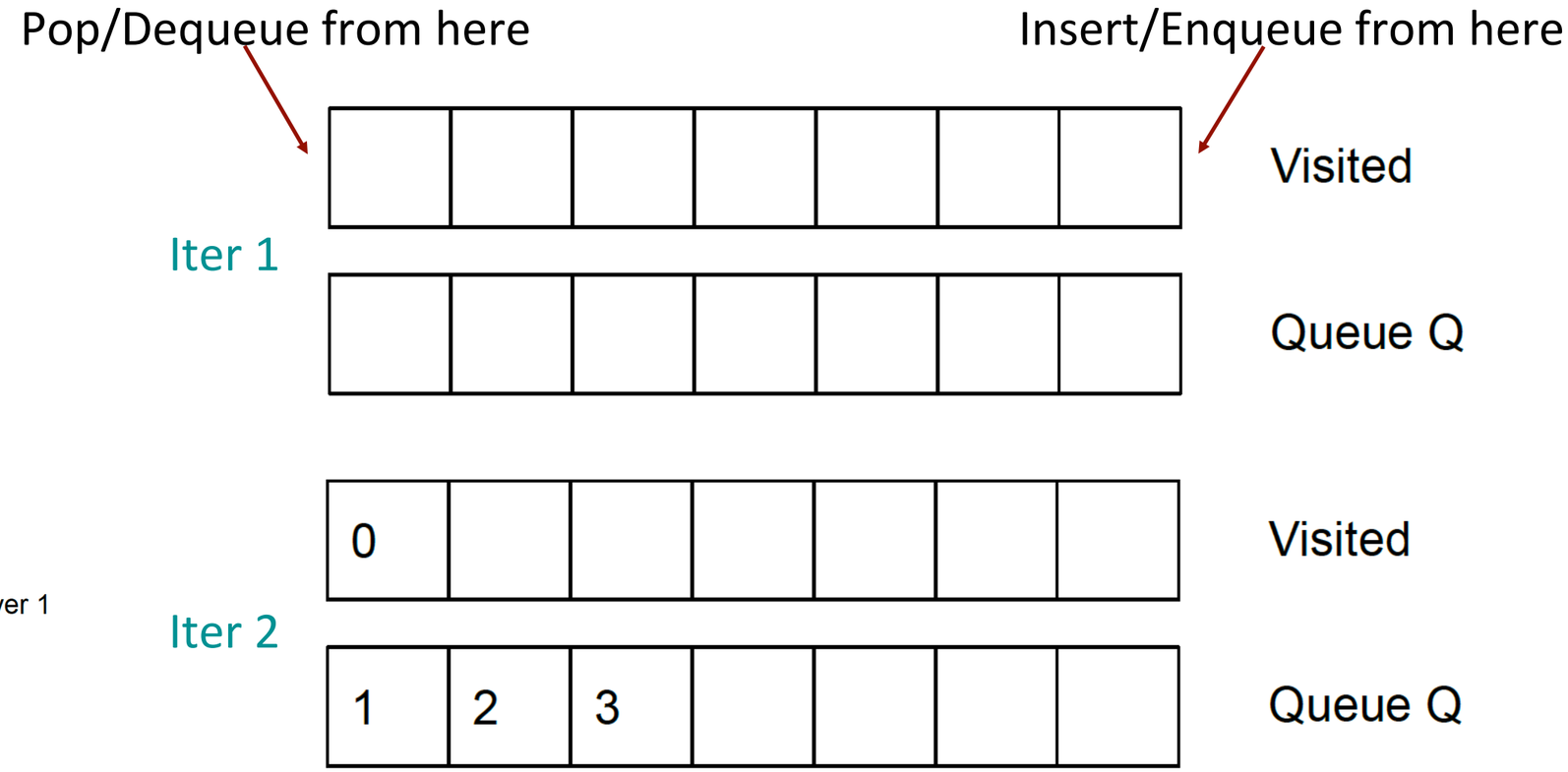
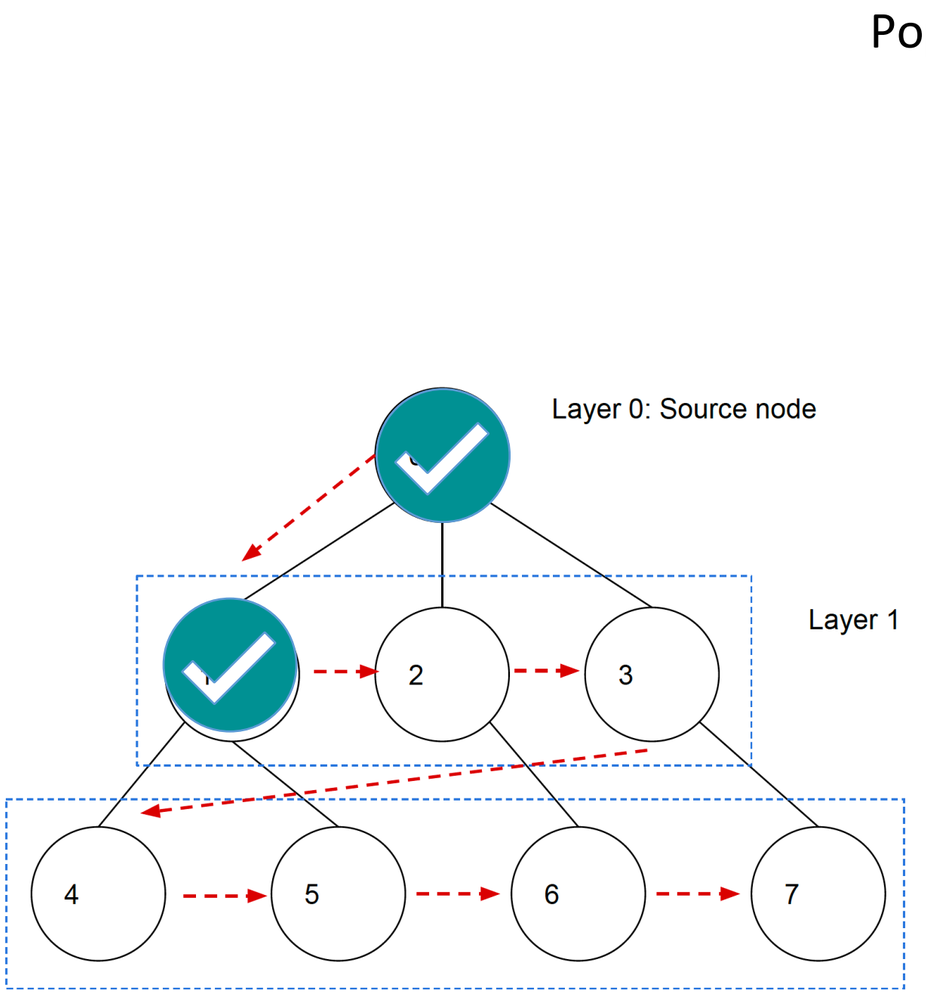


Figure 3. BFS Algorithm traversals with illustrations

BFS algorithm traversals with illustrations



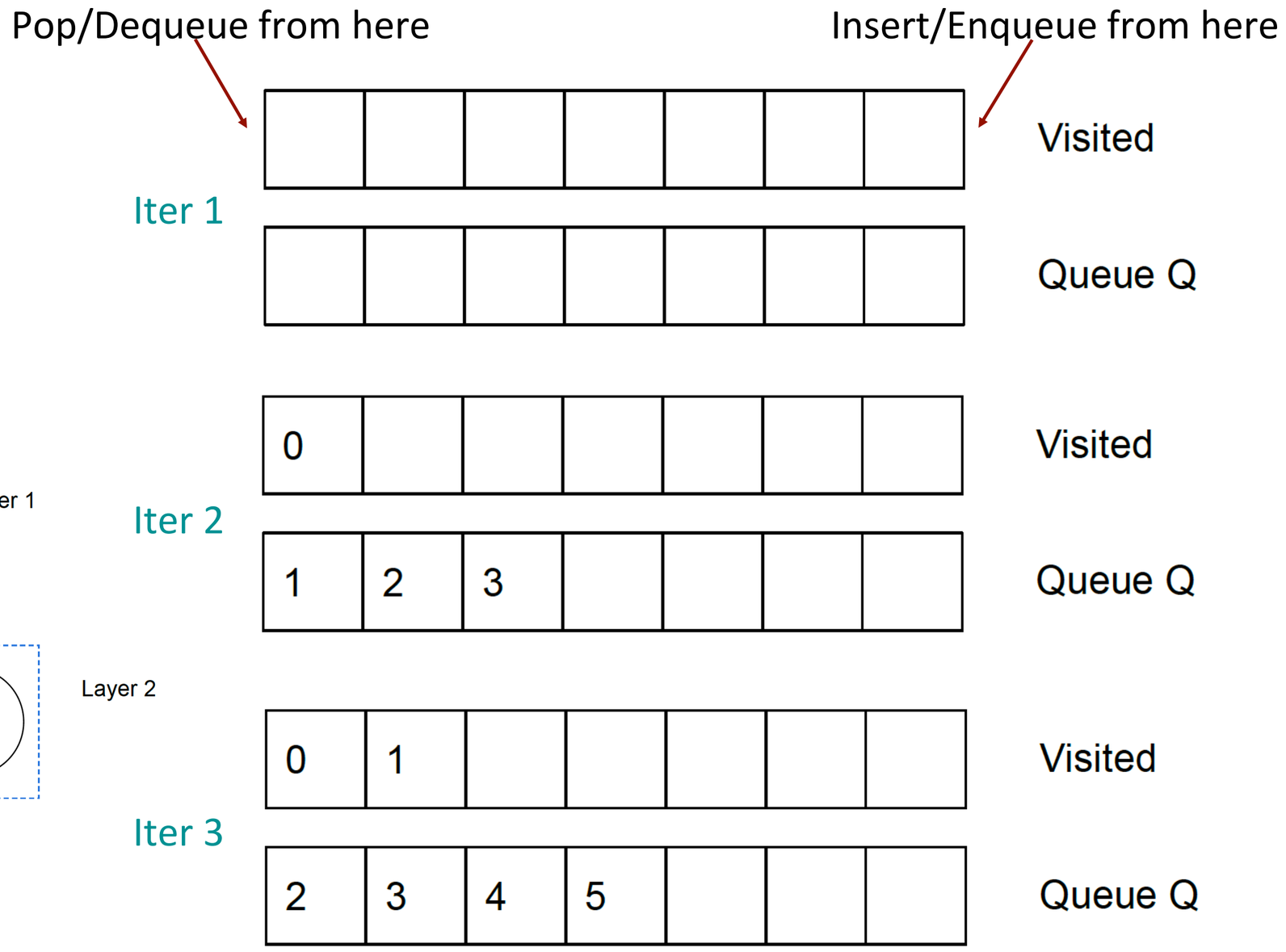
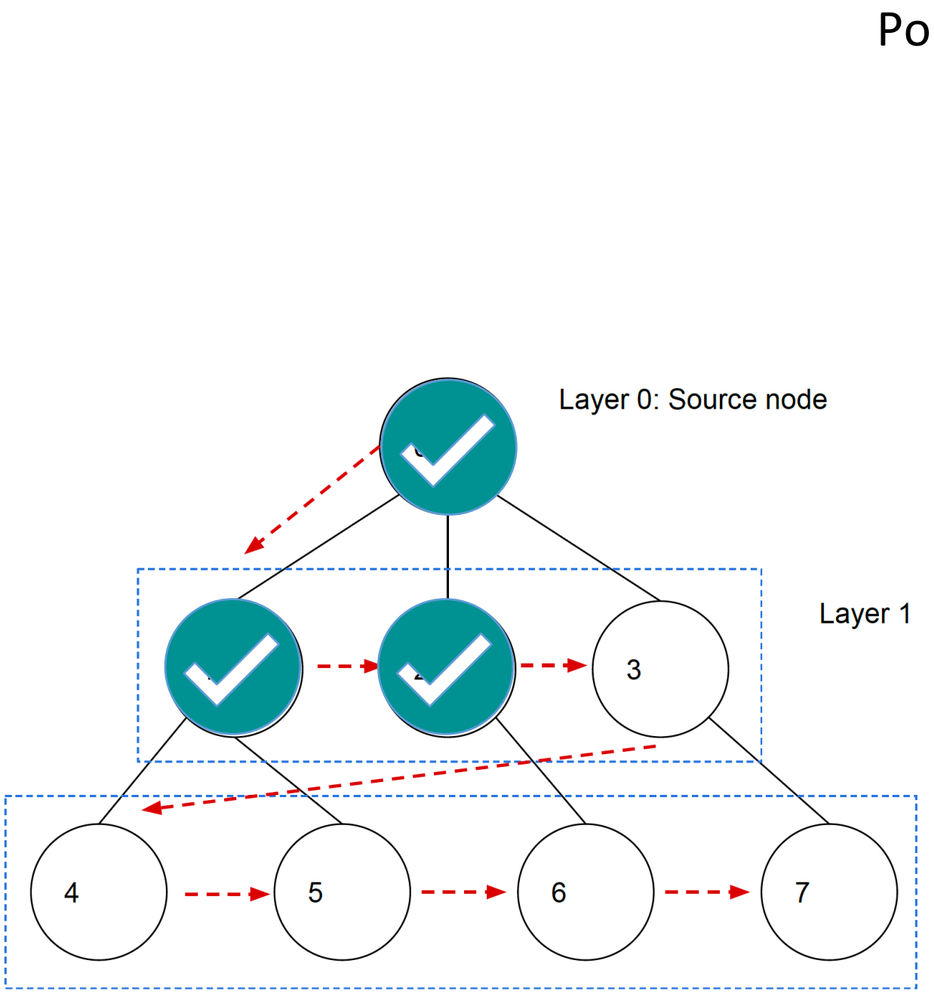


Figure 3. BFS Algorithm traversals with illustrations

BFS algorithm traversals with illustrations



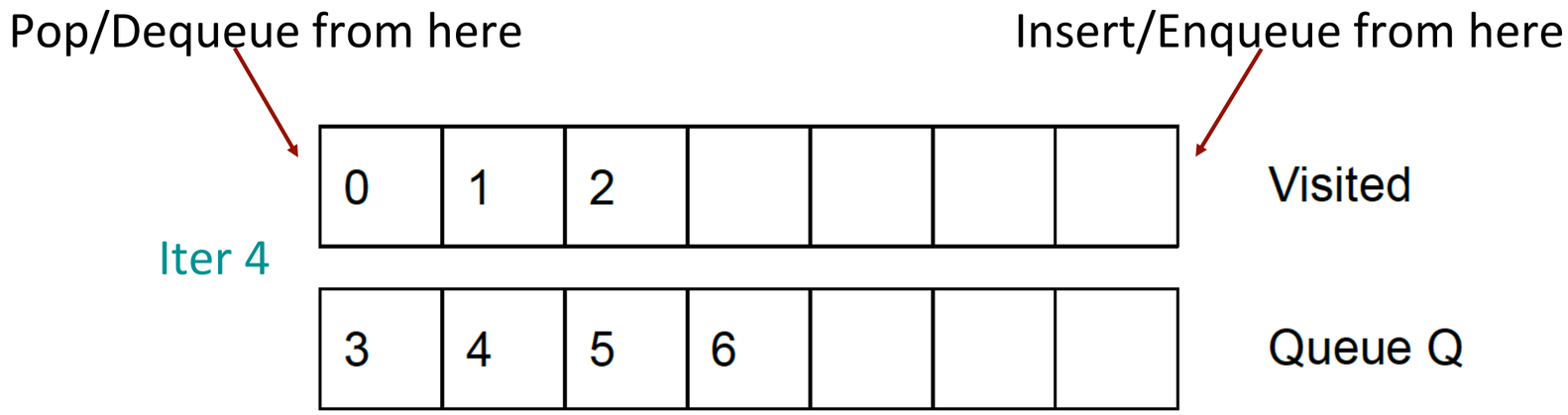
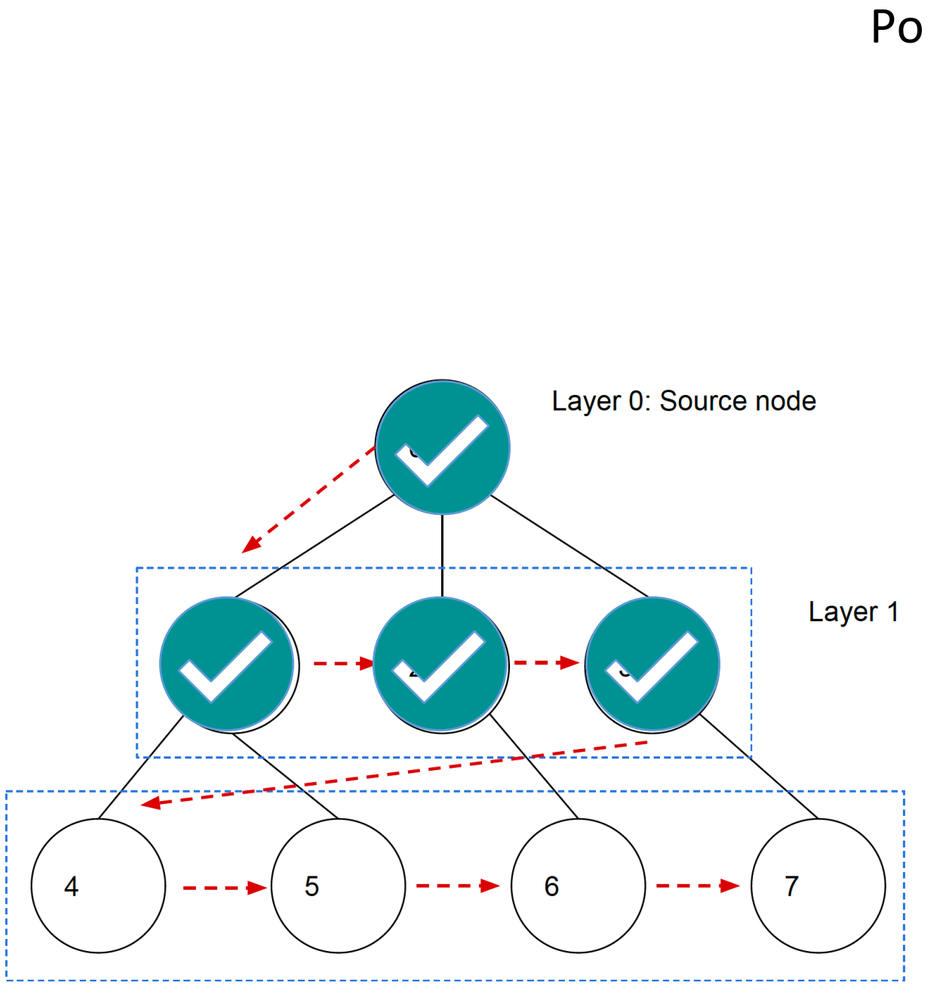


Figure 3. BFS Algorithm traversals with illustrations

BFS algorithm traversals with illustrations



Pop/Dequeue from here

Insert/Enqueue from here

Iter 4

0	1	2				

Visited

3	4	5	6			

Queue Q

Iter 5

0	1	2	3			

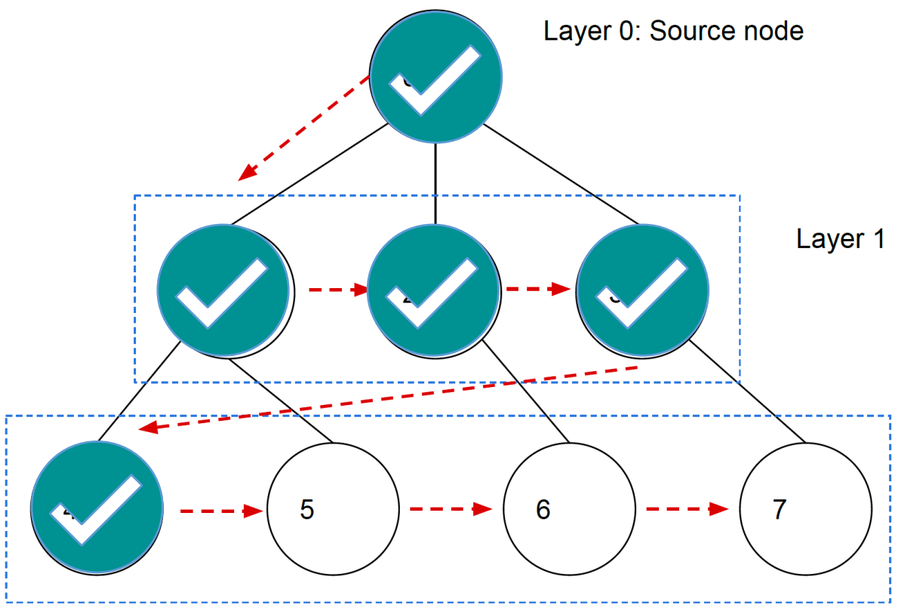
Visited

4	5	6	7			

Queue Q

Figure 3. BFS Algorithm traversals with illustrations

BFS algorithm traversals with illustrations



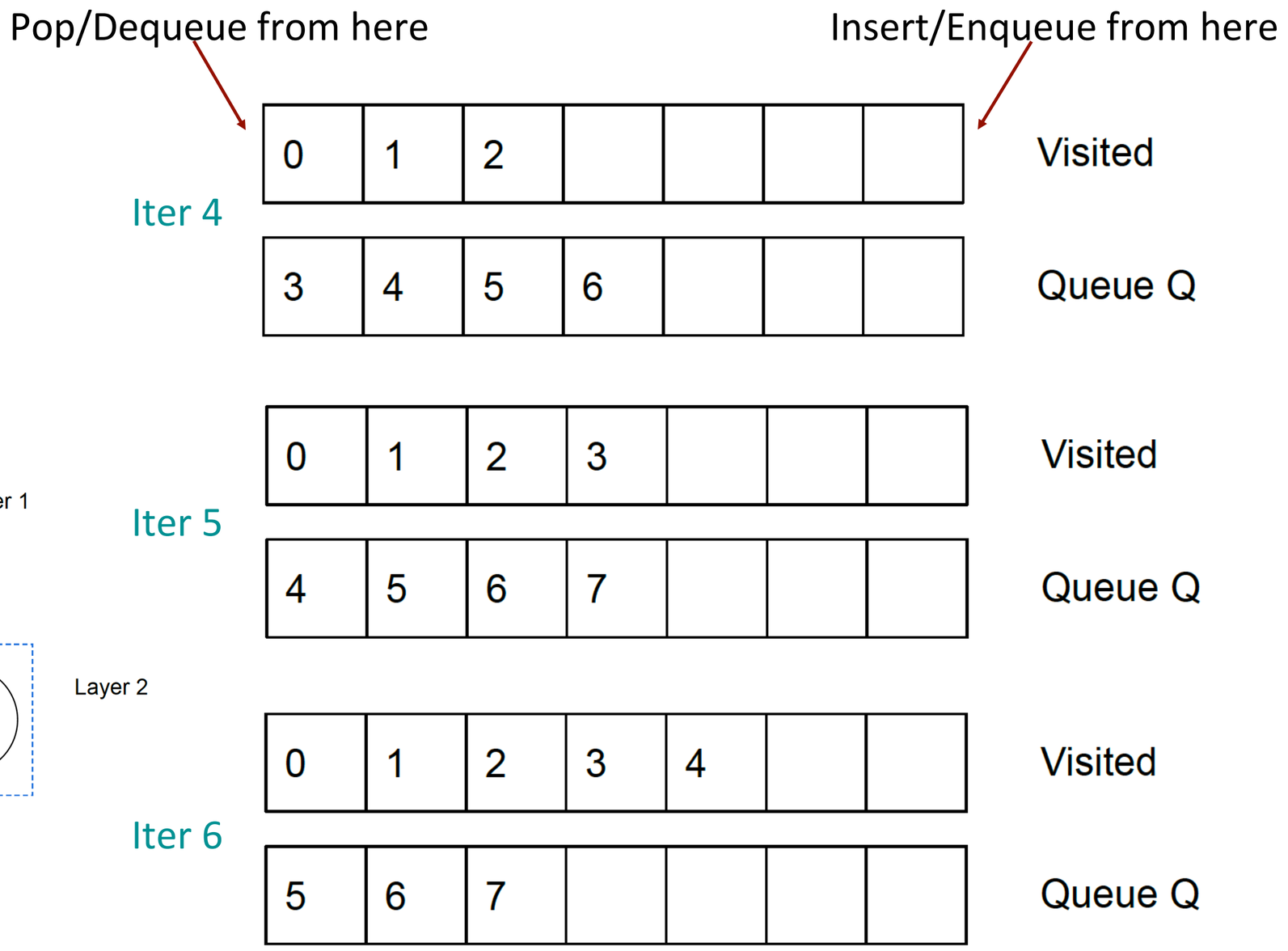
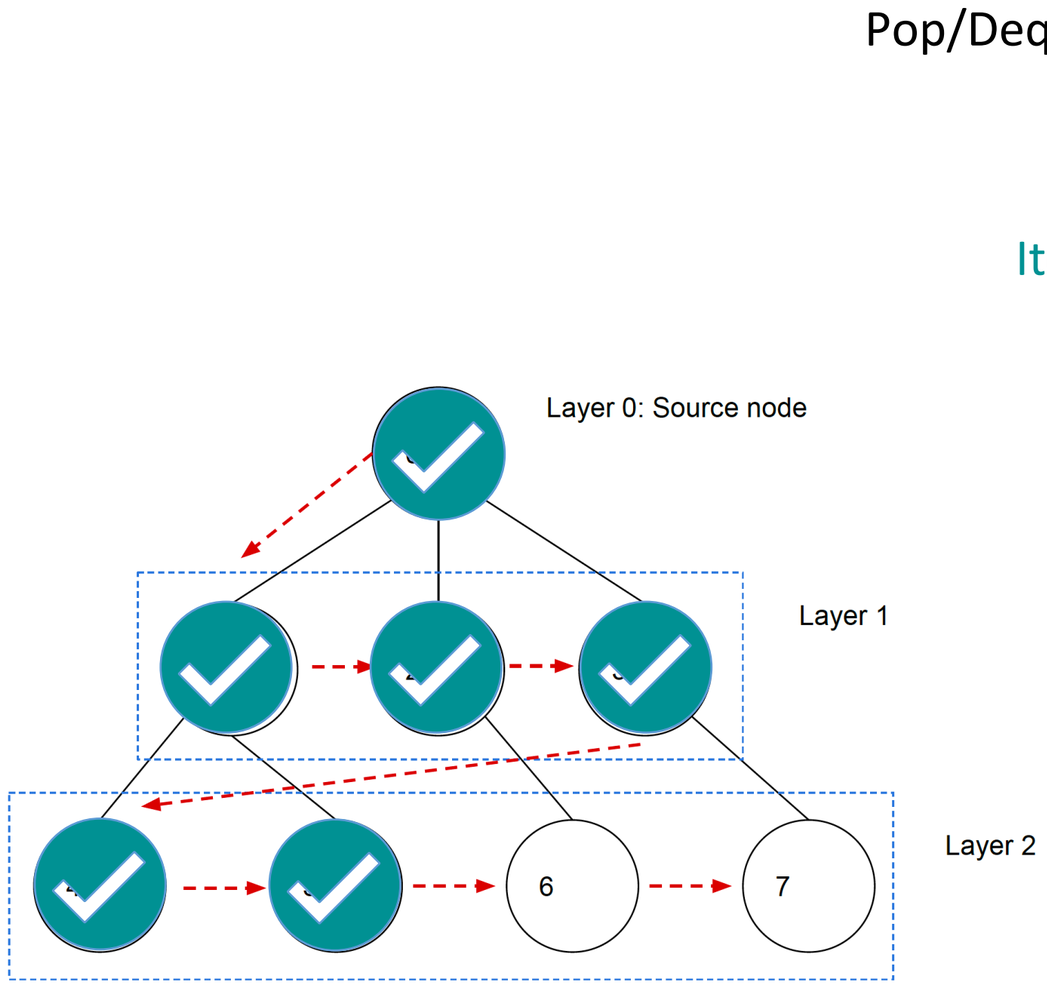


Figure 3. BFS Algorithm traversals with illustrations

BFS algorithm traversals with illustrations



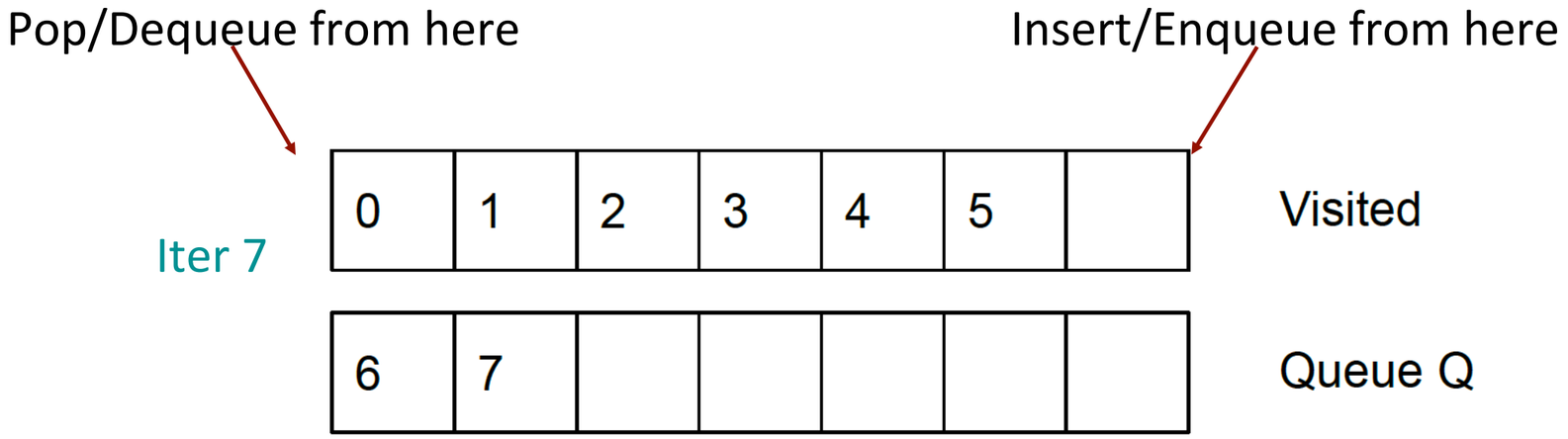
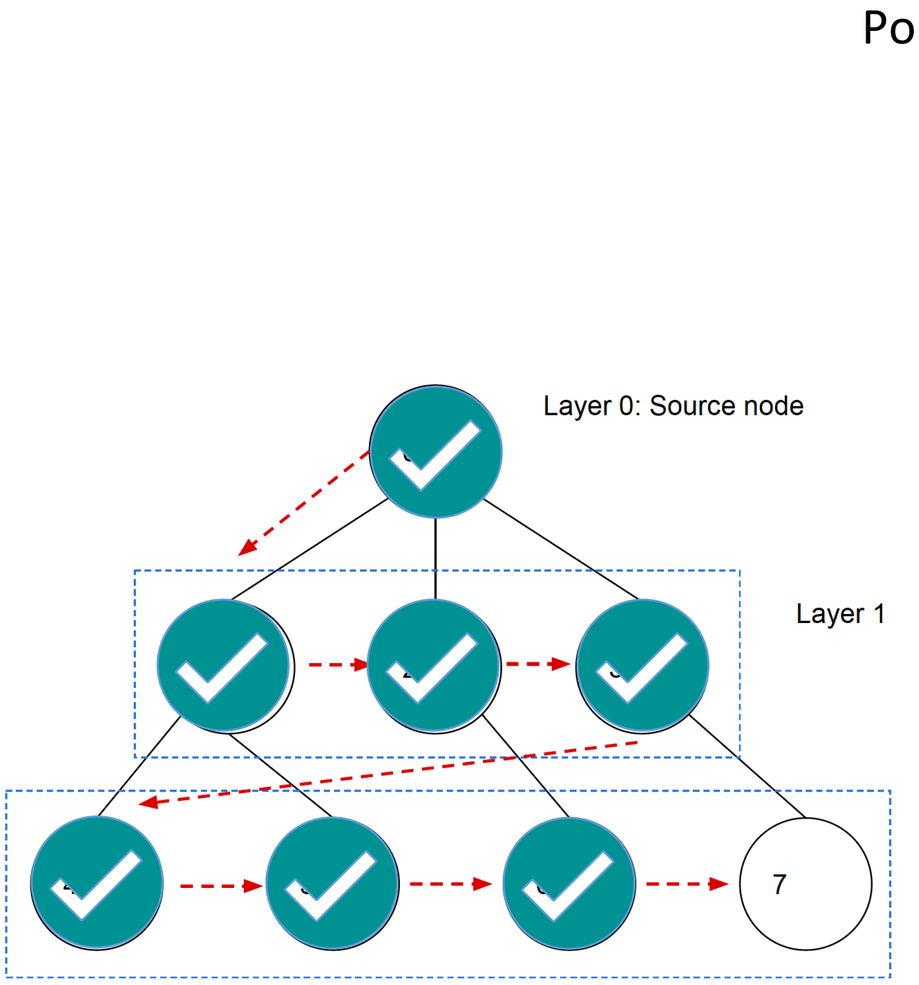


Figure 3. BFS Algorithm traversals with illustrations

BFS algorithm traversals with illustrations



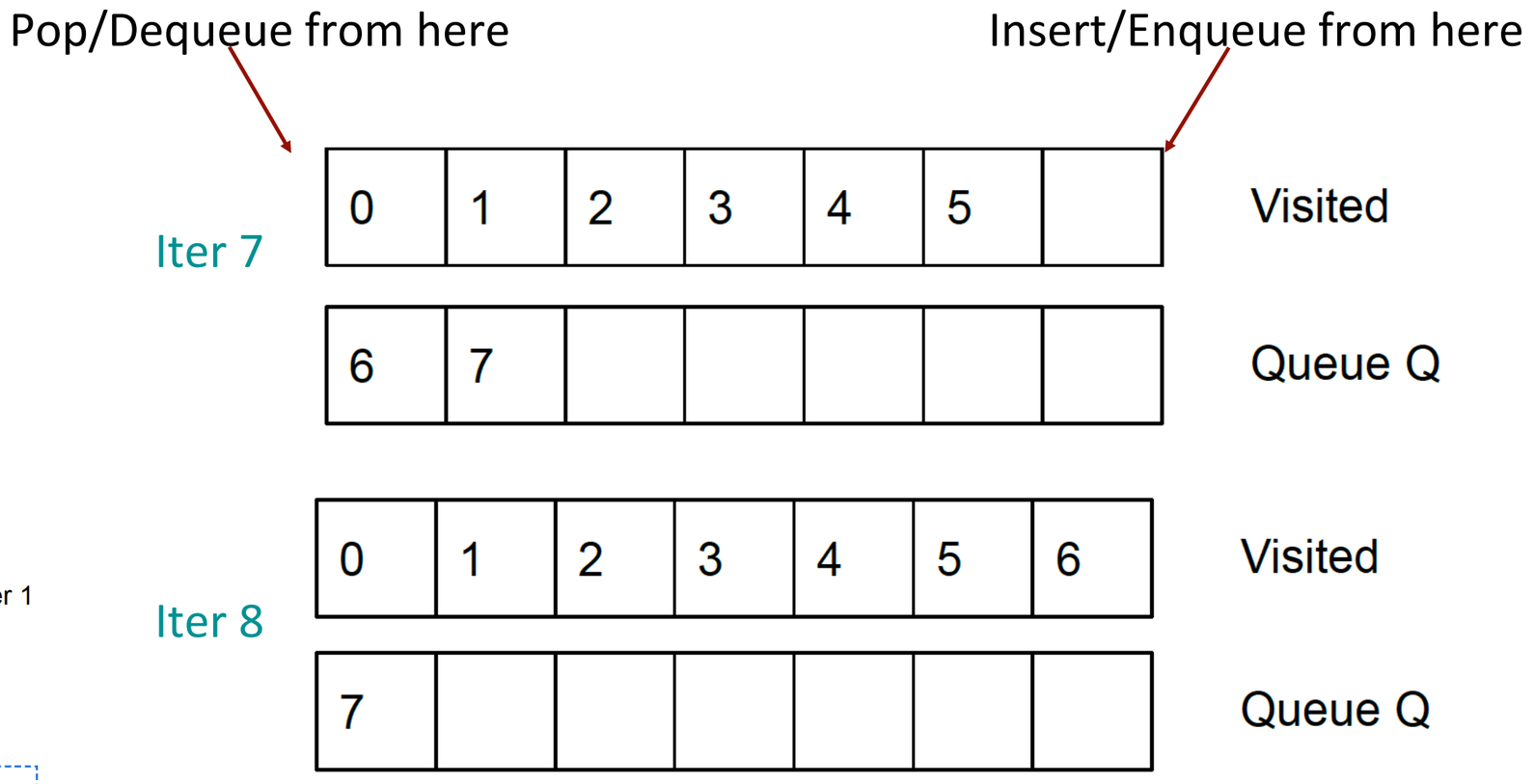
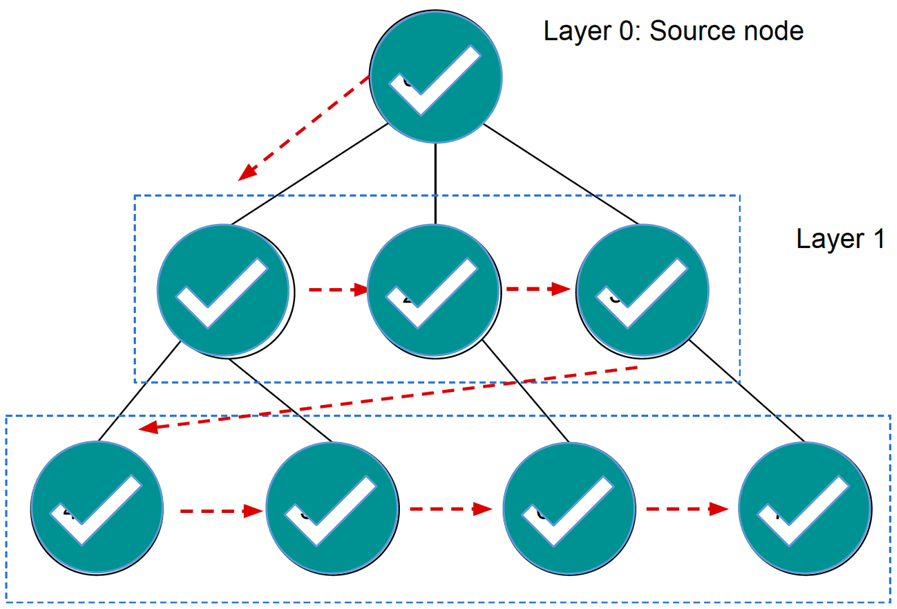


Figure 3. BFS Algorithm traversals with illustrations

BFS algorithm traversals with illustrations



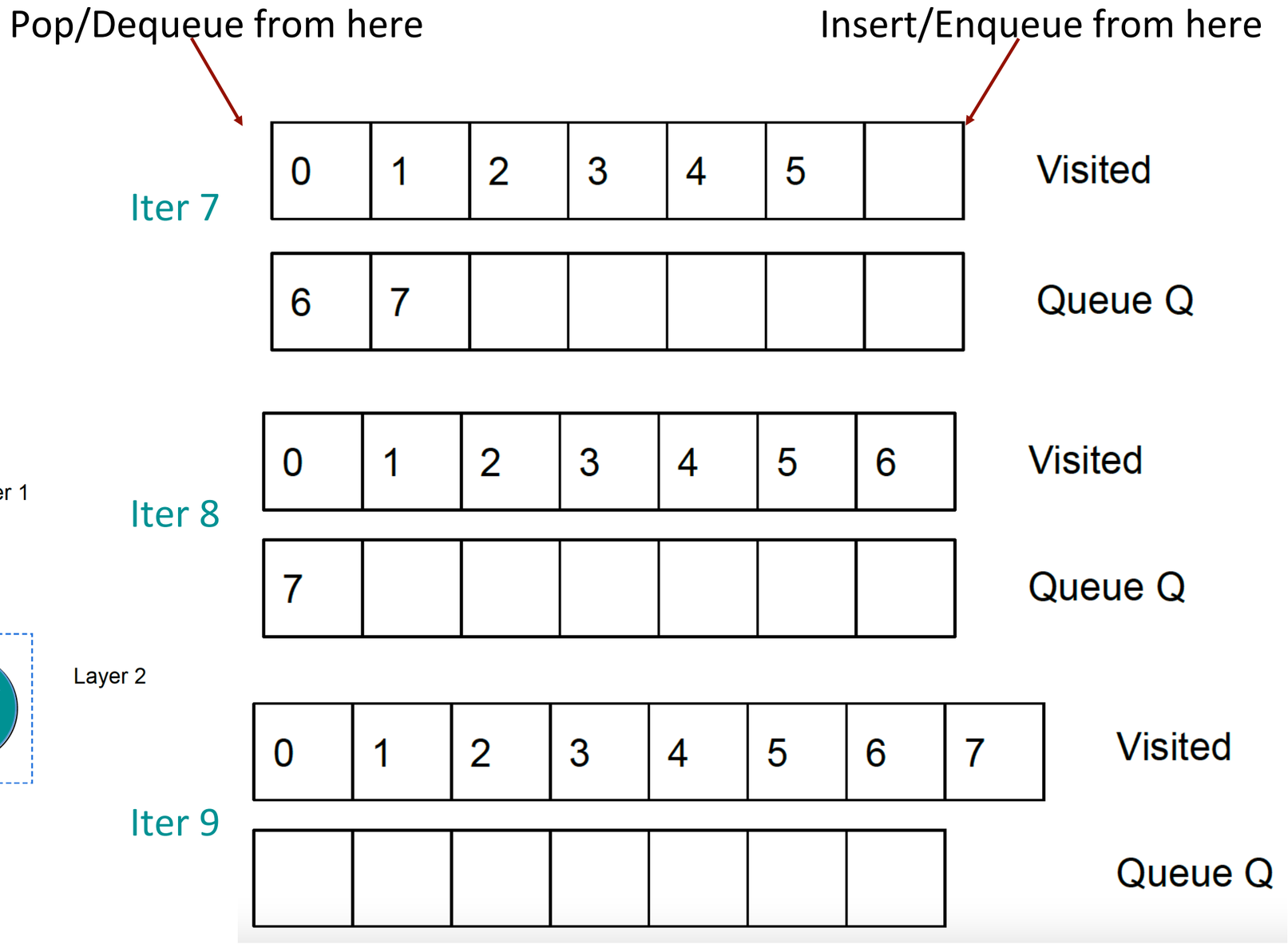


Figure 3. BFS Algorithm traversals with illustrations

PART II. How to traverse a graph?

Depth First Search (DFS) traversing algorithm: traverse a graph **branch** wise.

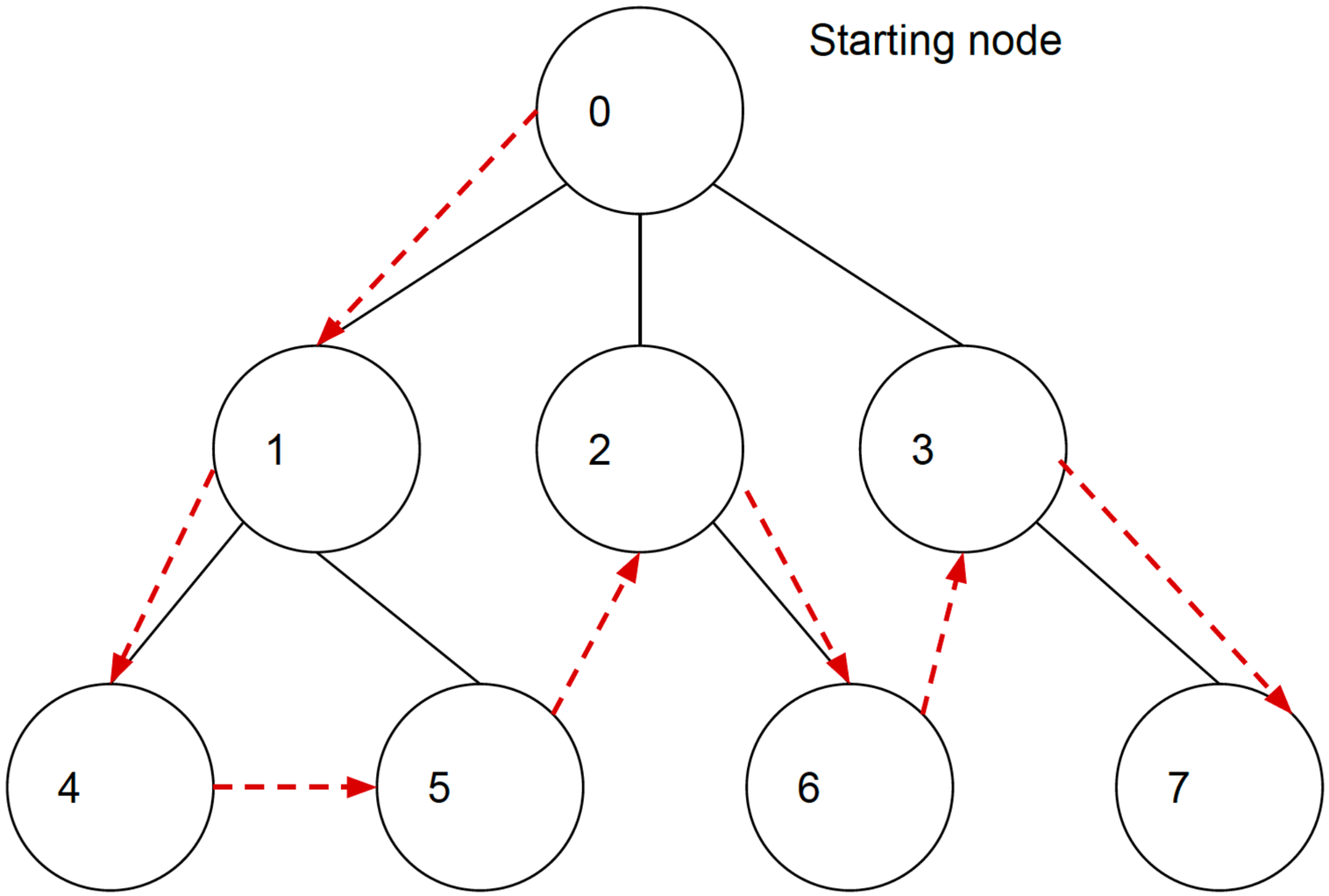


Figure 1. DFS Algorithm

Depth First Search algorithm pseudocode

Depth First Search (DFS) traversing algorithm: traverse a graph **branch** wise.

```
DFS(Graph G, Vertex V)
    Mark vertex V as visited
    For each vertex i in {neighbor vertices  of V}
        if vertex i is not visited before
            DFS(Graph G, Vertex i)
```

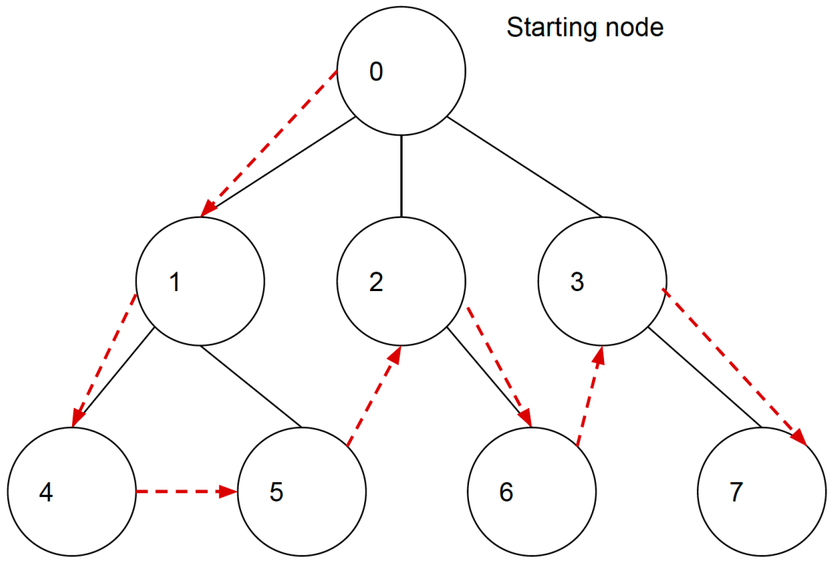


Figure 2. DFS Algorithm pseudocode

DFS algorithm traversals with illustrations

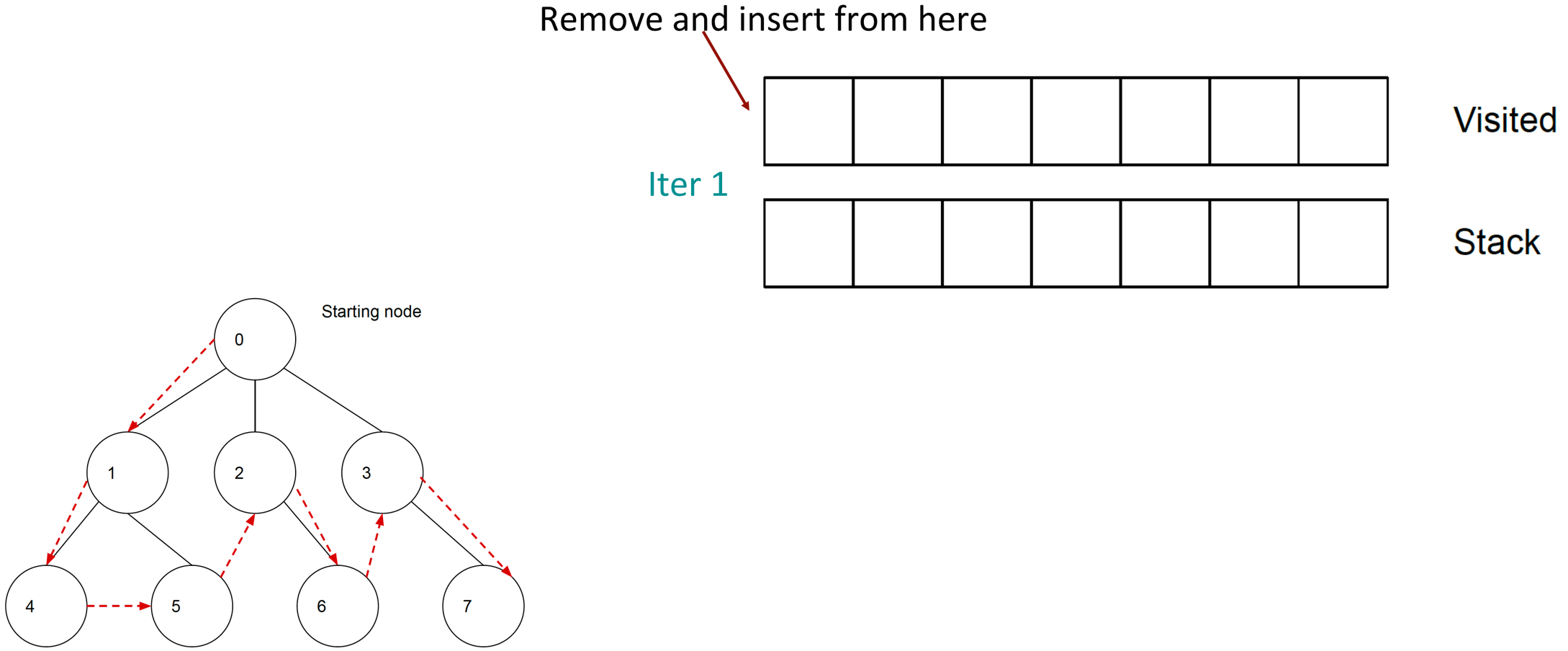
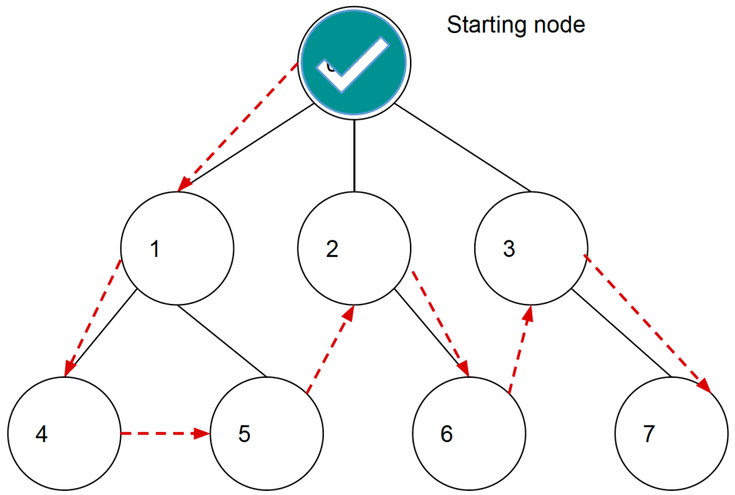


Figure 3. DFS Algorithm traversals with illustrations

DFS algorithm traversals with illustrations



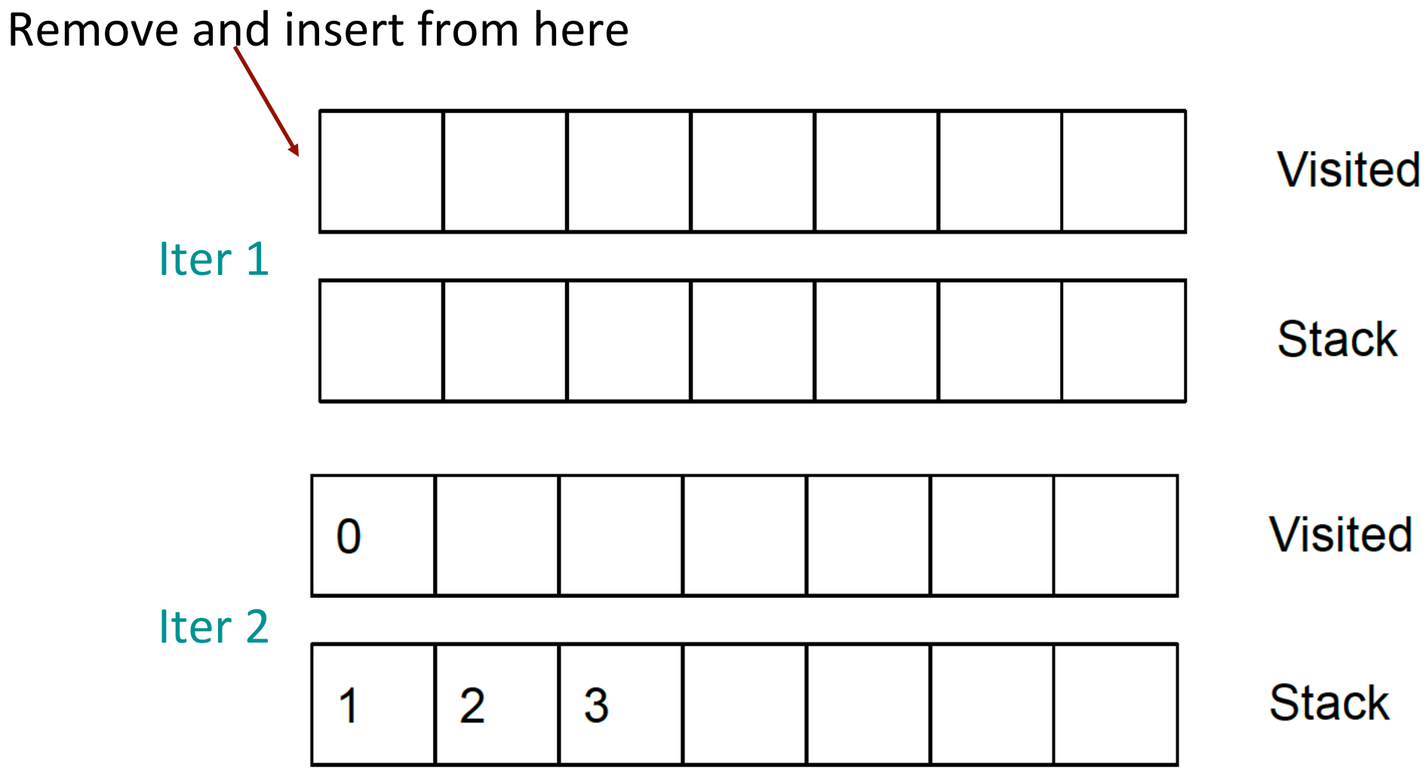
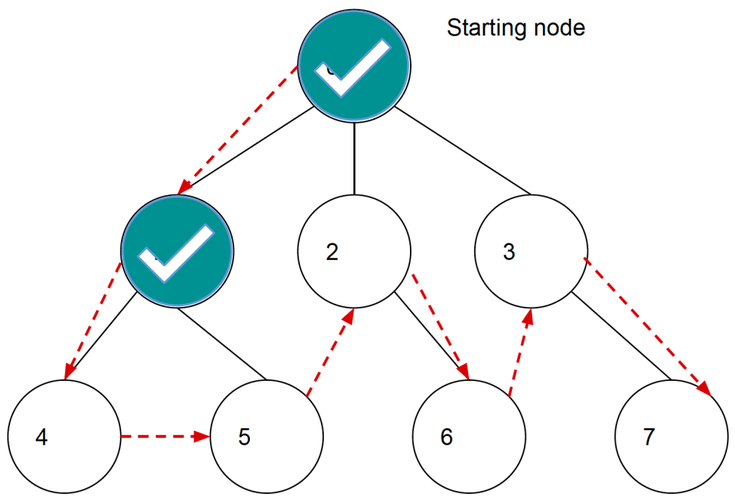


Figure 3. DFS Algorithm traversals with illustrations

DFS algorithm traversals with illustrations



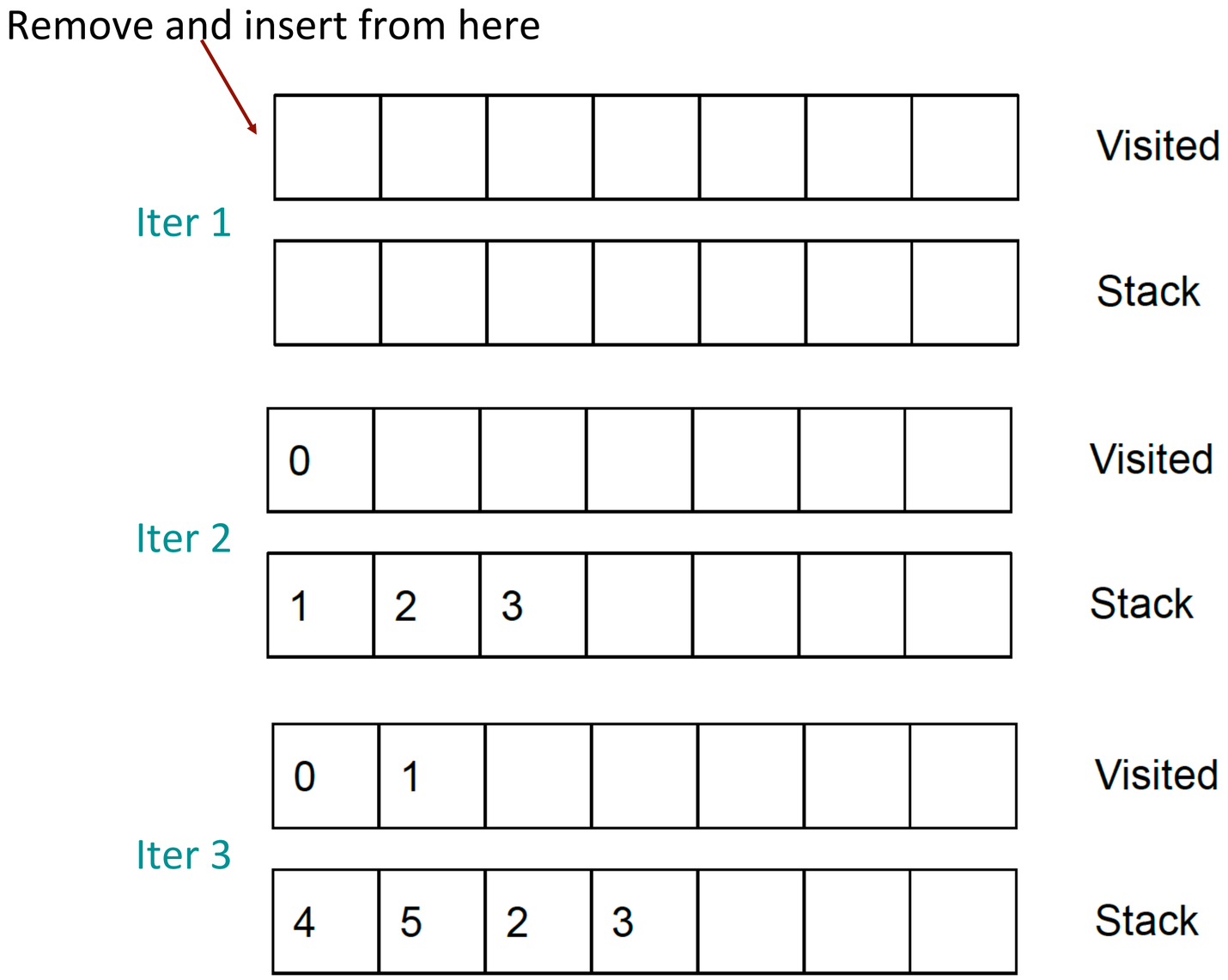
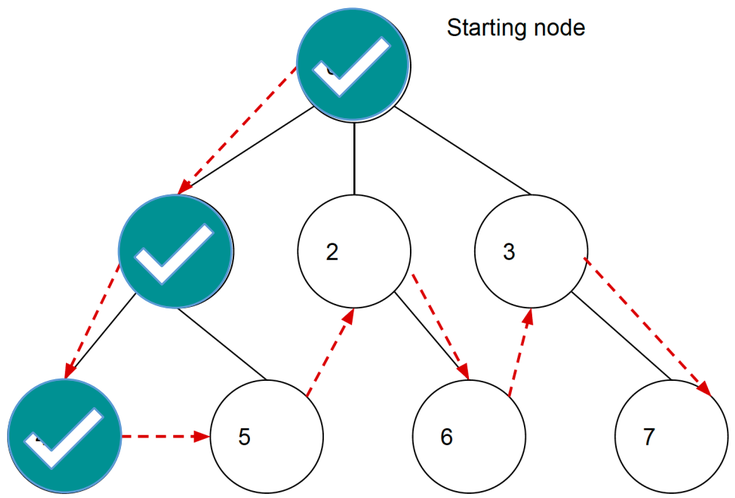


Figure 3. DFS Algorithm traversals with illustrations

DFS algorithm traversals with illustrations



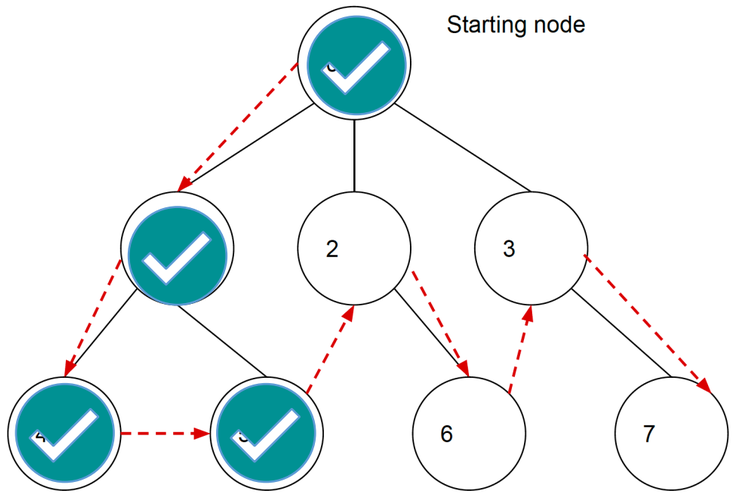
Remove and insert from here

Iter 4

0	1	4					Visited
5	2	3					Stack

Figure 3. DFS Algorithm traversals with illustrations

DFS algorithm traversals with illustrations



Remove and insert from here

Iter 4

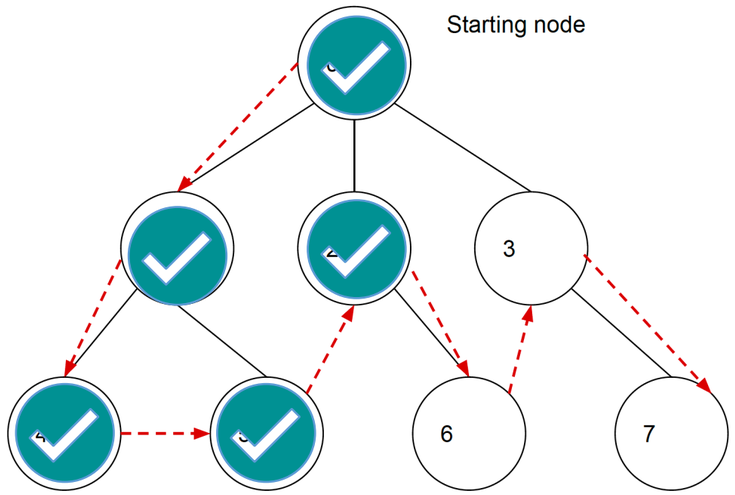
0	1	4					Visited
5	2	3					Stack

Iter 5

0	1	4	5				Visited
2	3						Stack

Figure 3. DFS Algorithm traversals with illustrations

DFS algorithm traversals with illustrations



Remove and insert from here

Iter 4

0	1	4					Visited
5	2	3					Stack

Iter 5

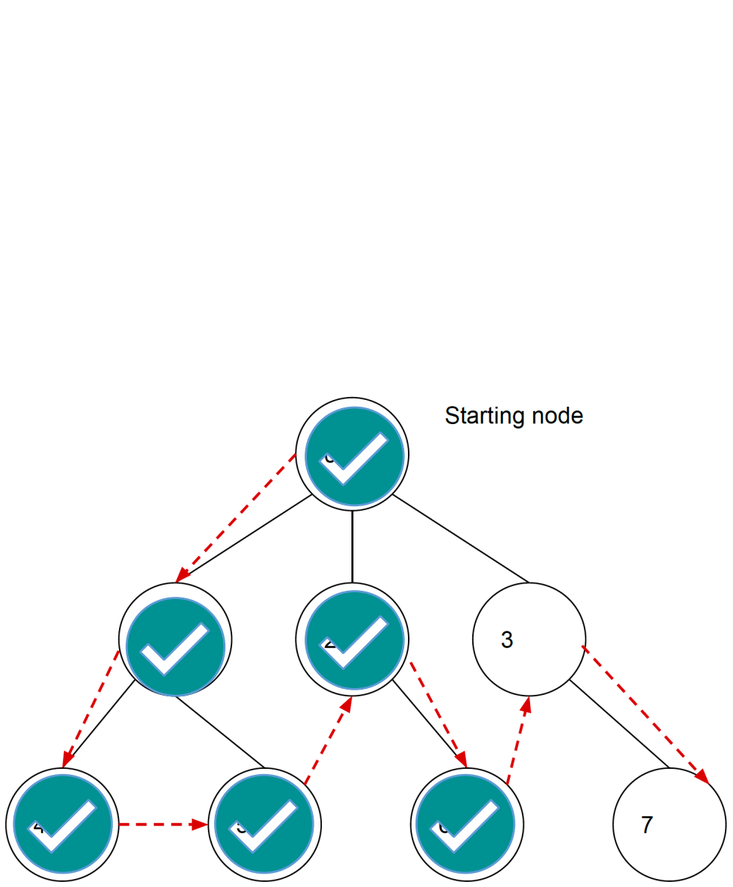
0	1	4	5				Visited
2	3						Stack

Iter 6

0	1	4	5	2			Visited
6	3						Stack

Figure 3. DFS Algorithm traversals with illustrations

DFS algorithm traversals with illustrations



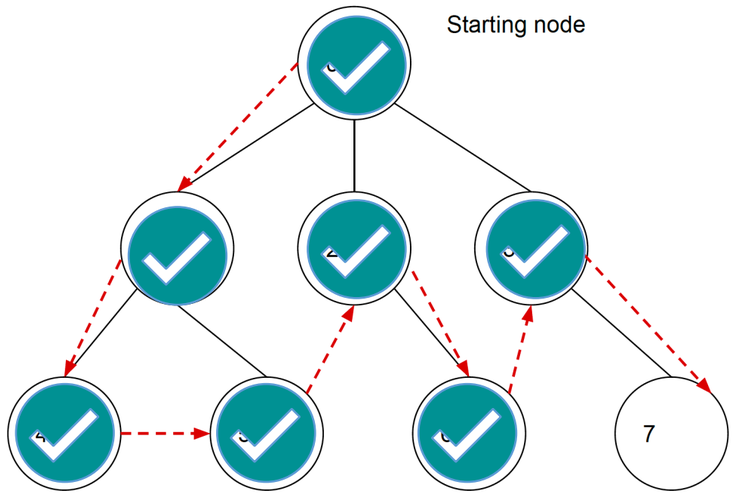
Remove and insert from here

Iter 7

0	1	4	5	2	6		Visited
3							Stack

Figure 3. DFS Algorithm traversals with illustrations

DFS algorithm traversals with illustrations



Remove and insert from here

Iter 7

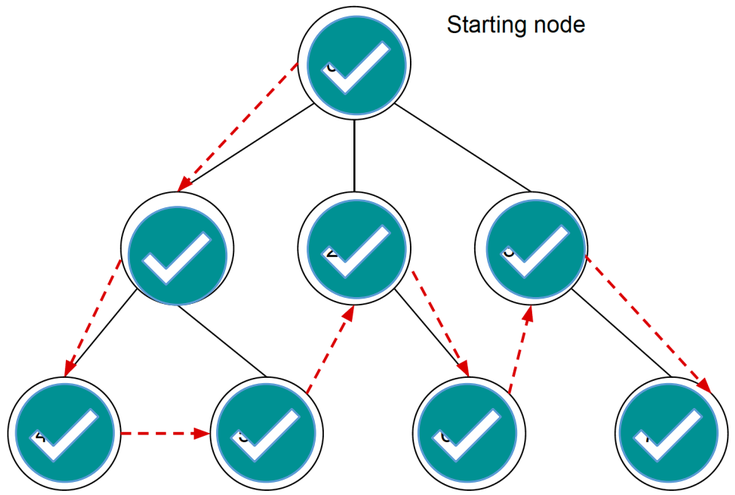
0	1	4	5	2	6		Visited
3							Stack

Iter 8

0	1	4	5	2	6	3	Visited
7							Stack

Figure 3. DFS Algorithm traversals with illustrations

DFS algorithm traversals with illustrations



Remove and insert from here

Iter 7

0	1	4	5	2	6		Visited
3							Stack

Iter 8

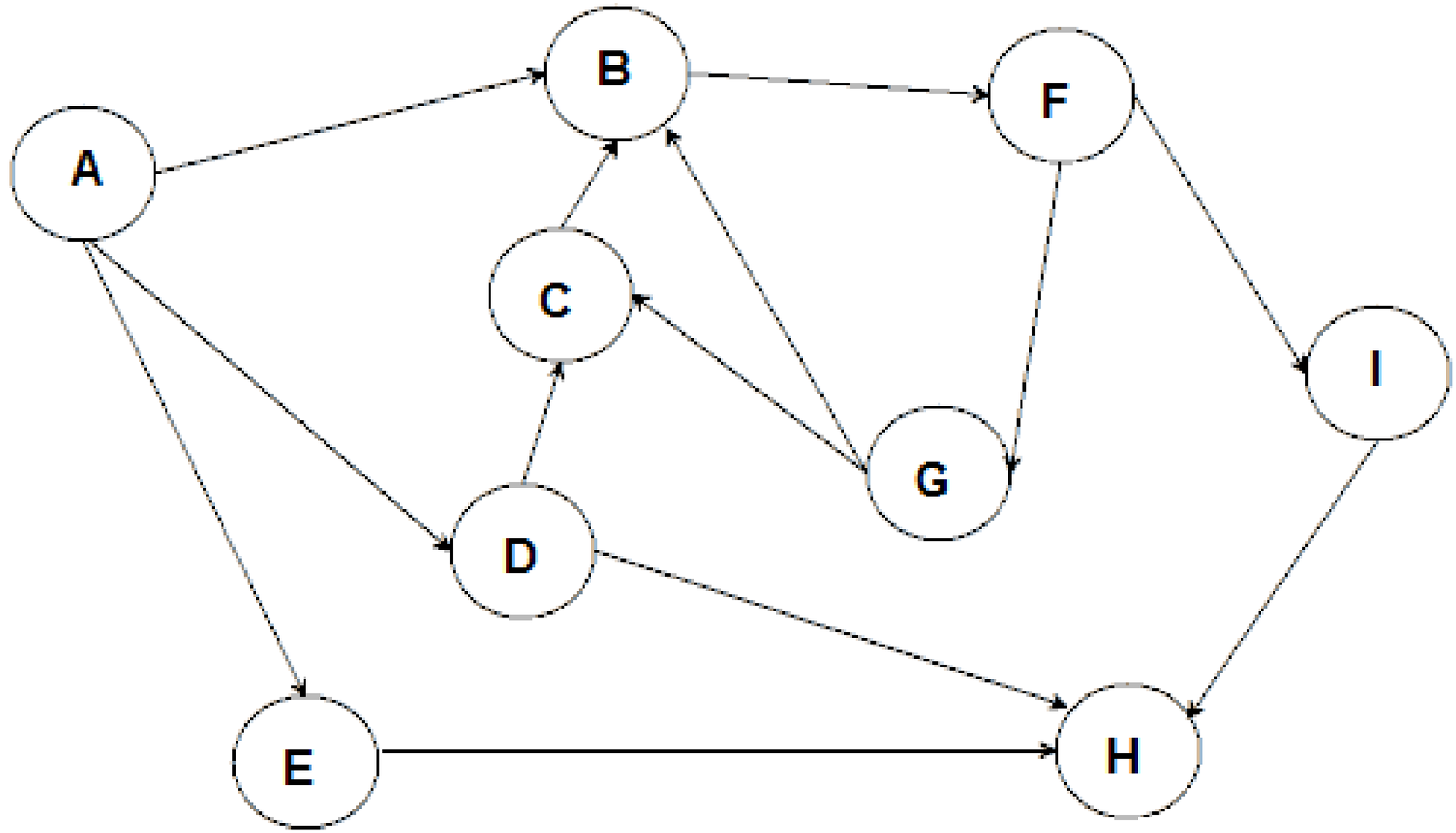
0	1	4	5	2	6	3	Visited
7							Stack

Iter 9

0	1	4	5	2	6	3	7	Visited
								Stack

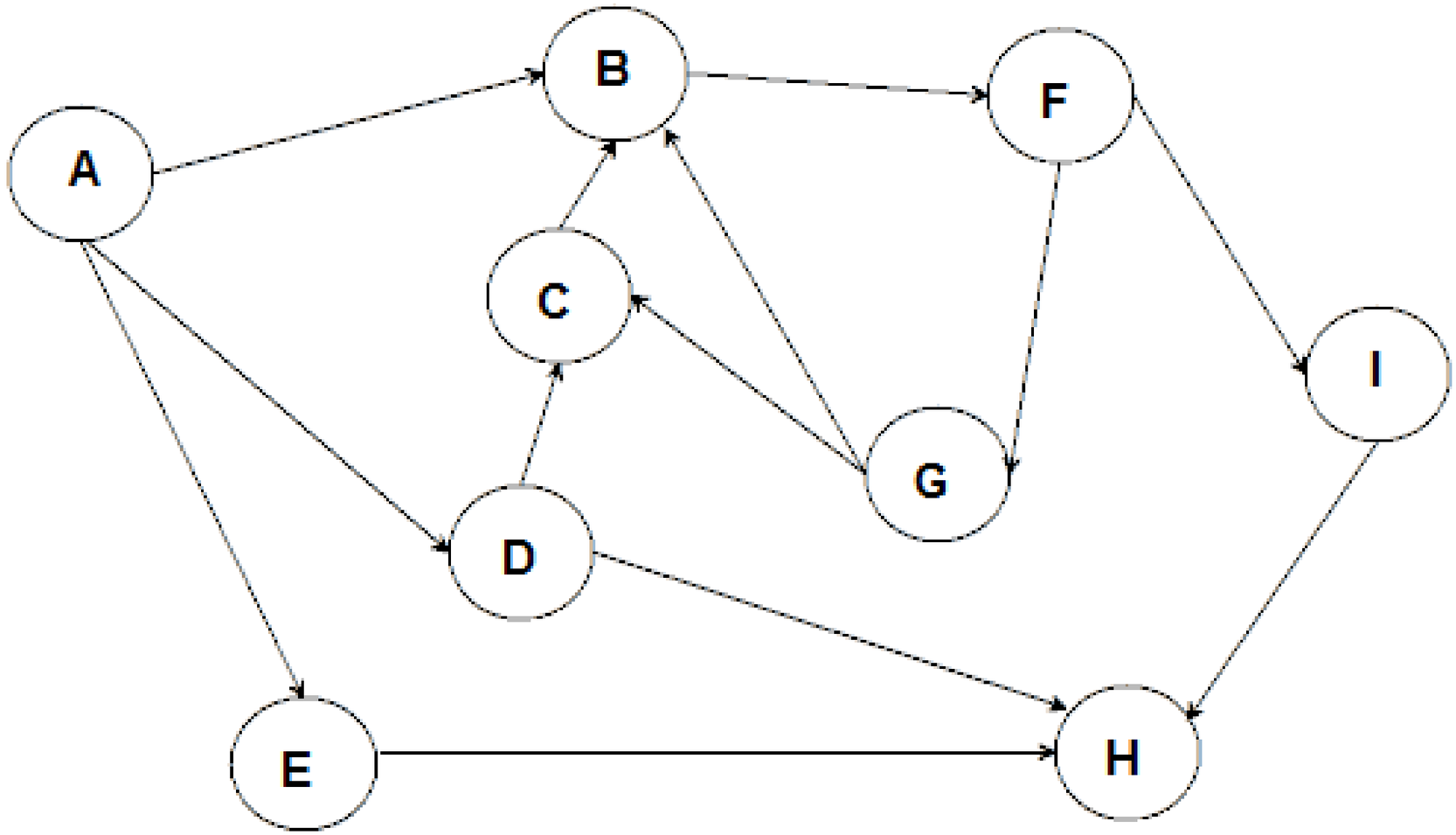
Figure 3. DFS Algorithm traversals with illustrations

Example problems for BFS & DFS



Start from A **BFS** **DFS**

Example problems for BFS & DFS



Start from A

BFS

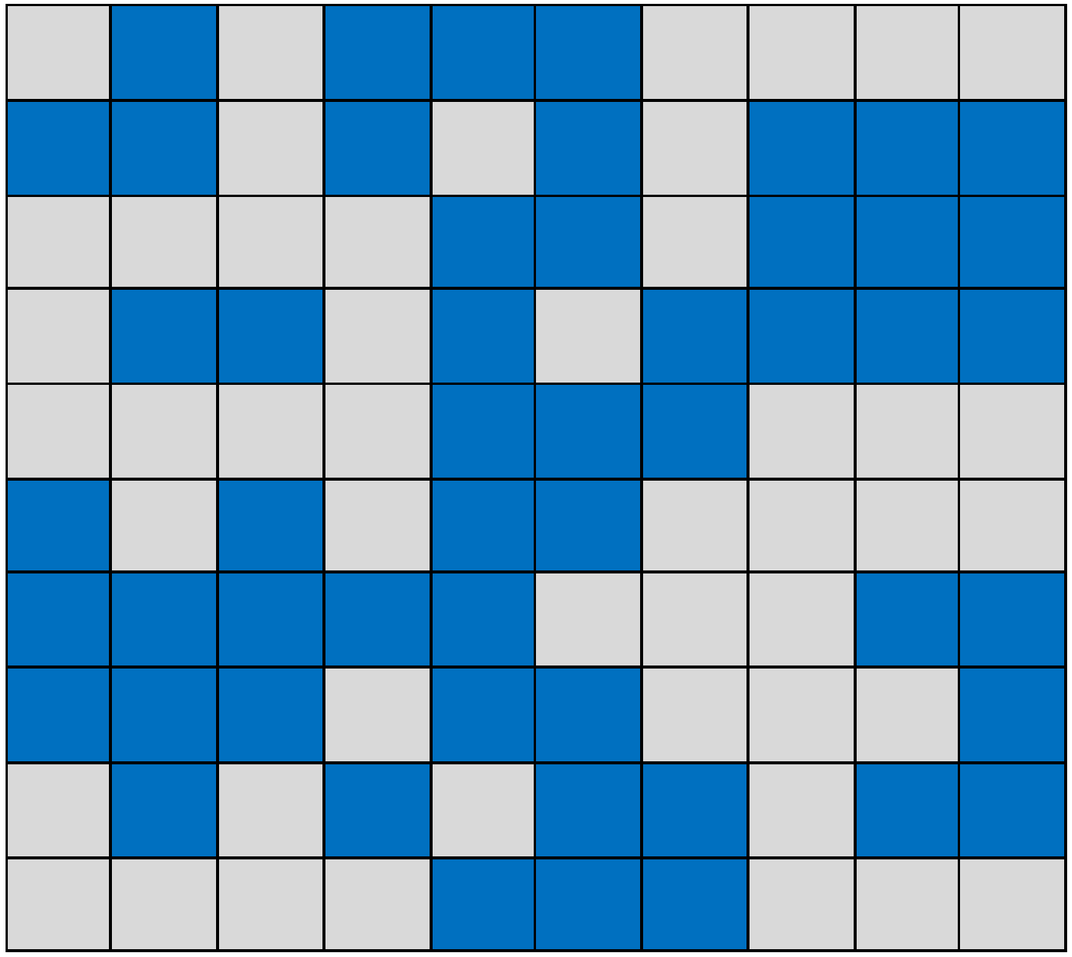
A -> B -> D -> E -> F -> C -> H -> G -> I

DFS

A -> B -> F -> G -> C
Backtrack to F
I -> H
Backtrack to A
D
Backtrack to A
E

Example problems for BFS & DFS

Given a binary matrix where 0 represents water and 1 represents land, and connected ones form an island, count the total islands.



Example problems for BFS & DFS

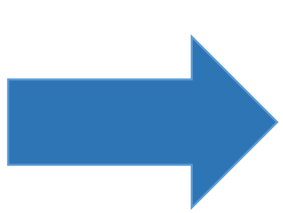
Given a binary matrix where 0 represents water and 1 represents land, and connected ones form an island, count the total islands.

1		2				3	3	3	3
		2		2		3			
2	2	2	2			3			
2			2		3				
2	2	2	2				5	5	5
	2		2			5	5	5	5
					5	5	5		
			4			5	5	5	
4		4		4			5		
4	4	4	4				5	5	5

Example problems for BFS & DFS

Given a binary matrix where 0 represents water and 1 represents land, and connected ones form an island, count the total islands.

1		2				3	3	3	3
		2		2		3			
2	2	2	2			3			
2			2		3				
2	2	2	2				5	5	5
	2		2			5	5	5	5
					5	5	5		
			4			5	5	5	
4		4		4			5		
4	4	4	4				5	5	5



Finding the connected components in a graph

Example problems for BFS & DFS

Given a binary matrix where 0 represents water and 1 represents land, and connected ones form an island, count the total islands.

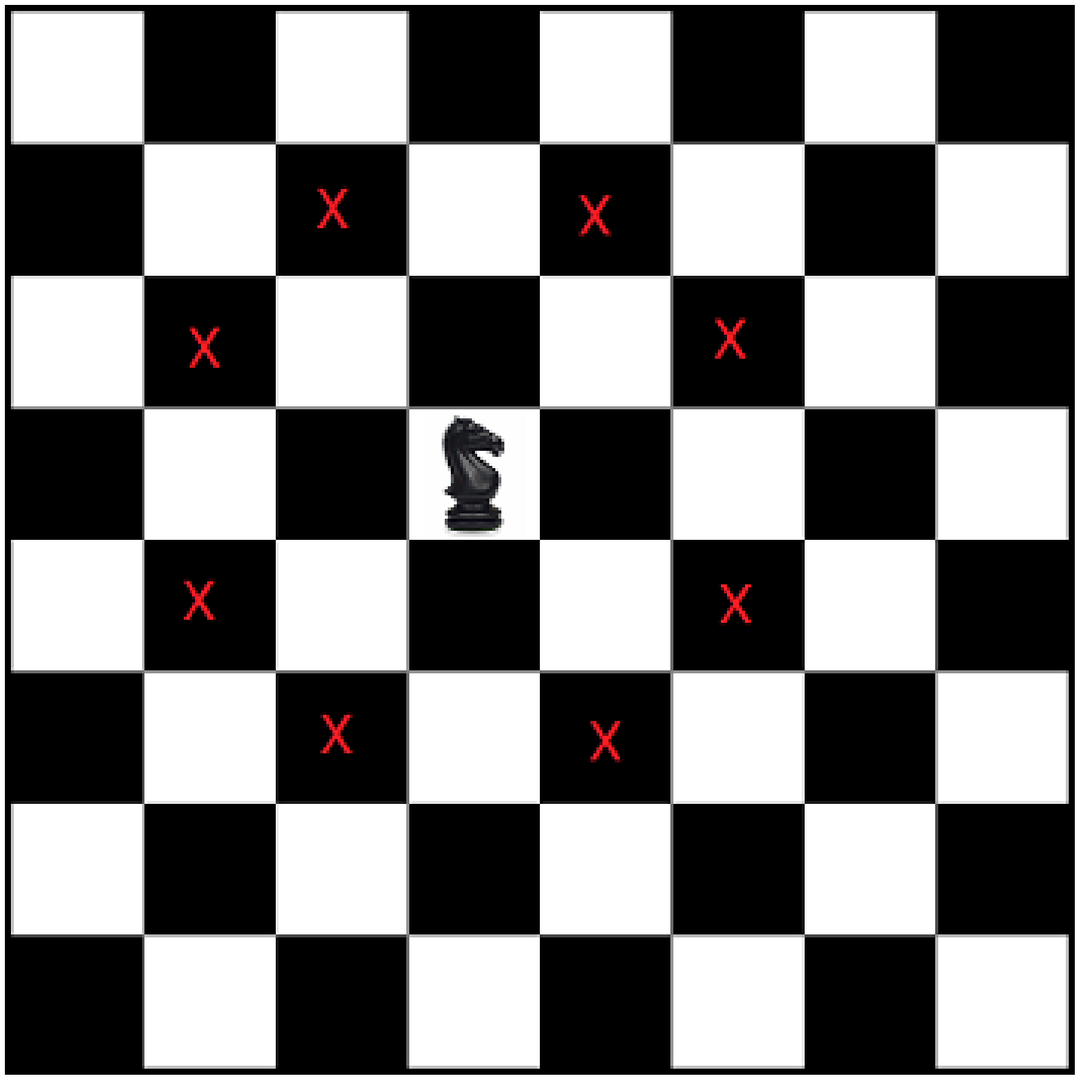
1		2				3	3	3	3
		2		2		3			
2	2	2	2			3			
2			2		3				
2	2	2	2				5	5	5
	2		2			5	5	5	5
					5	5	5		
			4			5	5	5	
4		4		4			5		
4	4	4	4				5	5	5

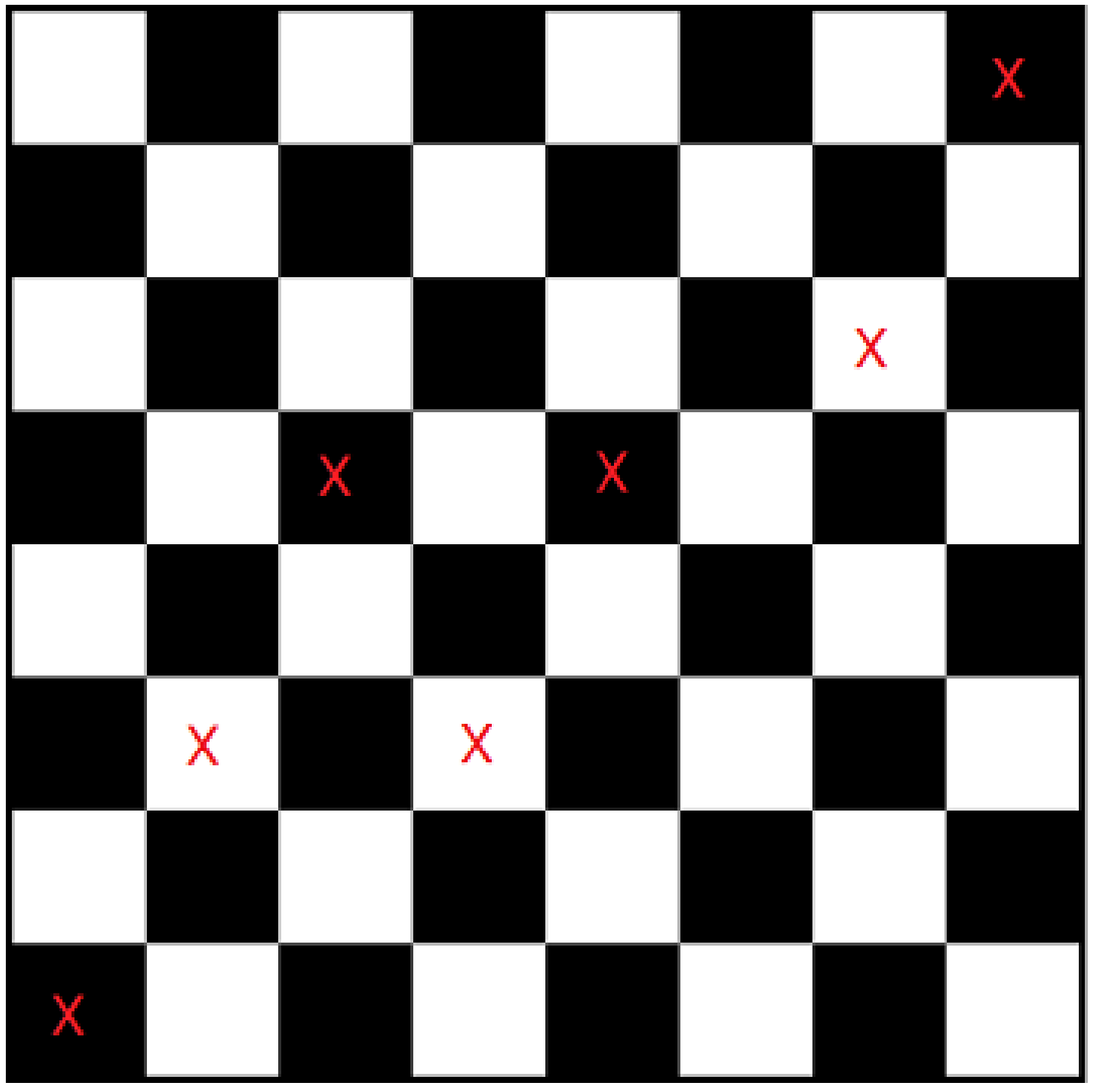
Finding the connected components in a graph

- Start BFS from unprocessed node
- Increment island count
- Each BFS traversal will mark all cells which make one island as processed.
- Problem reduces to finding the total number of BFS calls.

Example problems for BFS & DFS

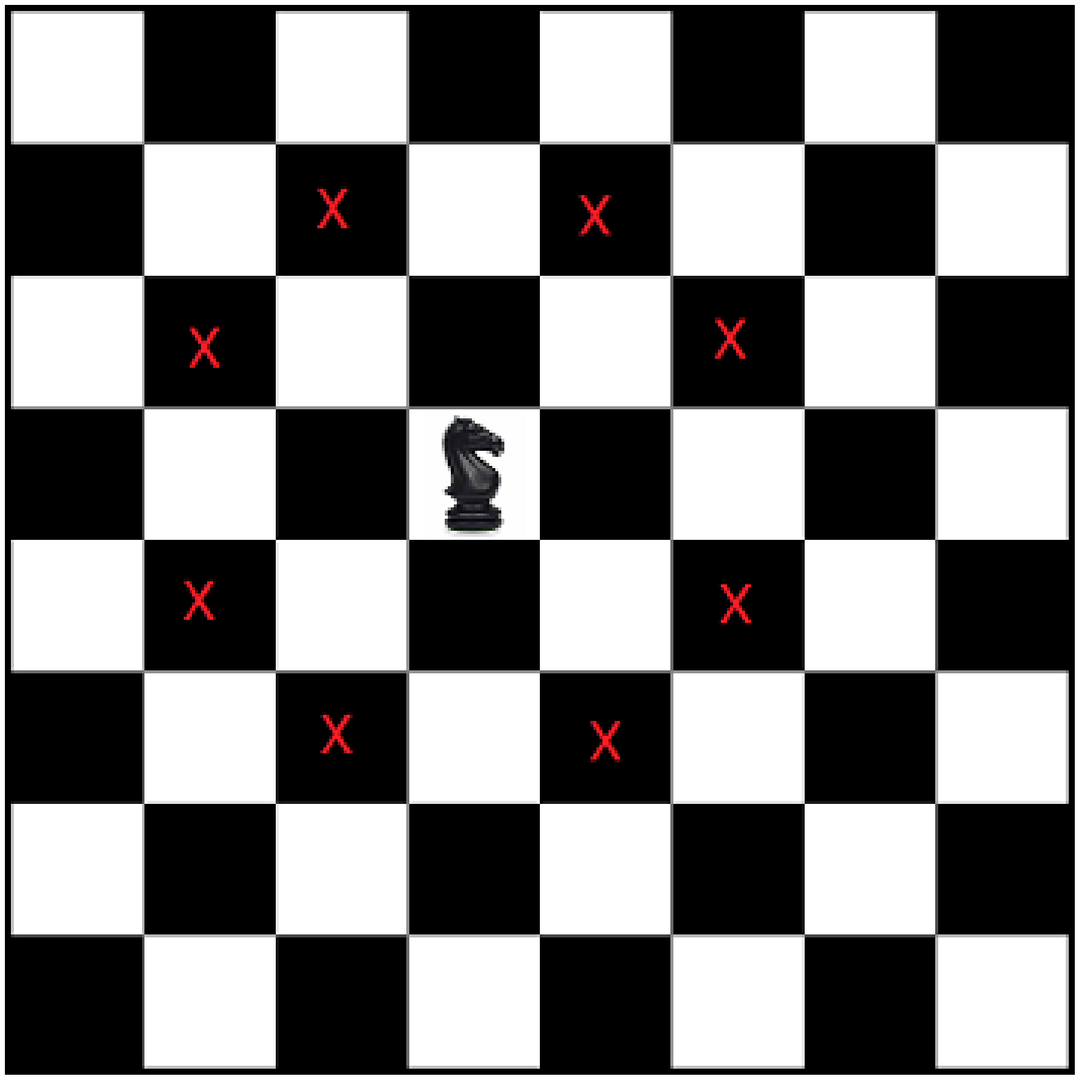
Given a chessboard, find the shortest distance (minimum number of steps) taken by a knight to reach a given destination from a given source.

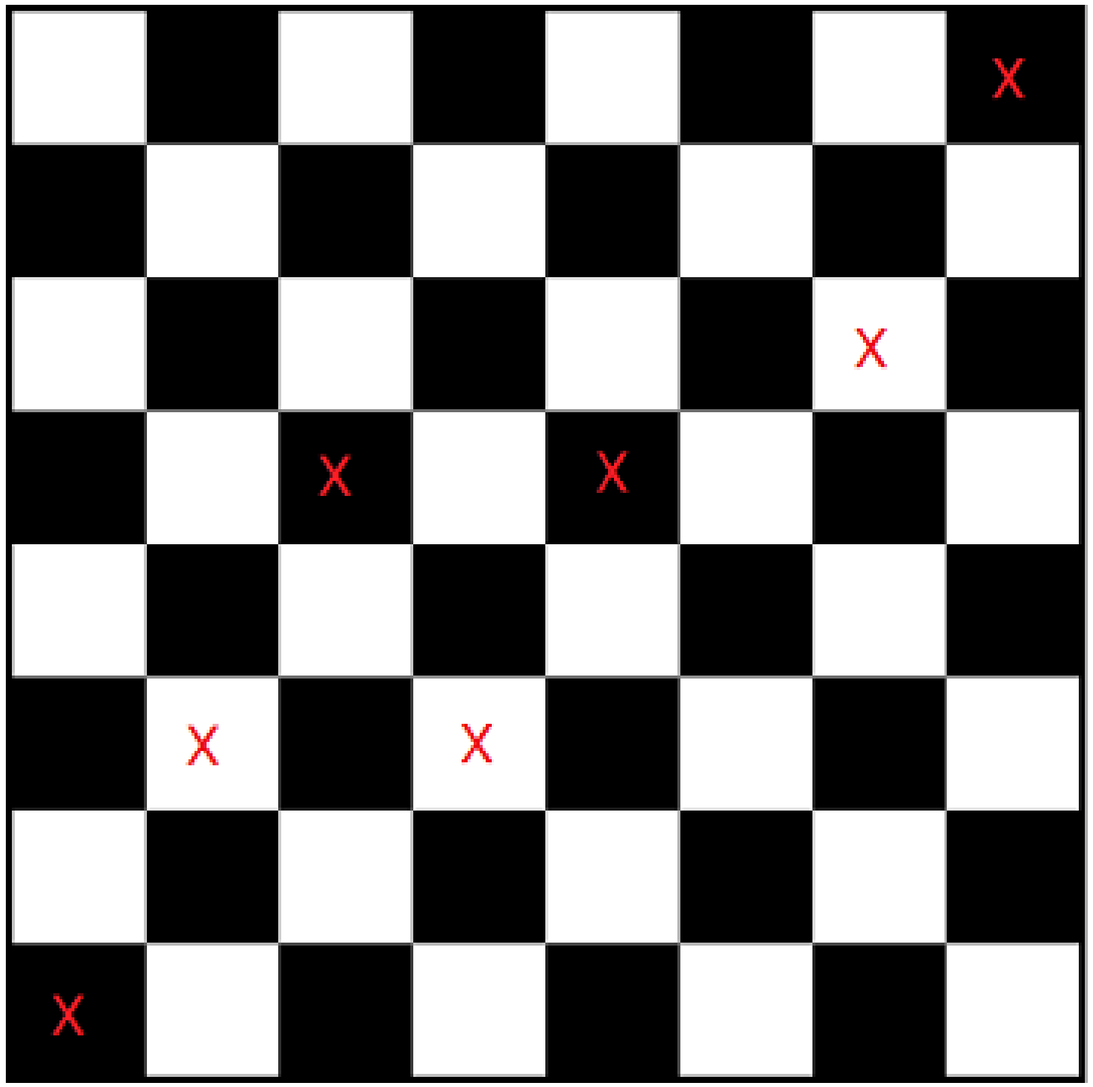




Example problems for BFS & DFS

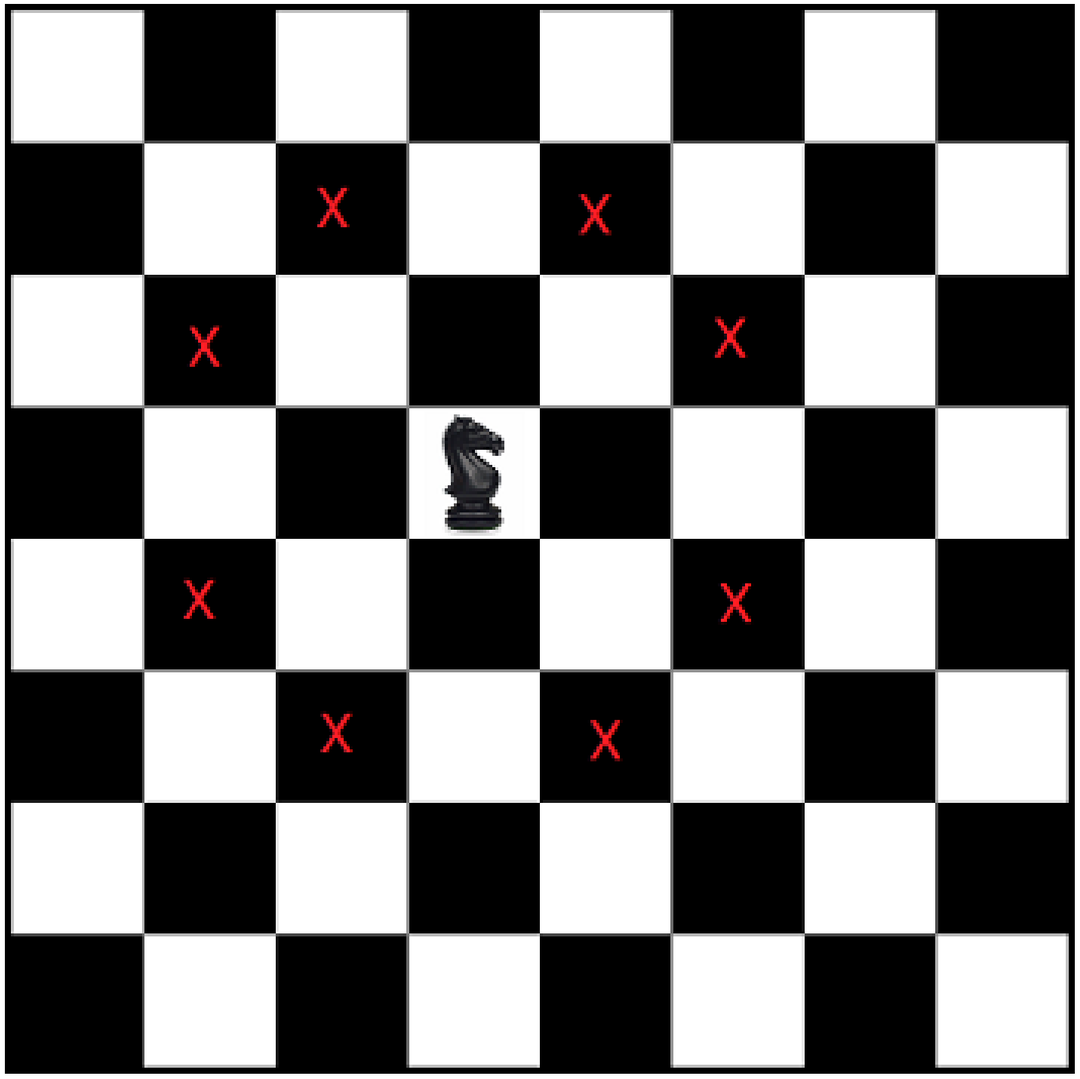
Given a chessboard, find the shortest distance (minimum number of steps) taken by a knight to reach a given destination from a given source.

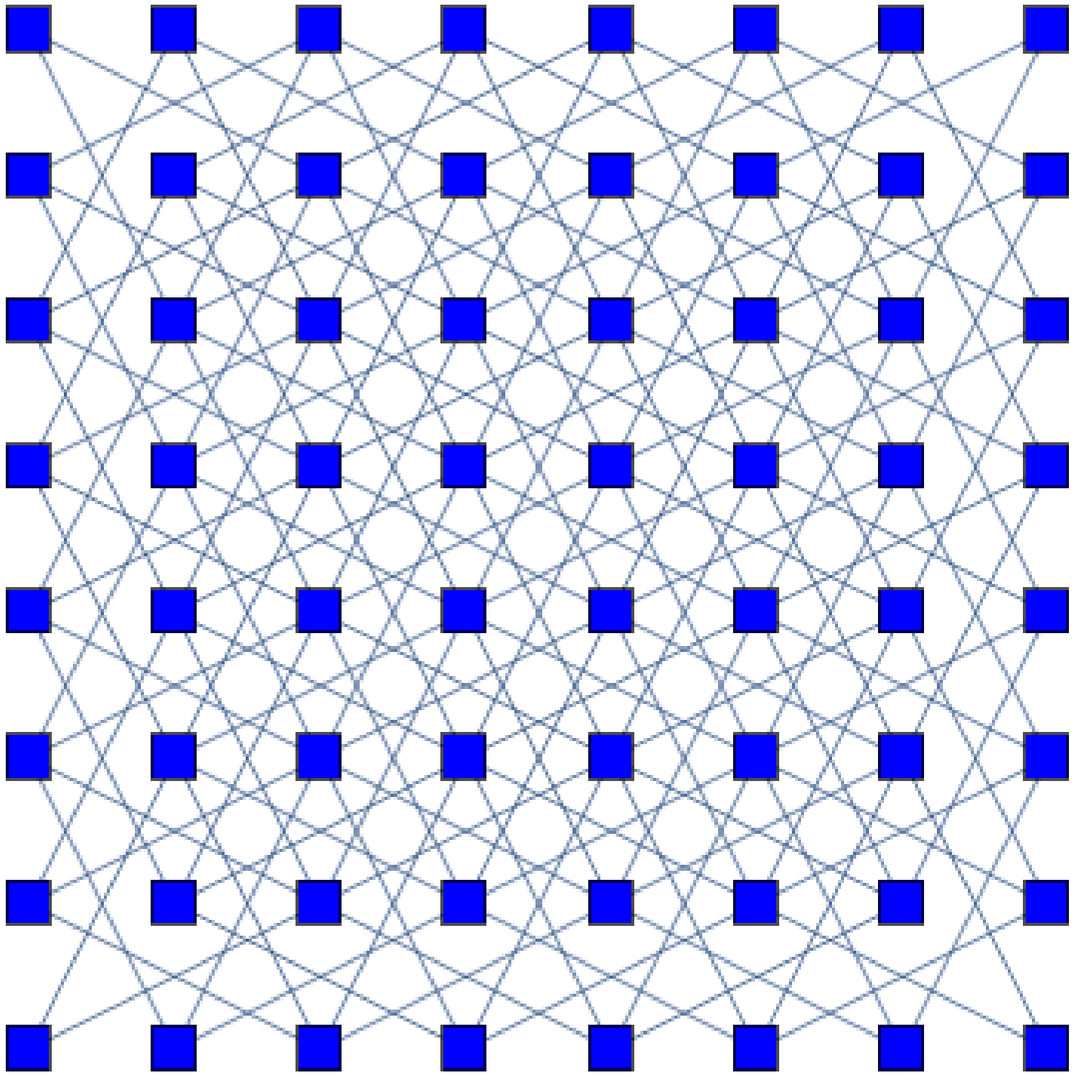




Example problems for BFS & DFS

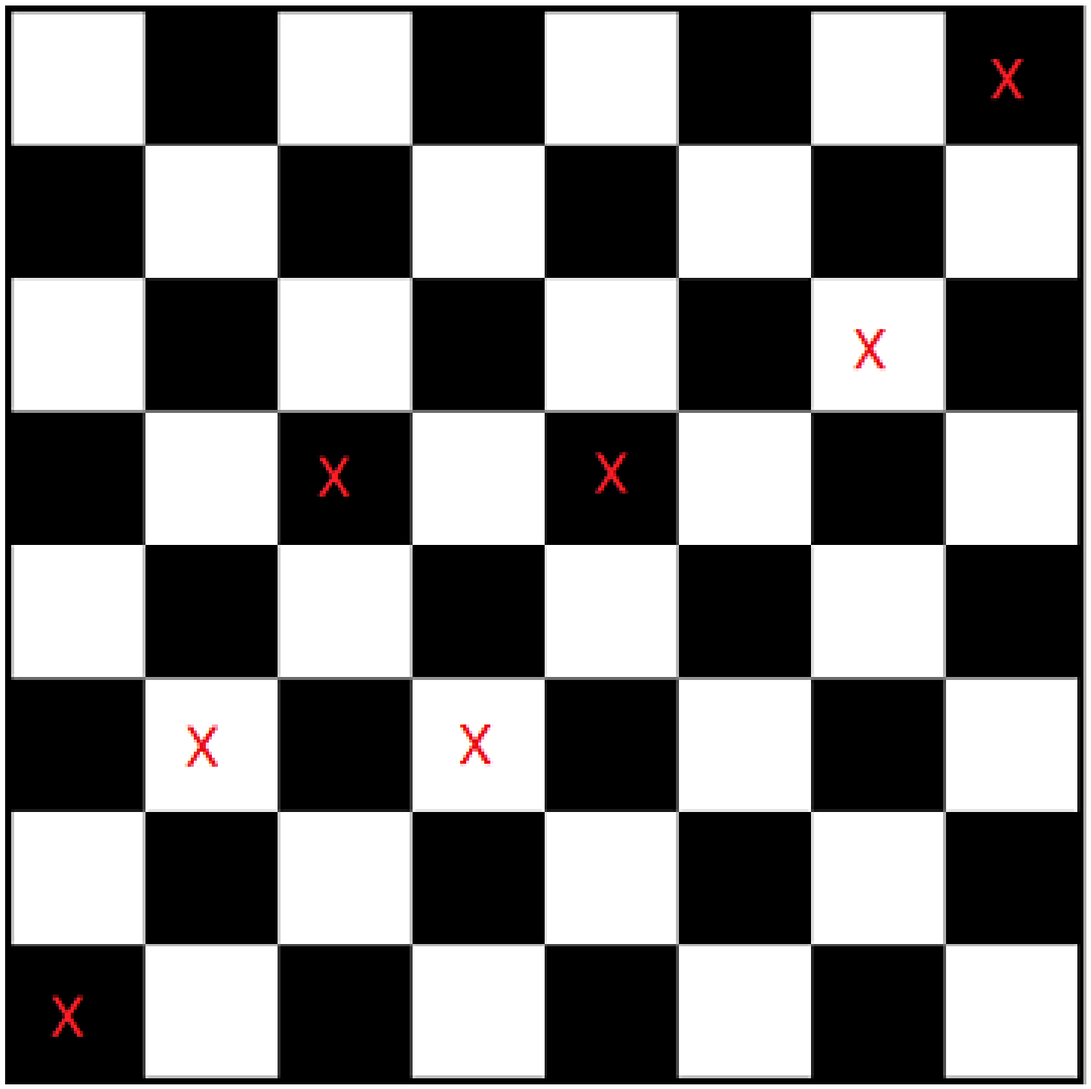
Given a chessboard, find the shortest distance (minimum number of steps) taken by a knight to reach a given destination from a given source.





Example problems for BFS & DFS

Given a chessboard, find the shortest distance (minimum number of steps) taken by a knight to reach a given destination from a given source.



1. Create an empty queue and enqueue the source cell having a distance of 0 from the source (itself).
2. Loop till queue is empty:
   2.1 Dequeue next unvisited node.
   2.2 If the popped node is the destination node, return its distance.
   2.3 Otherwise, mark the current node as visited. For each of eight possible movements for a knight, enqueue each valid movement with +1 distance (minimum distance of a given node from the source is one more than the minimum distance of parent from source).

Example problems for BFS & DFS

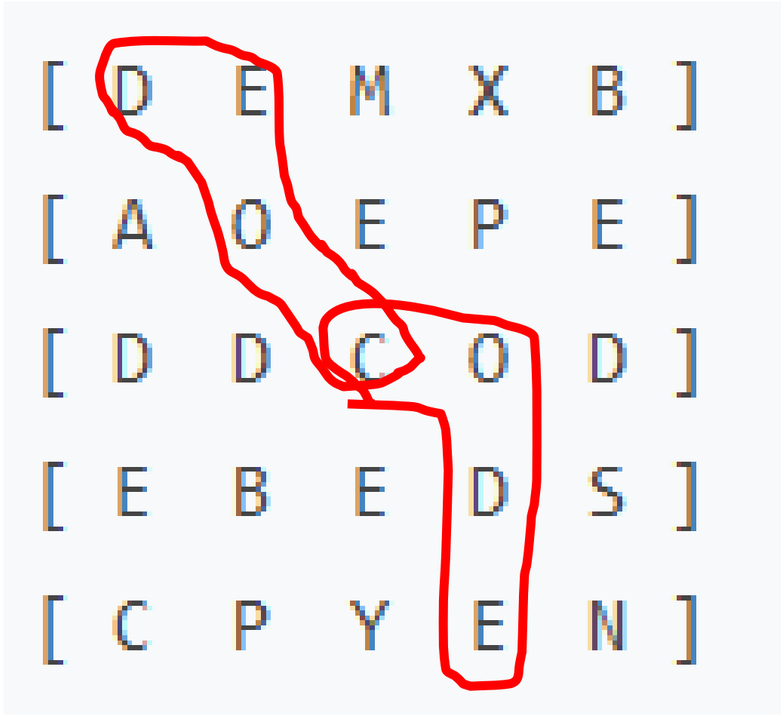
Given an M × N matrix of characters, find all occurrences of a given string in the matrix. We are allowed to search the string in all eight possible directions, i.e., North, West, South, East, North-East, North-West, South-East, South-West. Note that there should not be any cycles in the output path.

```
[ D  E  M  X  B ]
[ A  O  E  P  E ]
[ D  D  C  O  D ]
[ E  B  E  D  S ]
[ C  P  Y  E  N ]
```

CODE

Example problems for BFS & DFS

Given an M × N matrix of characters, find all occurrences of a given string in the matrix. We are allowed to search the string in all eight possible directions, i.e., North, West, South, East, North-East, North-West, South-East, South-West. Note that there should not be any cycles in the output path.



Example problems for BFS & DFS

Given an M × N matrix of characters, find all occurrences of a given string in the matrix. We are allowed to search the string in all eight possible directions, i.e., North, West, South, East, North-East, North-West, South-East, South-West. Note that there should not be any cycles in the output path.

```
[ D  E  M  X  B ]
[ A  O  E  P  E ]
[ D  D  C  O  D ]
[ E  B  E  D  S ]
[ C  P  Y  E  N ]
```

➡ CODE ➡

```
C(2, 2)   O(1, 1)   D(0, 0)   E(0, 1)
C(2, 2)   O(1, 1)   D(2, 0)   E(3, 0)
C(2, 2)   O(1, 1)   D(2, 1)   E(1, 2)
C(2, 2)   O(1, 1)   D(2, 1)   E(3, 0)
C(2, 2)   O(1, 1)   D(2, 1)   E(3, 2)
C(2, 2)   O(2, 3)   D(2, 4)   E(1, 4)
C(2, 2)   O(2, 3)   D(3, 3)   E(3, 2)
C(2, 2)   O(2, 3)   D(3, 3)   E(4, 3)
```

Example problems for BFS & DFS

Given an M × N matrix of characters, find all occurrences of a given string in the matrix. We are allowed to search the string in all eight possible directions, i.e., North, West, South, East, North-East, North-West, South-East, South-West. Note that there should not be any cycles in the output path.

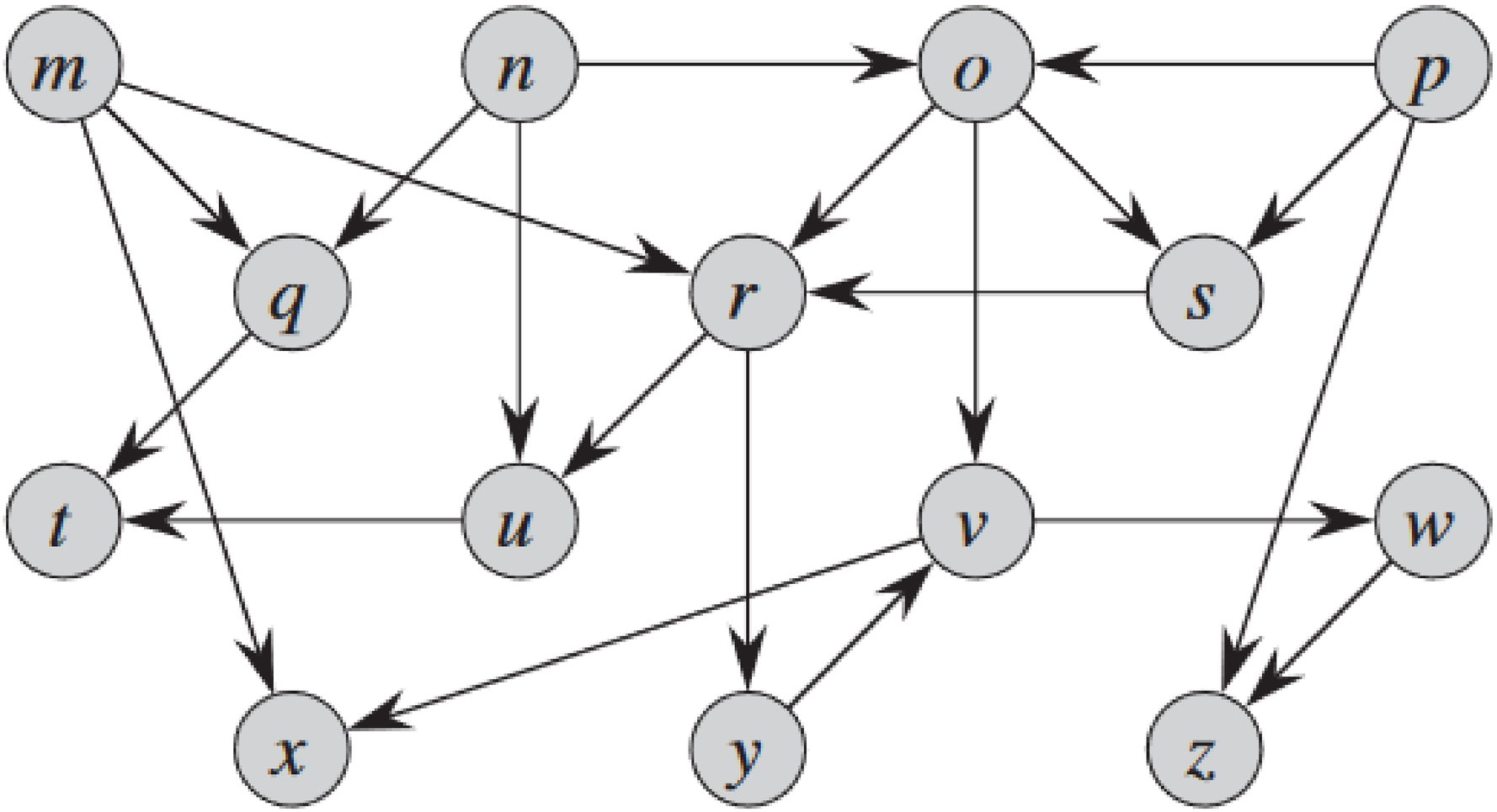
1. Use DFS

2. Start from each cell in the matrix and explore all eight paths possible

3. Recursively check if they will lead to the solution or not.

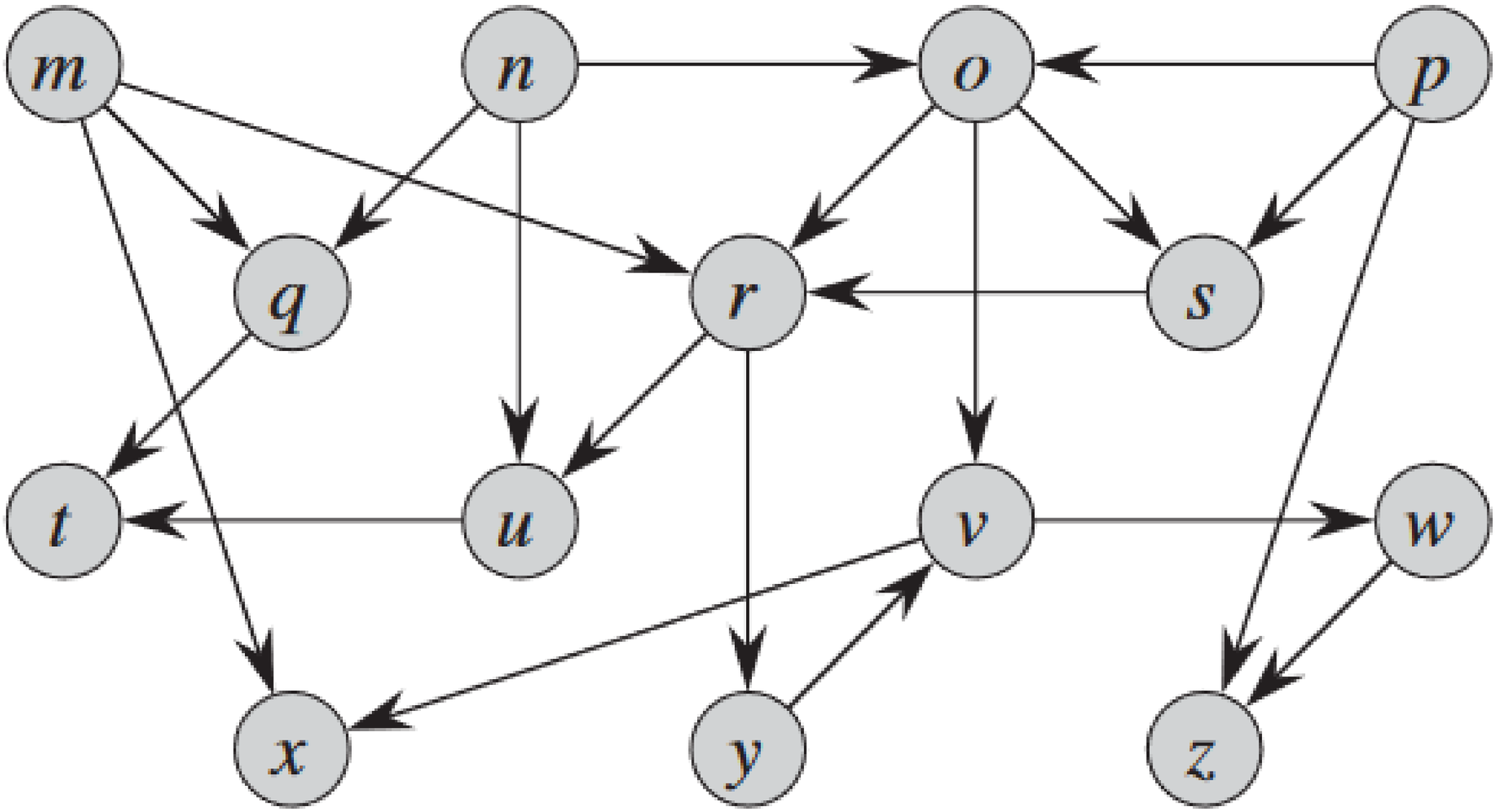
4. To make sure that the path is simple and doesn't contain any cycles, keep track of cells involved in the current path in a matrix, and before exploring any cell, ignore the cell if it is already covered in the current path. (Backtracking)

Topological Sorting



Consider the graph above and determine whether it is a Directed Acyclic Graph. If it is, run a topological sorting to get topological order.

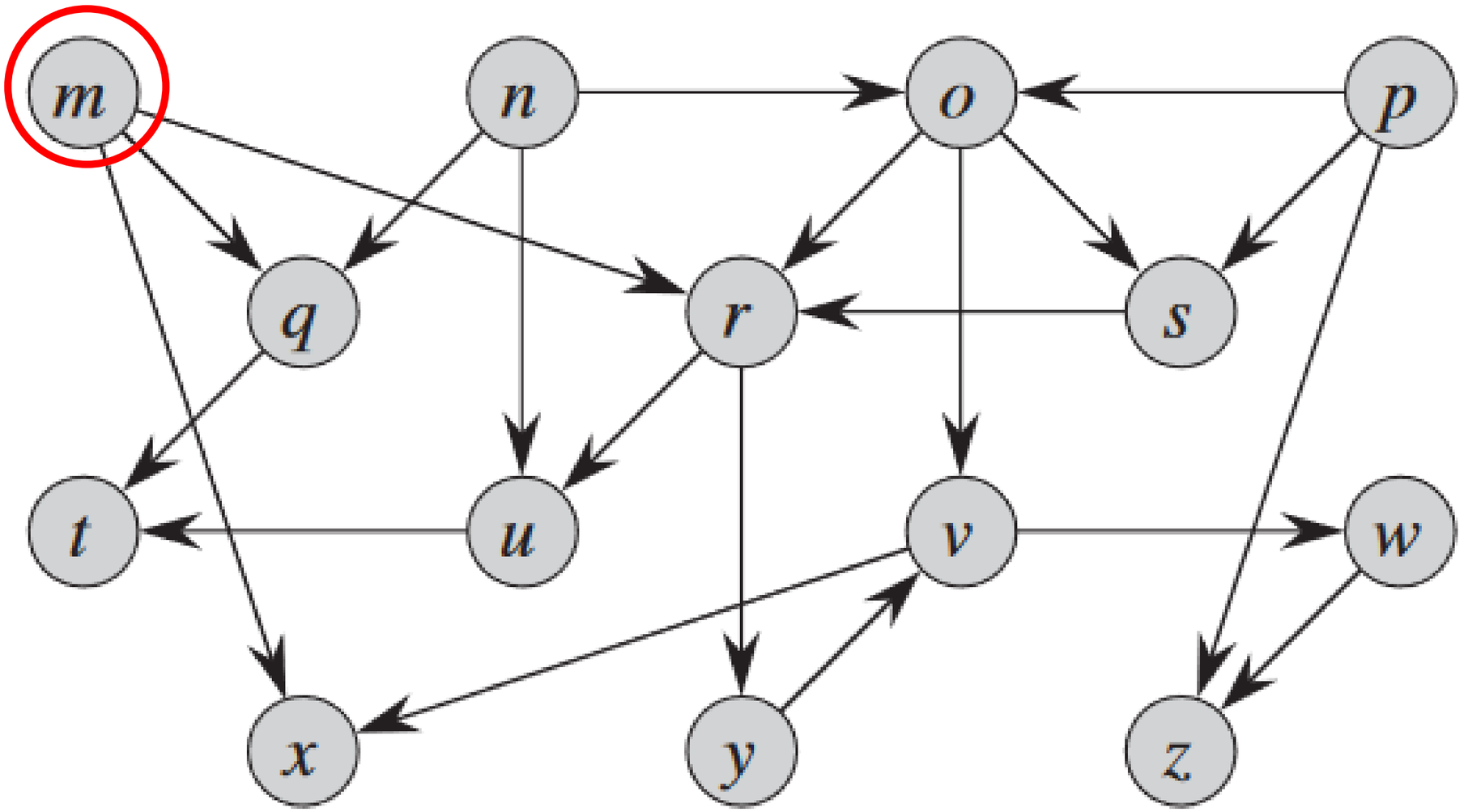
Topological Sorting



Apply DFS to find Topological order

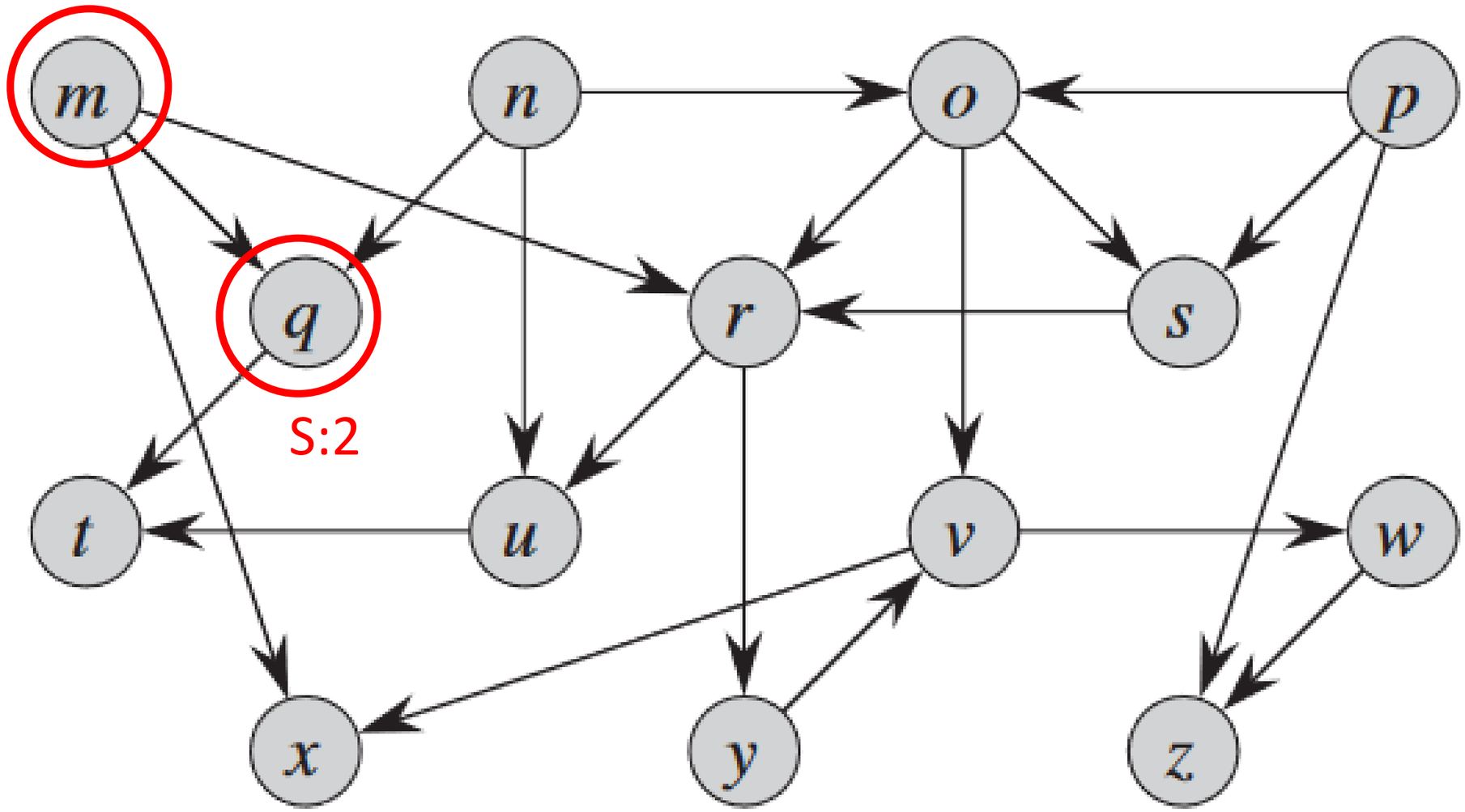
Topological Sorting

S:1



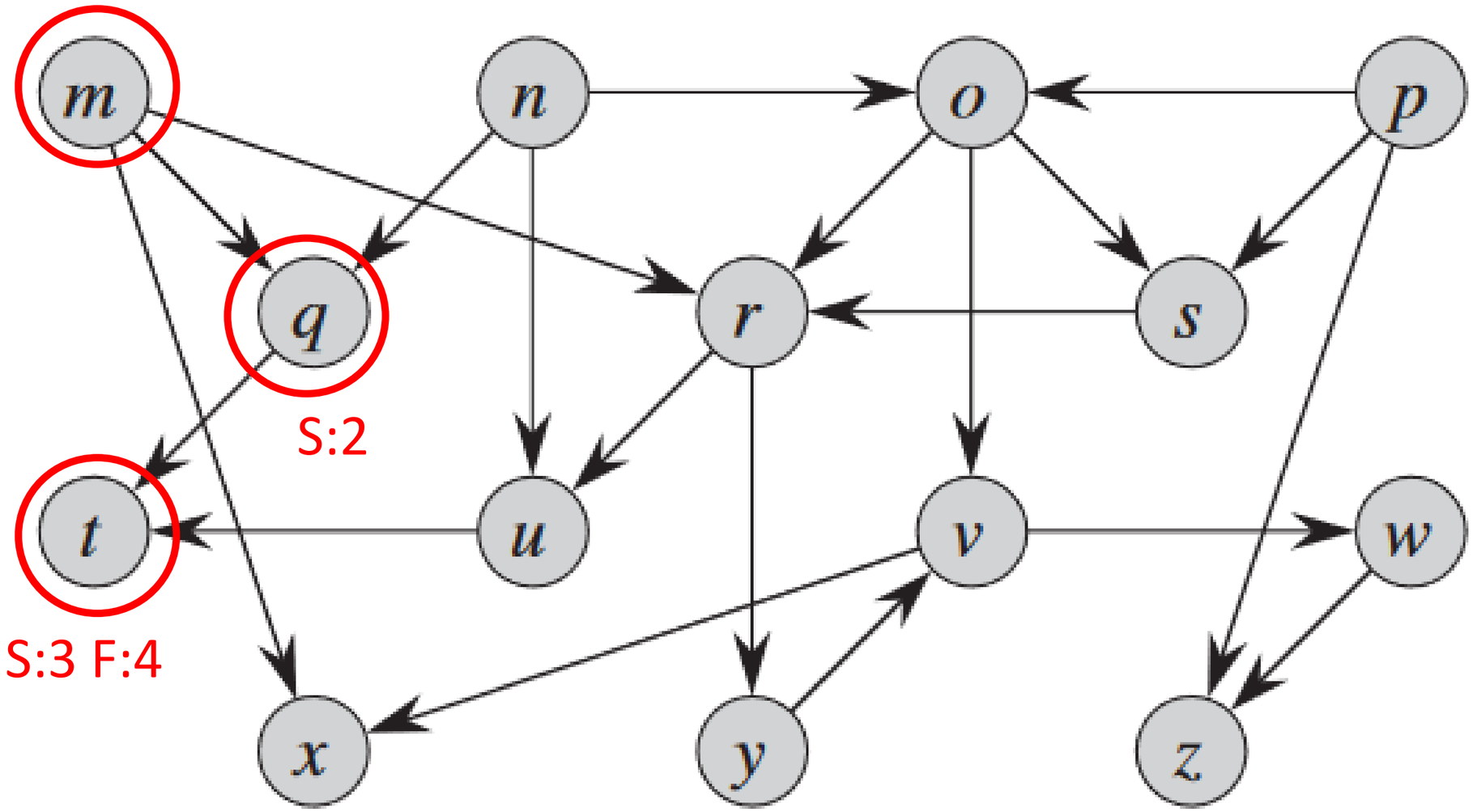
Topological Sorting

S:1

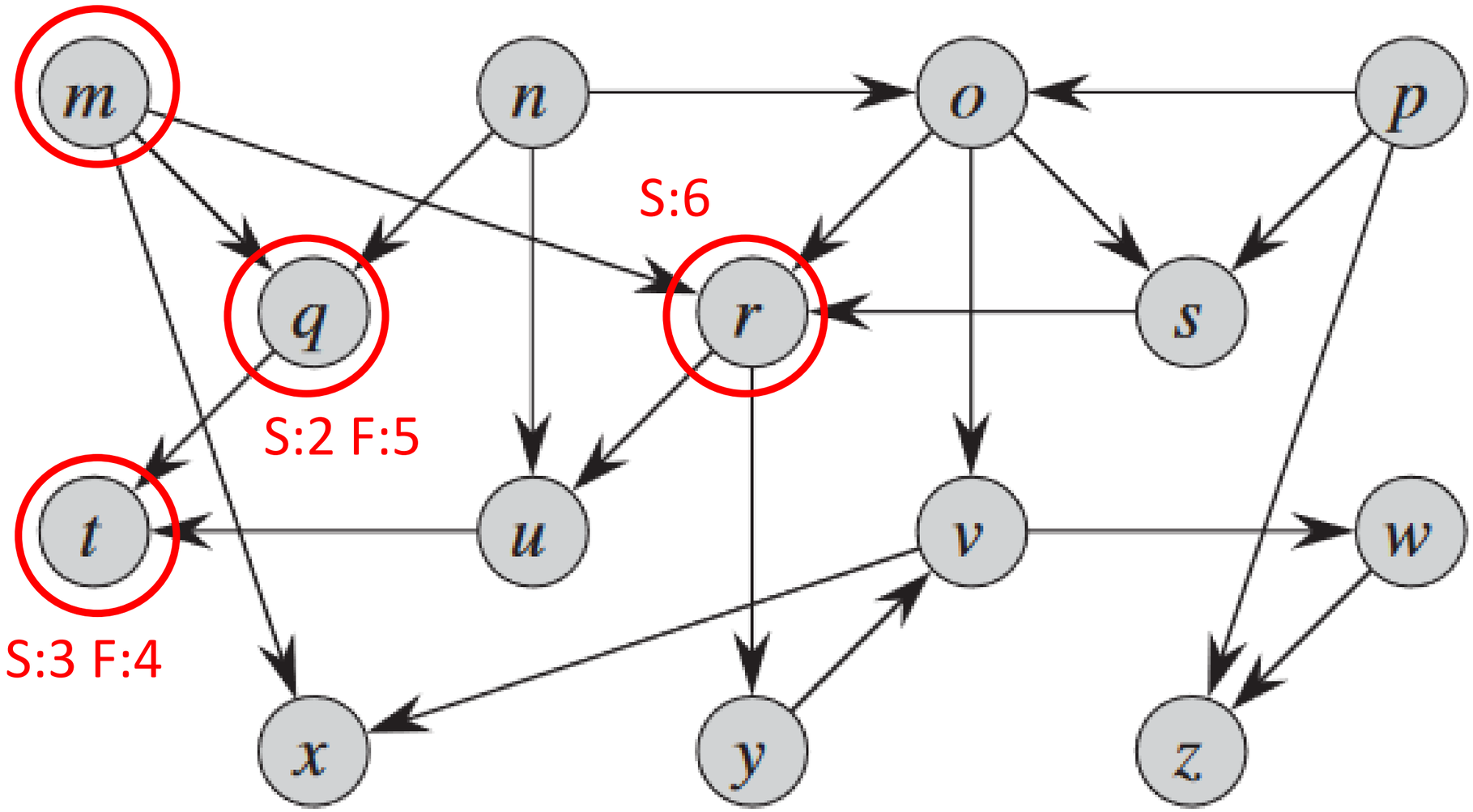


S:2

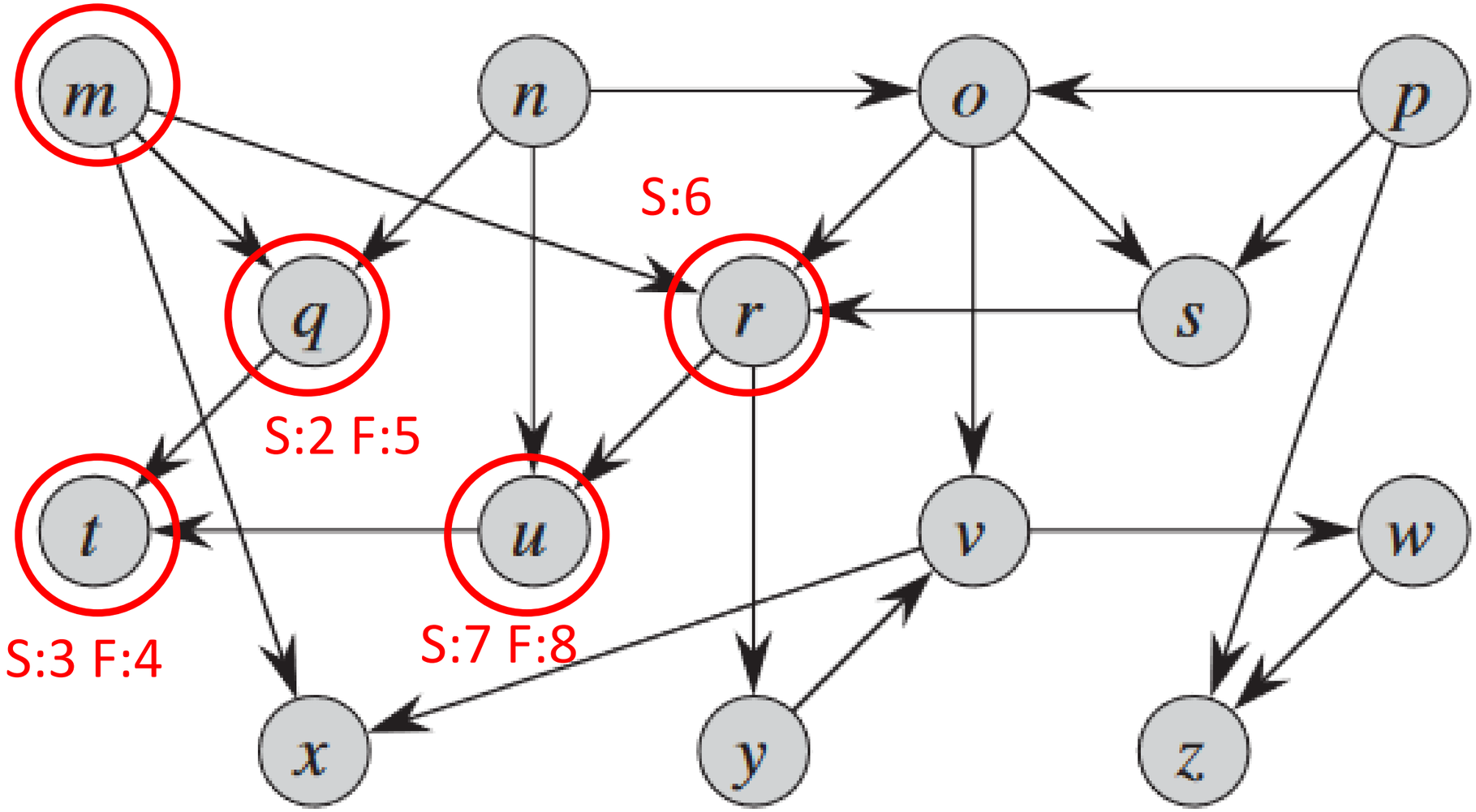
Topological Sorting



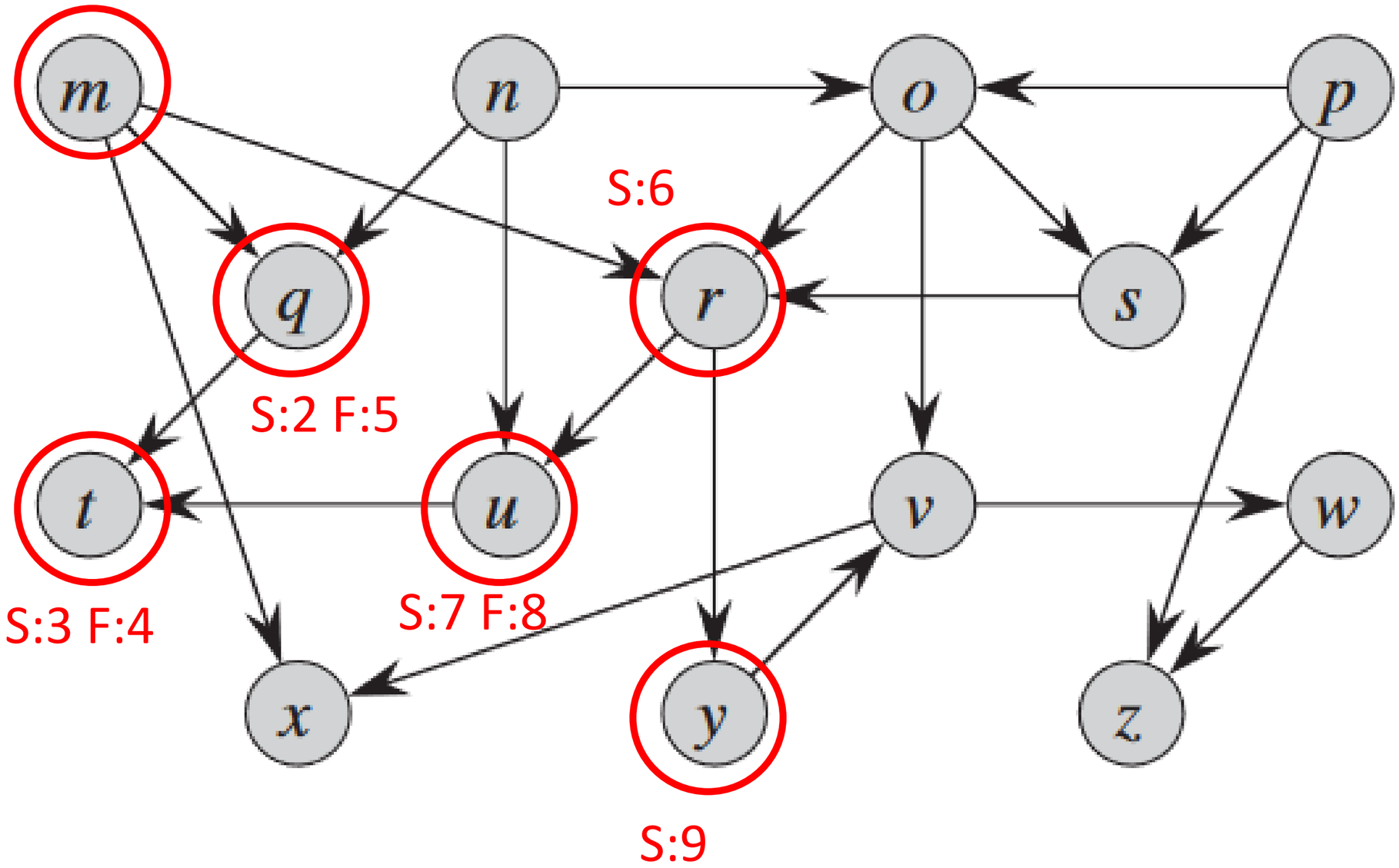
Topological Sorting



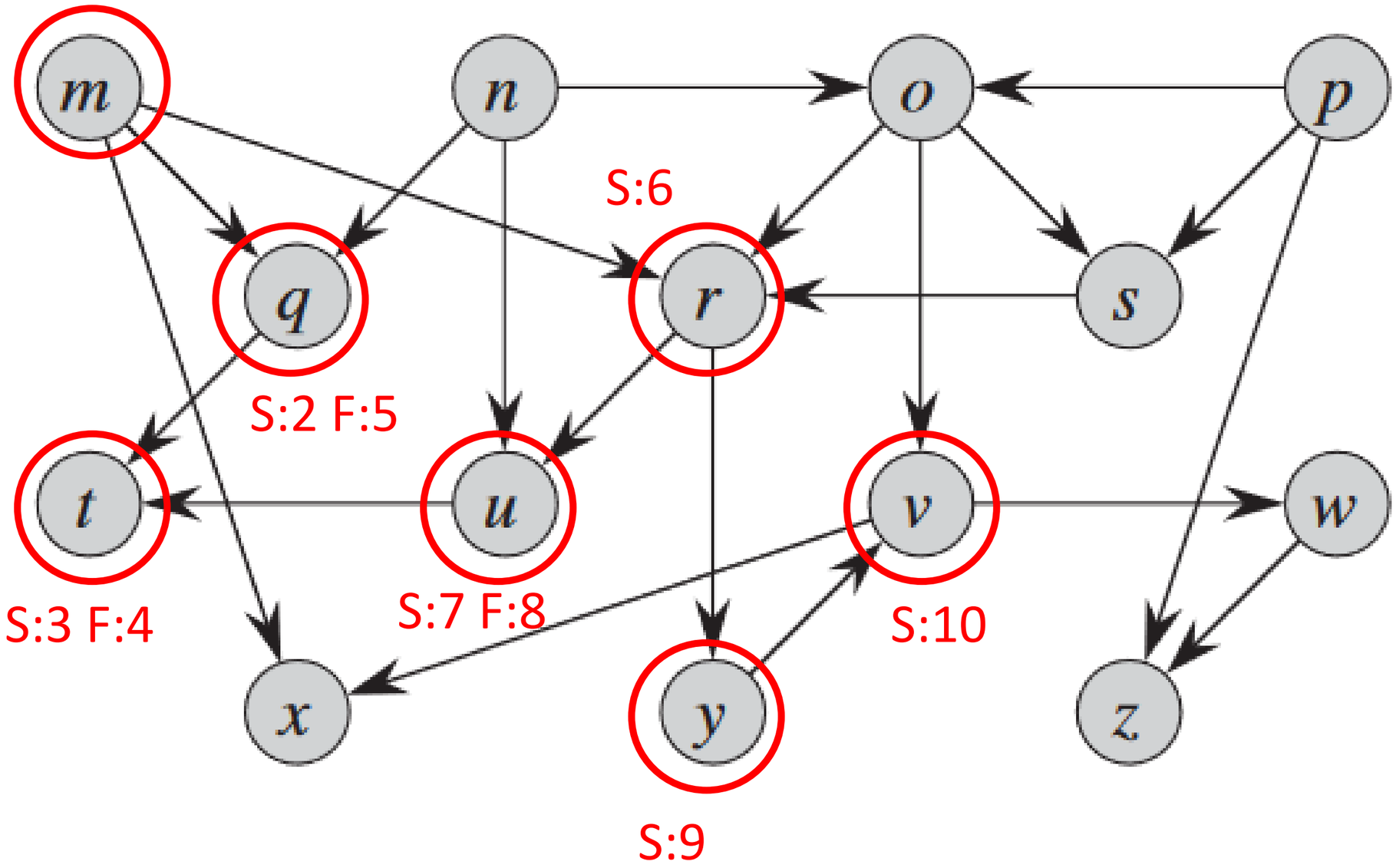
Topological Sorting



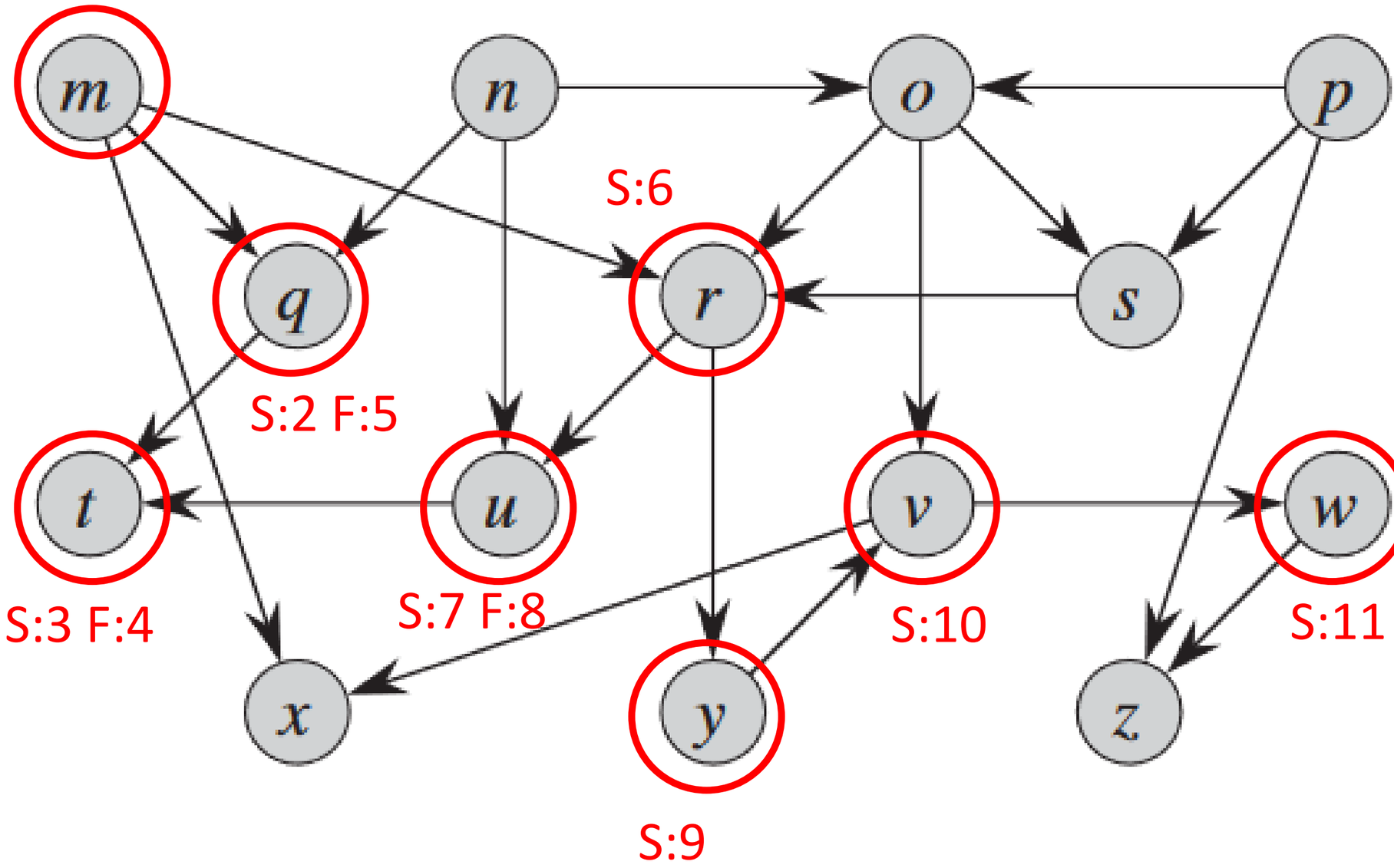
Topological Sorting



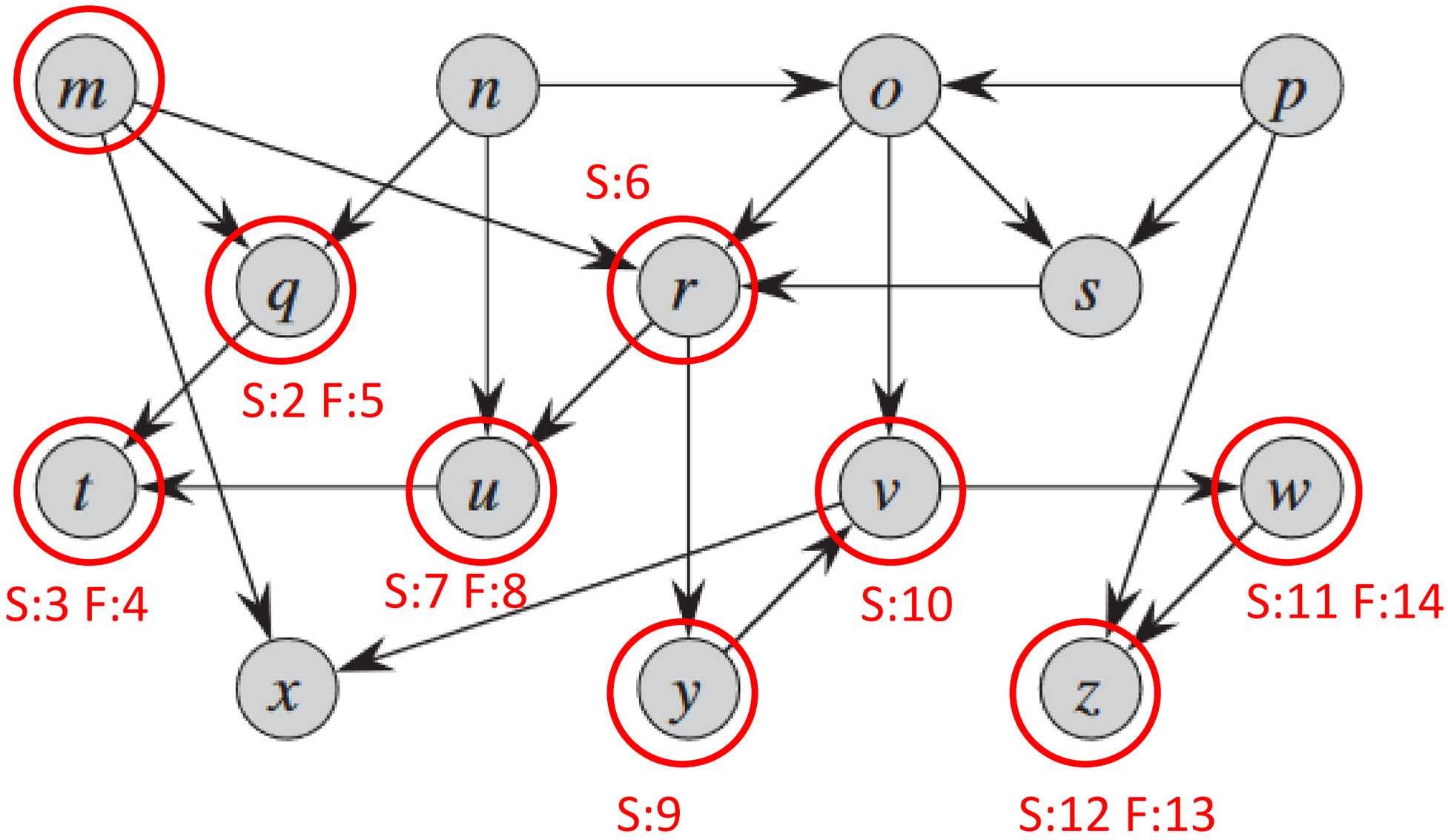
Topological Sorting



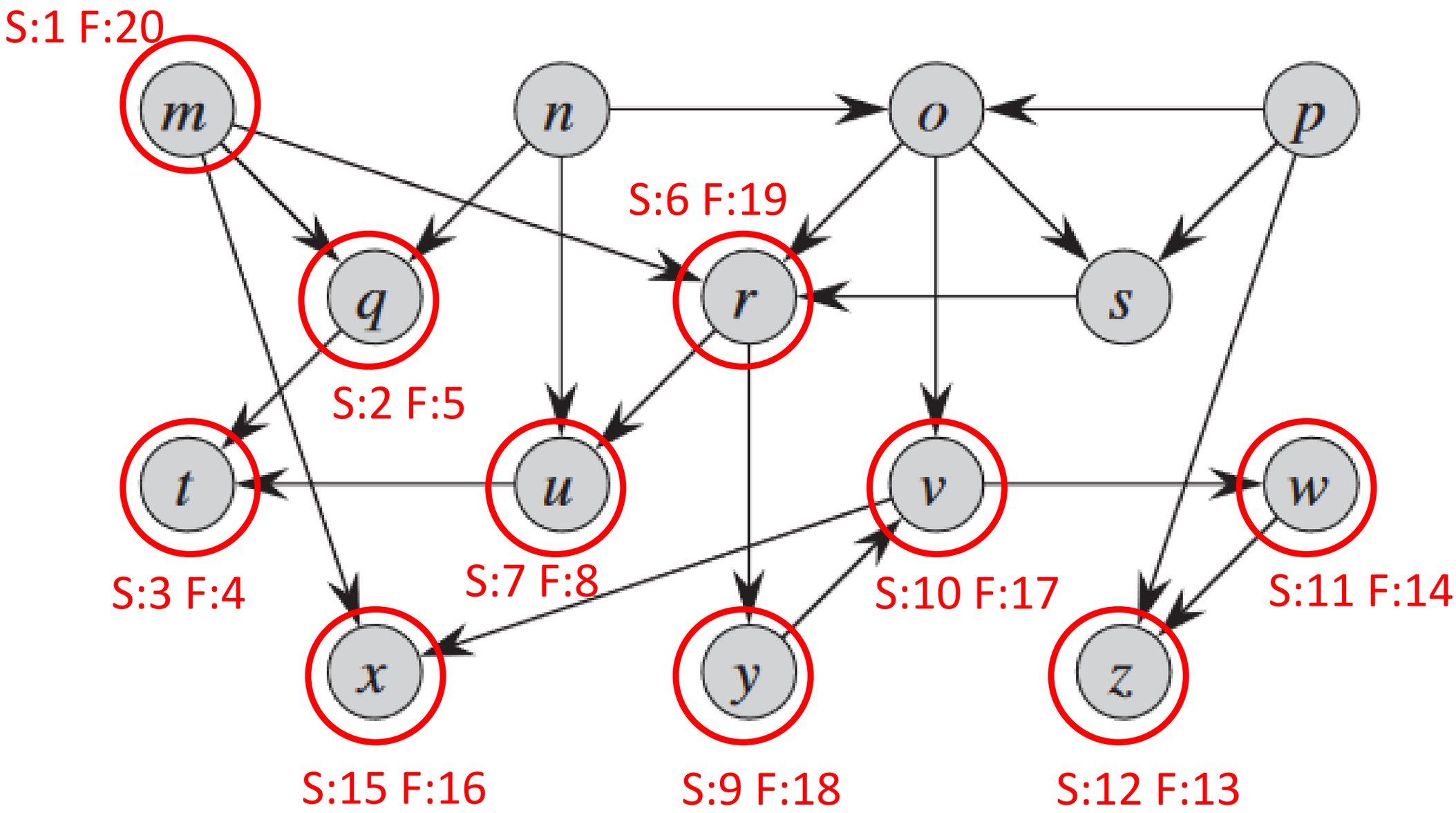
Topological Sorting



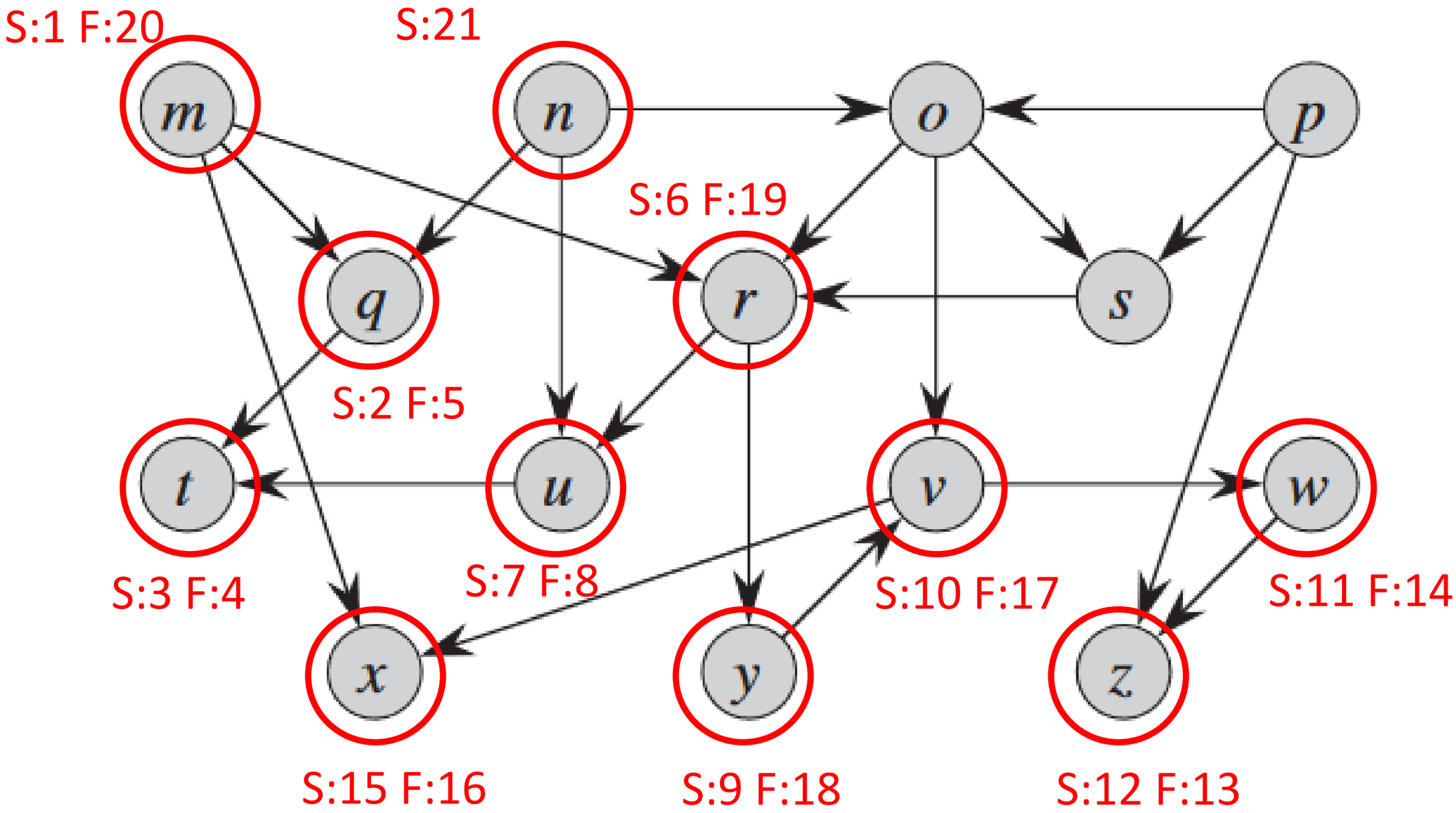
Topological Sorting



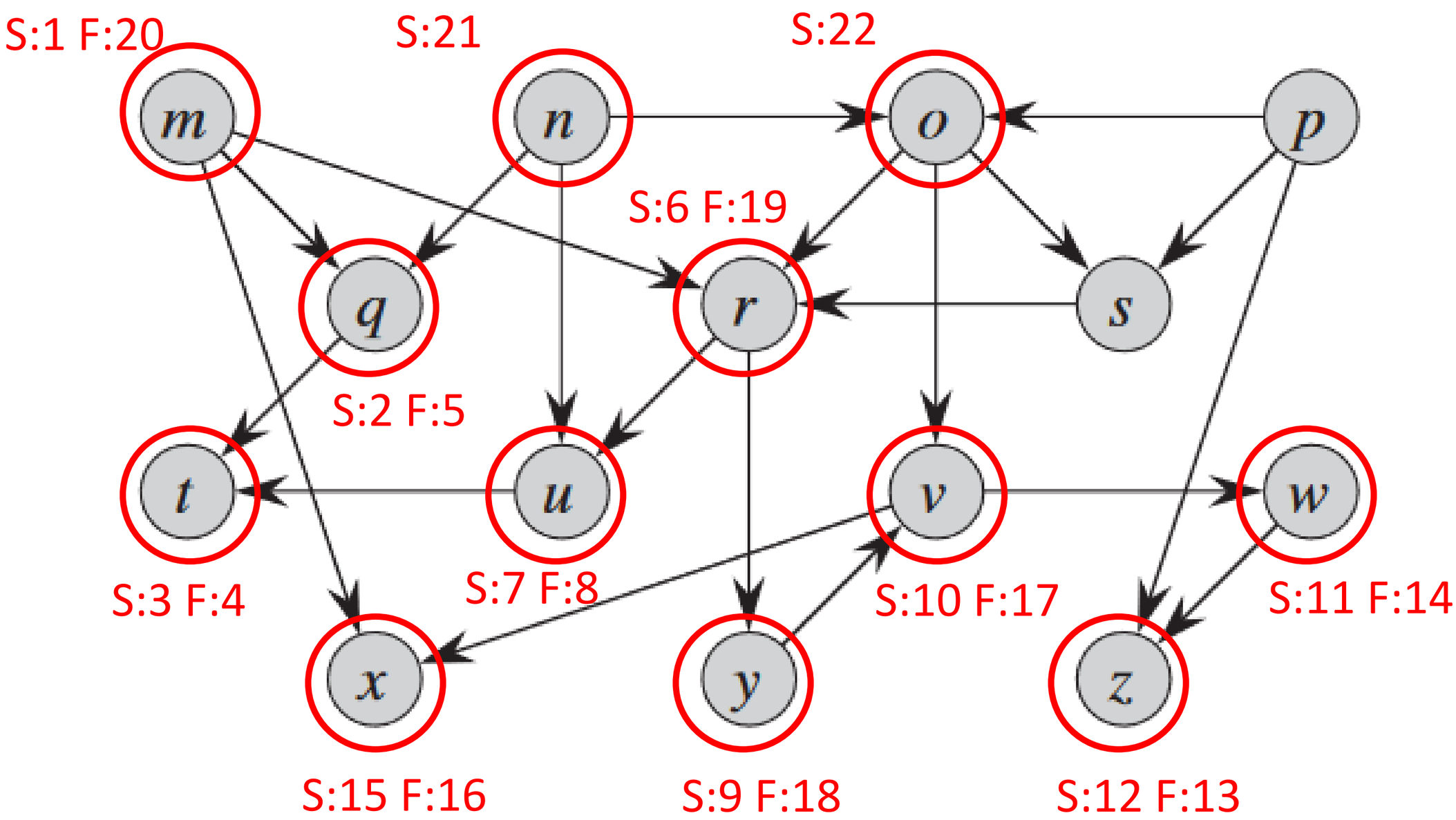
Topological Sorting



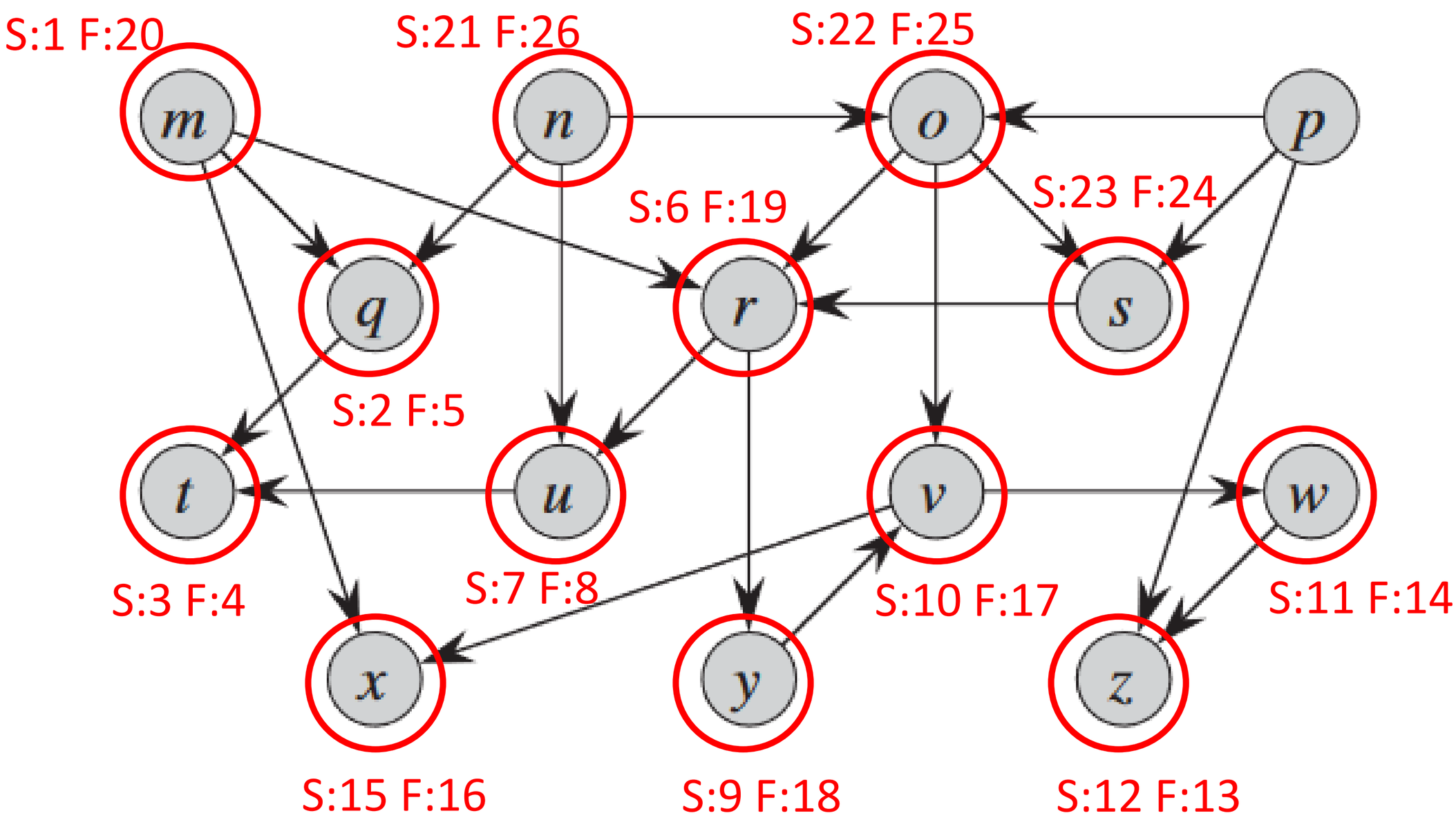
Topological Sorting



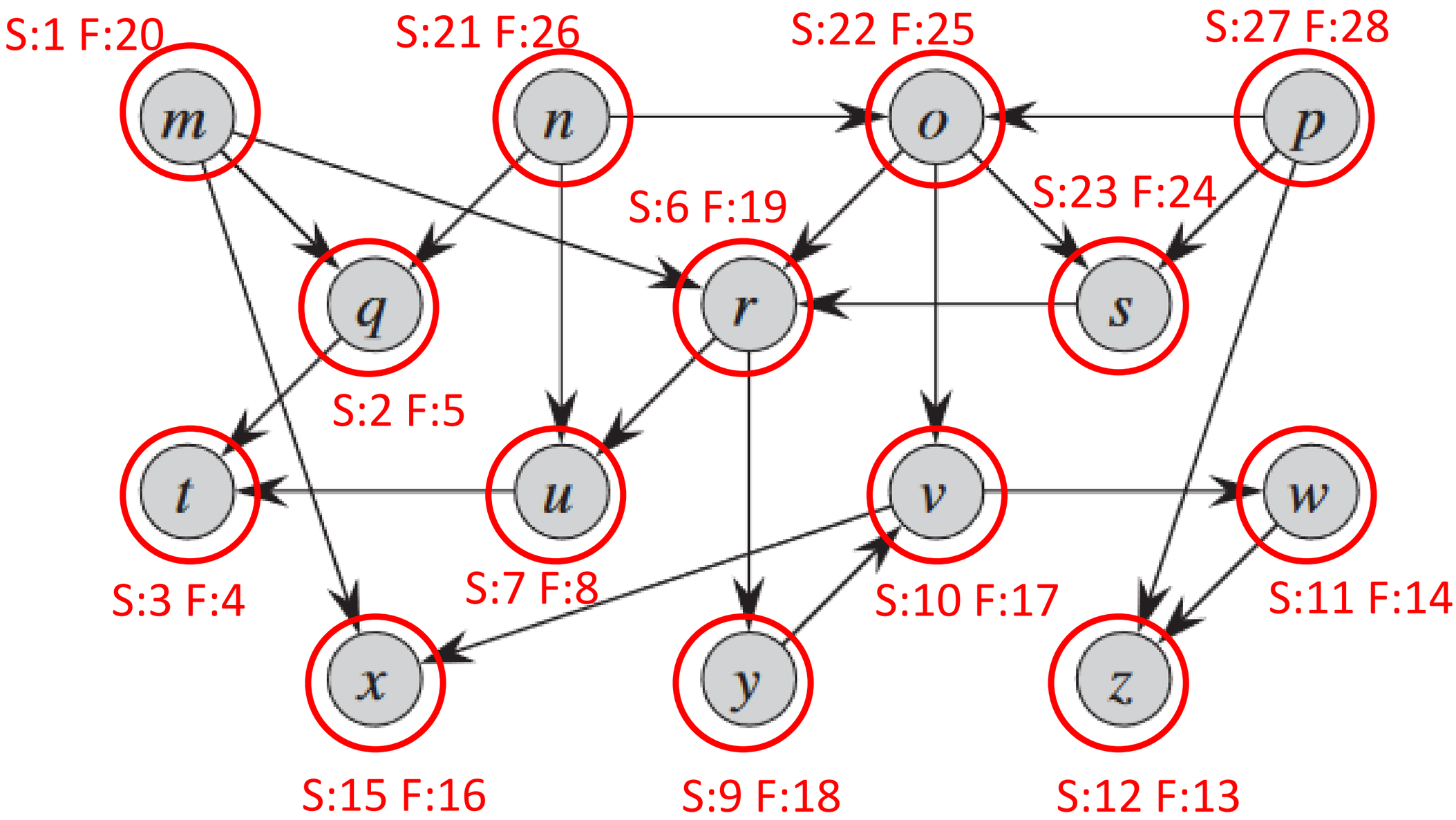
Topological Sorting



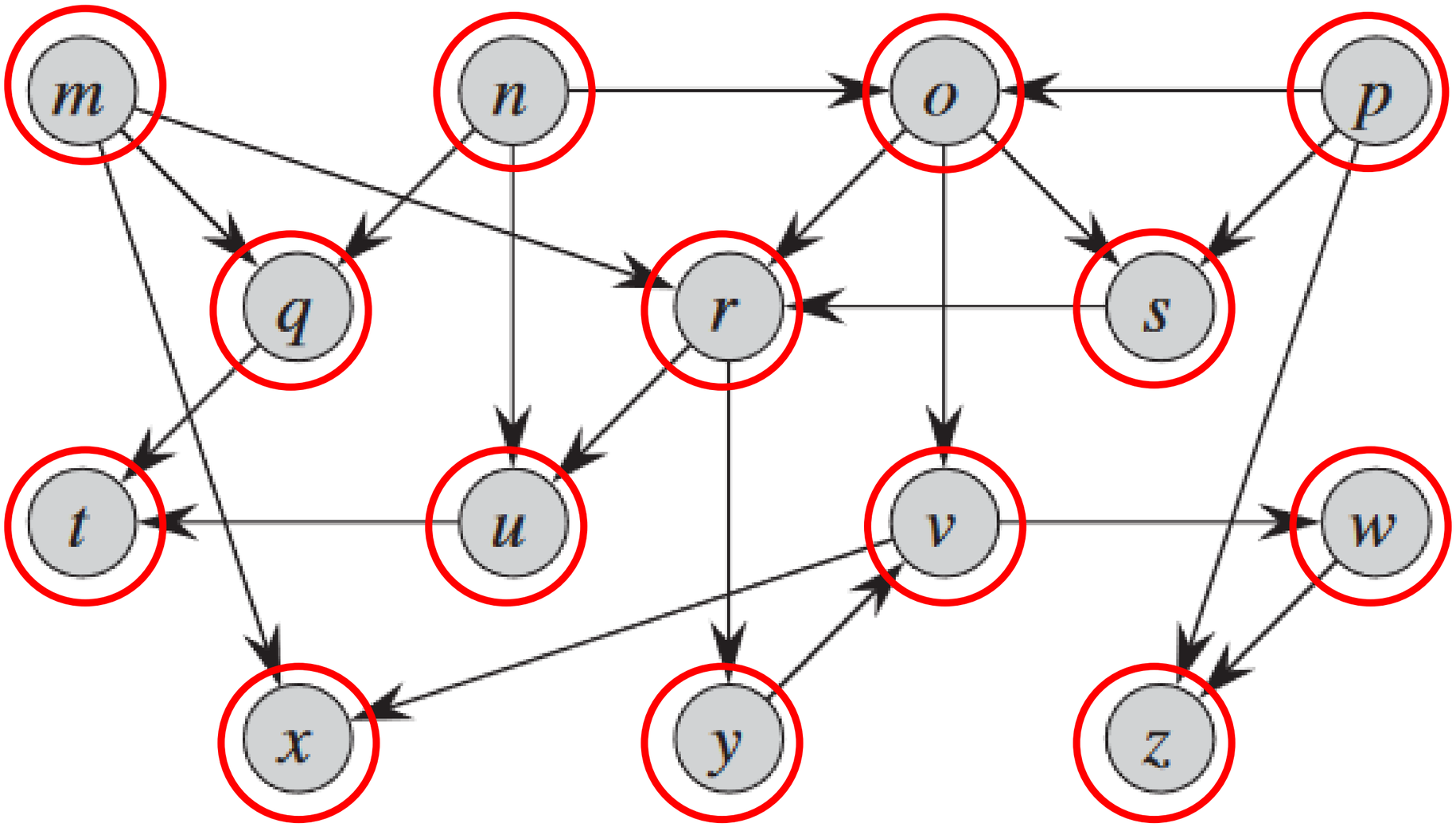
Topological Sorting



Topological Sorting



Topological Sorting



label	d	f
m	1	20
q	2	5
t	3	4
r	6	19
u	7	8
y	9	18
v	10	17
w	11	14
z	12	13
x	15	16
n	21	26
o	22	25
s	23	24
p	27	28

Read off the entries in decreasing order of finish time

So p, n, o, s, m, r, y, v, x, w, z, u, q, t

Topological Sorting