Lecture#9

Data Structures

Dr. Abu Nowshed Chy

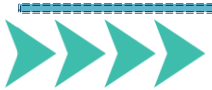
Department of Computer Science and Engineering

University of Chittagong

February 23, 2025

Faculty Profile

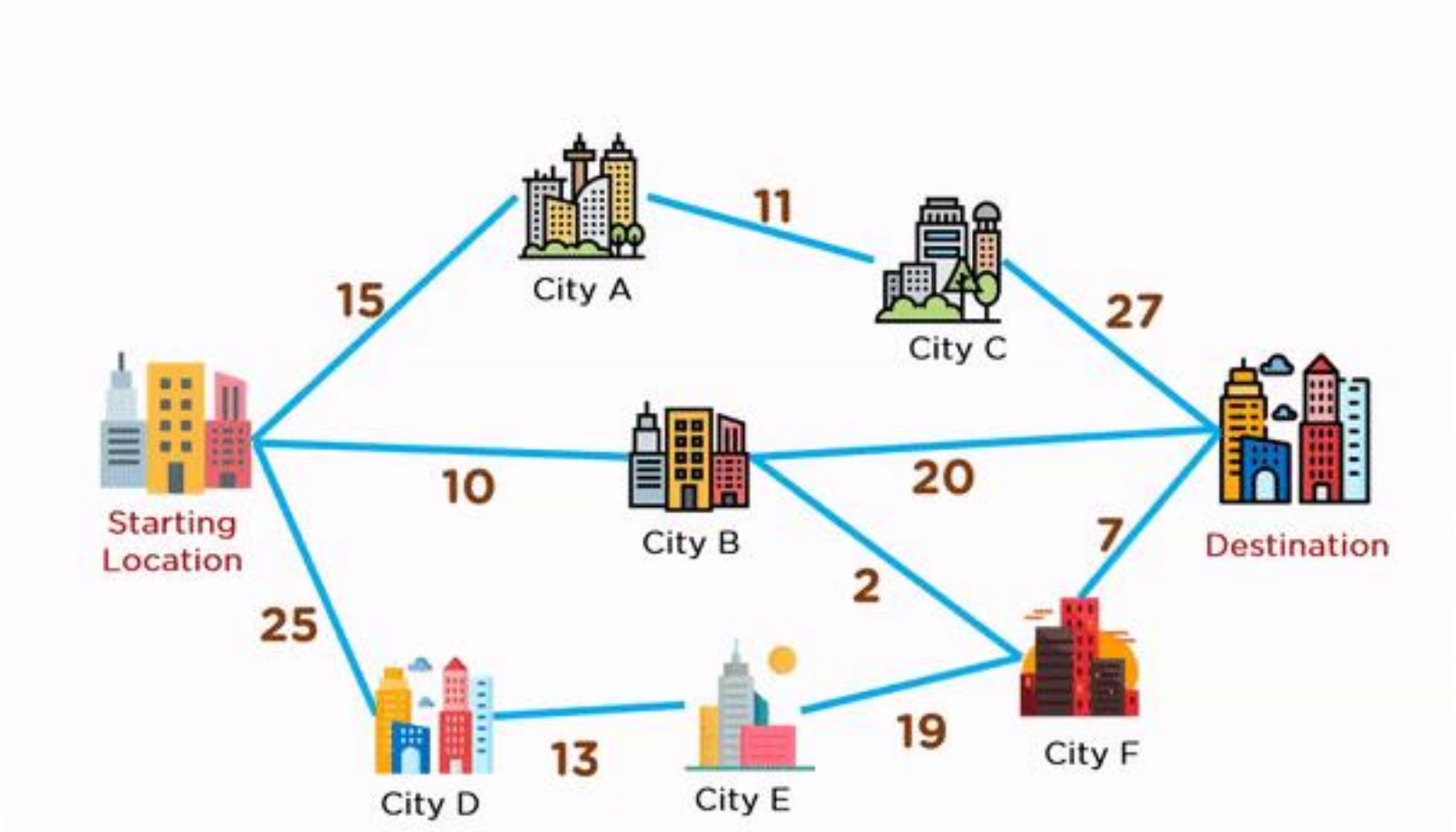
Graphs



Graphs Basics

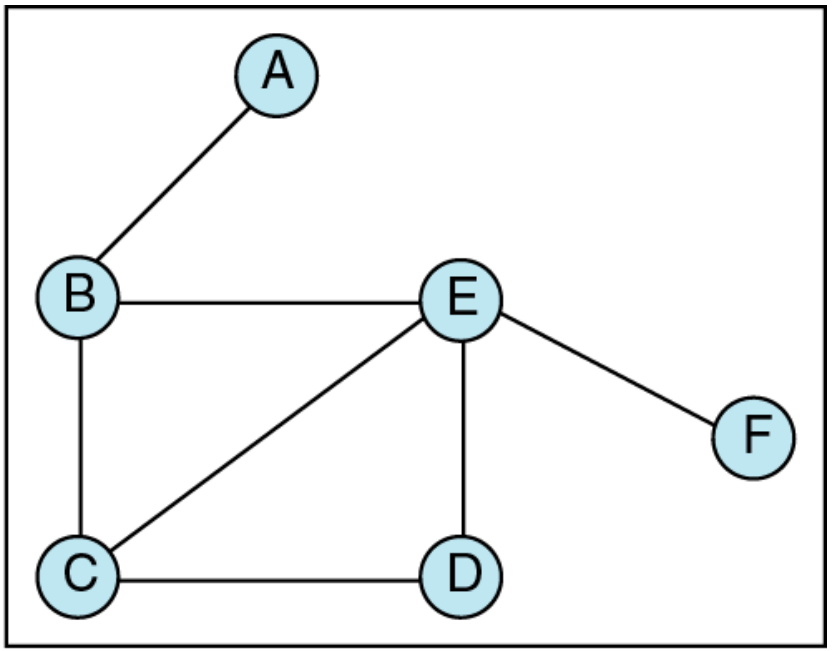
- A ***graph*** is a collection of nodes, called ***vertices***, and a collection of line segments, called ***lines, edges,*** or ***arcs***, connecting pairs of vertices.
- In other words, a graph consists of 2 sets:
  - A set of lines
  - A set of vertices

Graphs Basics



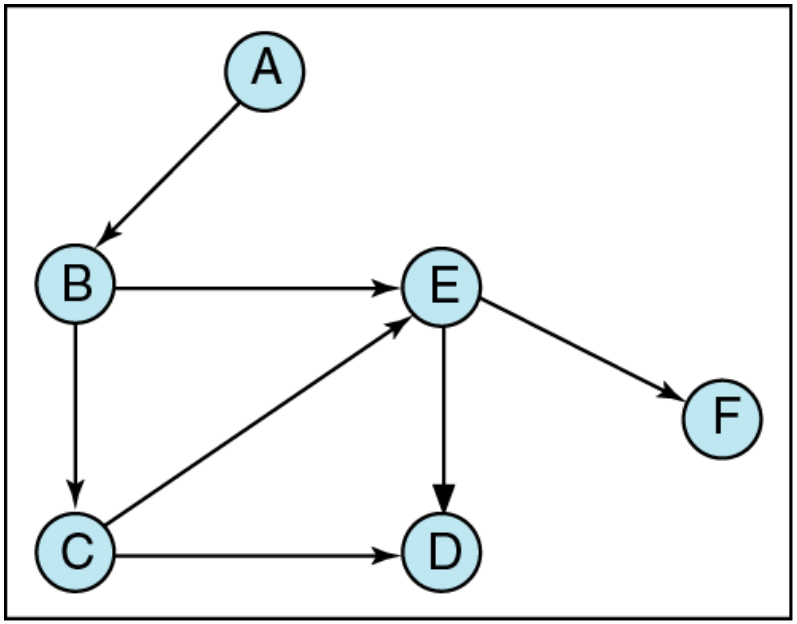
Graphs Basics

- Graphs may be either directed or undirected.
- An ***undirected graph*** is a graph in which there is no direction associated with the lines.
- In an undirected graph, the flow between two vertices can go in either direction.



Graphs Basics

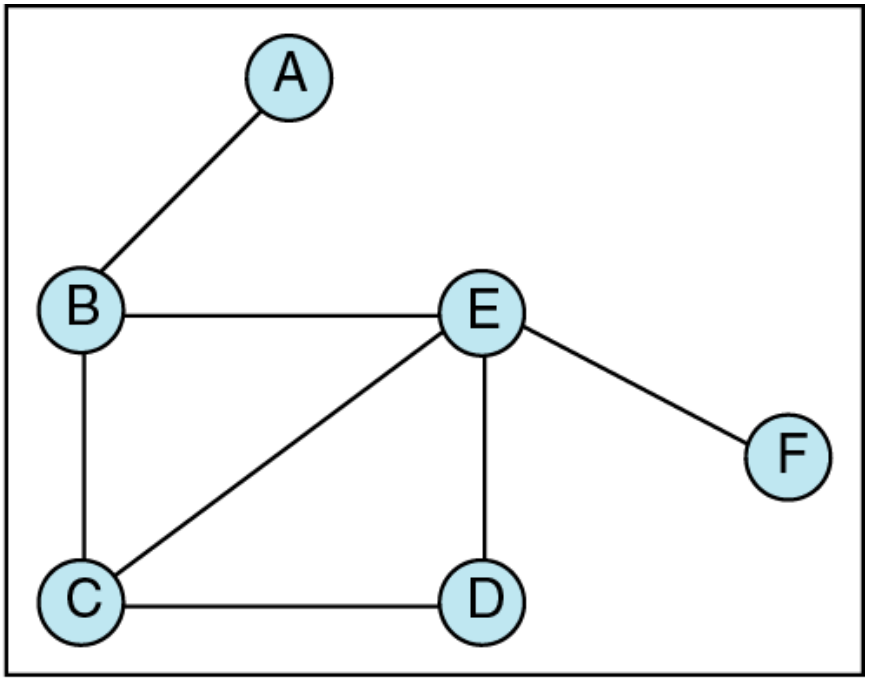
- A ***directed graph*** or ***digraph*** is a graph in which each line has a direction associated with it.
- In a directed graph, the flow along an edge between 2 vertices can only follow the indicated direction.



Graphs Basics

- 2 vertices in a graph are said to be ***adjacent*** (or neighbors) if an edge directly connects them.

A and B are adjacent
D and F are not adjacent

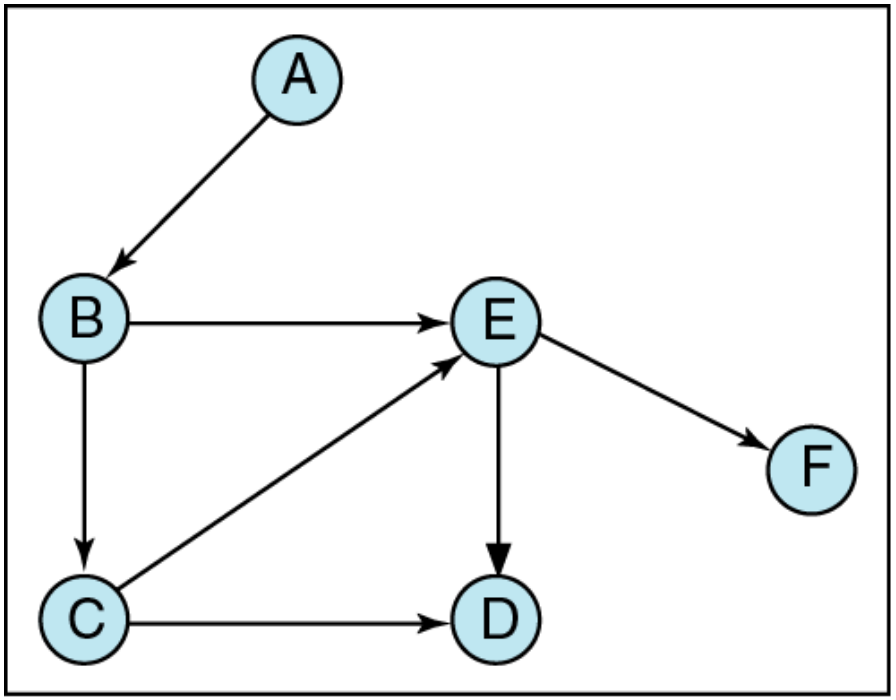


Graphs Basics

- A ***path*** is a sequence of vertices in which each vertex is adjacent to the next one.

A,B,C,E is a path
A,B,E,F is a path

A,B,E,C is *NOT* a path because, although E is adjacent to C, C is not adjacent to E (note direction).



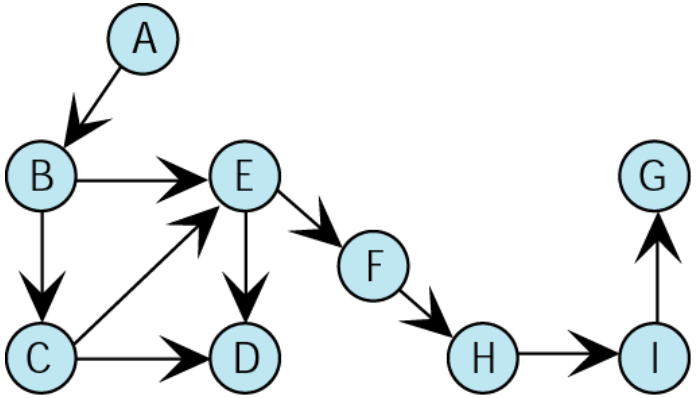
Graphs Basics

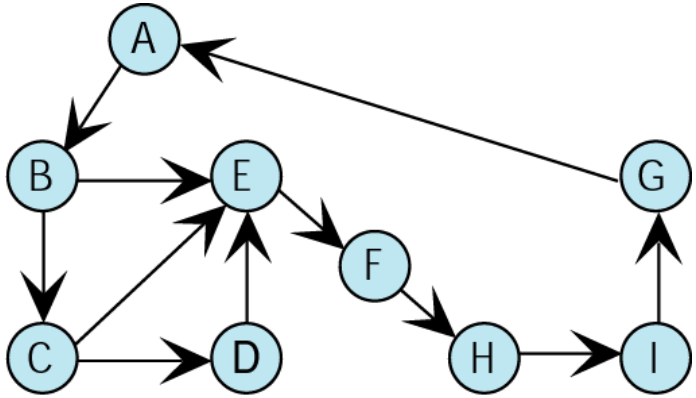
- Note that both directed and undirected graphs have paths.
- In an undirected graph, you may travel in either direction.
- In a directed graph, you may only travel along the given direction of the arcs.

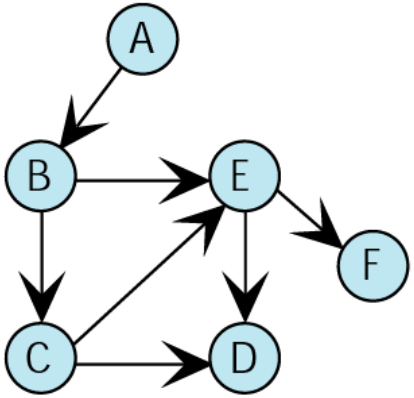
Graphs Basics

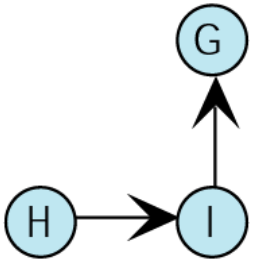
- Two vertices are said to be ***connected*** if there is a path between them.
- A graph is said to be connected if, suppressing direction, there is a path from each vertex to every other vertex.
- Furthermore, a directed graph is ***strongly connected*** if there is a path from each vertex to every other vertex.
- A directed graph is ***weakly connected*** if at least 2 vertices are not connected.
- A graph is ***disconnected*** or ***disjoint*** if it is not connected.

Graphs Basics





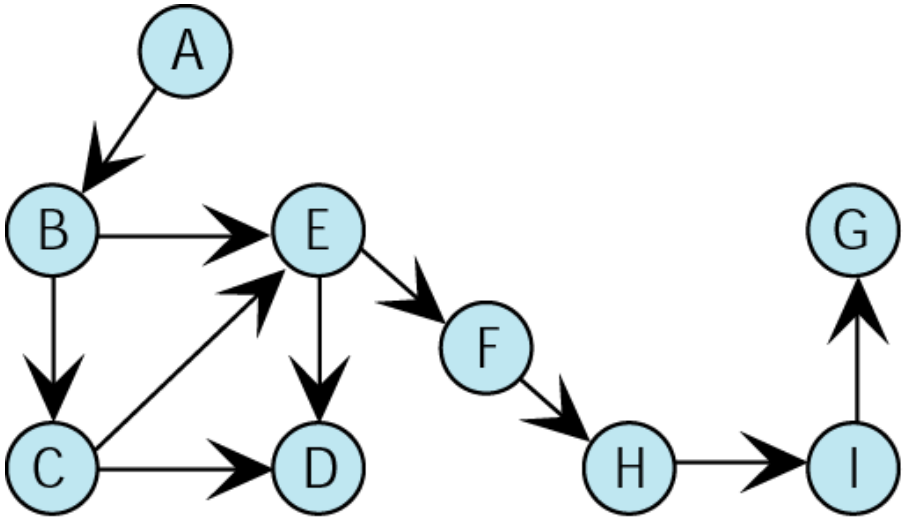




Graphs Basics

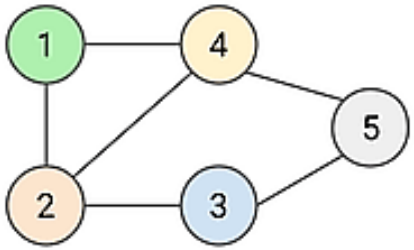
- The ***degree*** of a vertex is the number of edges incident to it.
- The ***outdegree*** of a vertex in a digraph is the number of arcs leaving the vertex.
- The ***indegree*** of a vertex in a digraph is the number of arcs entering the vertex.

Graphs Basics

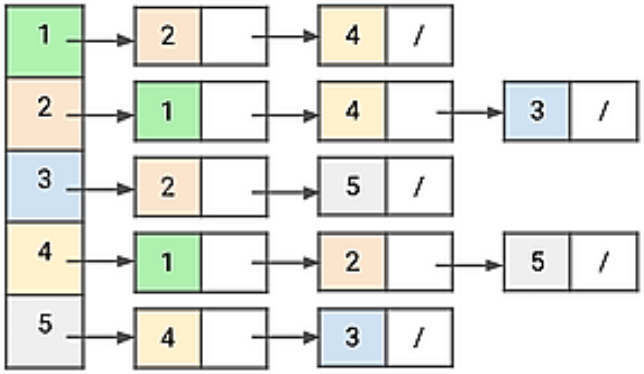


- The degree of vertex B is … 3
- The outdegree of vertex B is … 2
- The indegree of vertex B is … 1

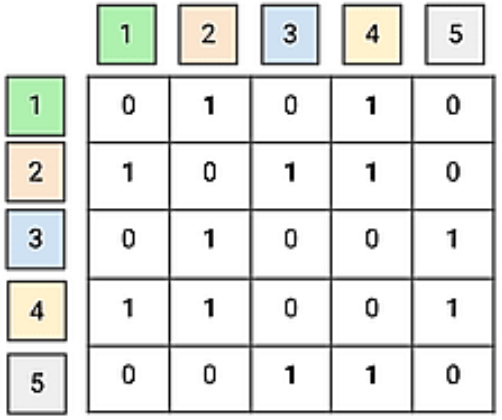
Graph Representation



Undirected Graph

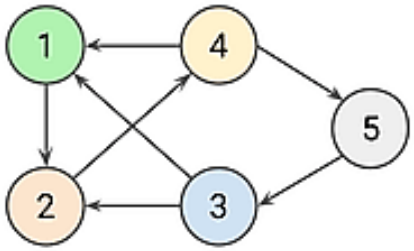


Adjacency List Representation

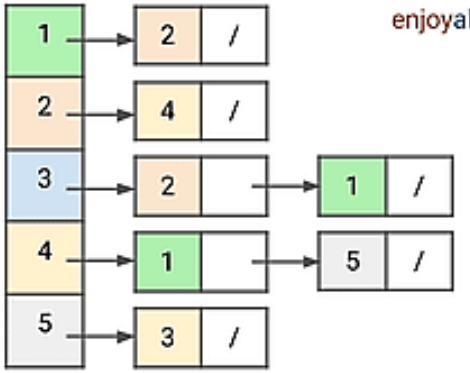


	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	1	0
3	0	1	0	0	1
4	1	1	0	0	1
5	0	0	1	1	0

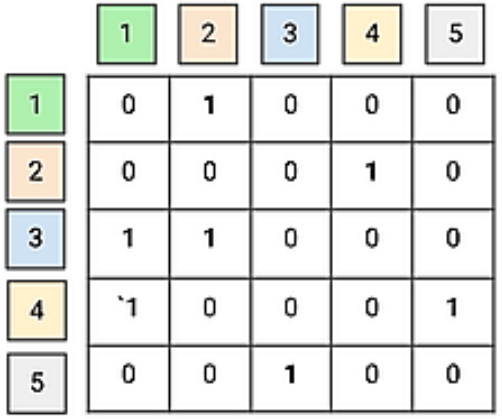
Adjacency Matrix Representation



Directed Graph



enjoyalgorithms.com



	1	2	3	4	5
1	0	1	0	0	0
2	0	0	0	1	0
3	1	1	0	0	0
4	`1	0	0	0	1
5	0	0	1	0	0

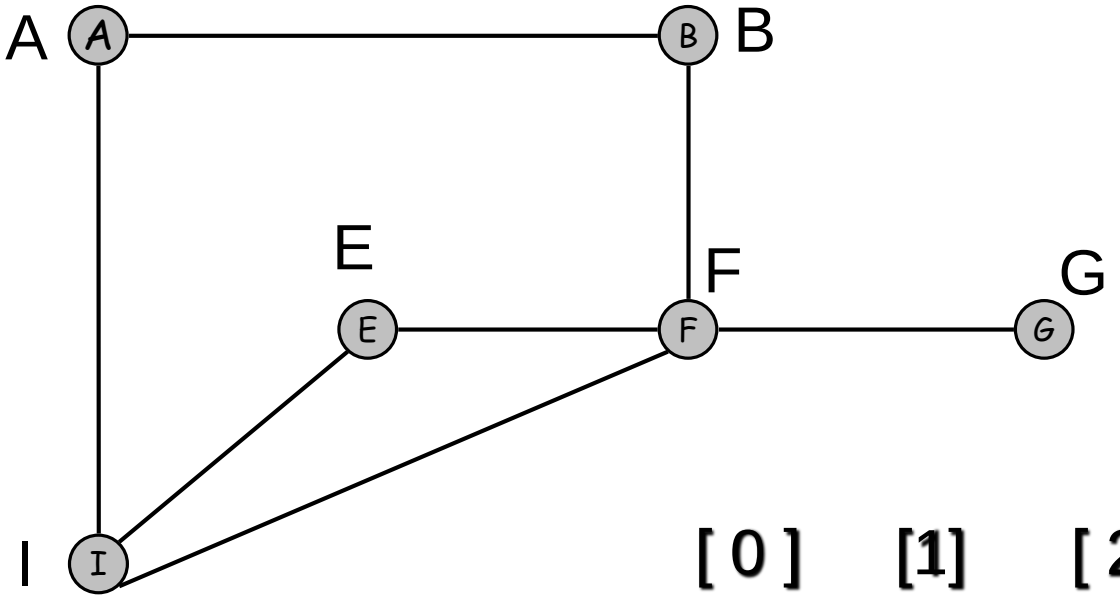
Graph Traversal

- Because a vertex in a graph can have multiple parents, the traversal of a graph must ensure that we process the data in each vertex once and only once.
- The traditional solution to this problem is to include a 'visited' flag at each vertex.
- Before the traversal, we set the visited flag in each vertex to 'off.'
- Then, as we traverse the graph, we set the visited flag to 'on' to indicate that the data in a given vertex has been processed.

Graph Traversal

- There are 2 standard graph traversals:
  - Breadth First Search (Traversal) → QUEUE
  - Depth First Search (Traversal) → STACK
- Both of these methods use the 'visited' flag.

Breadth First Search (BFS) Traversal



front

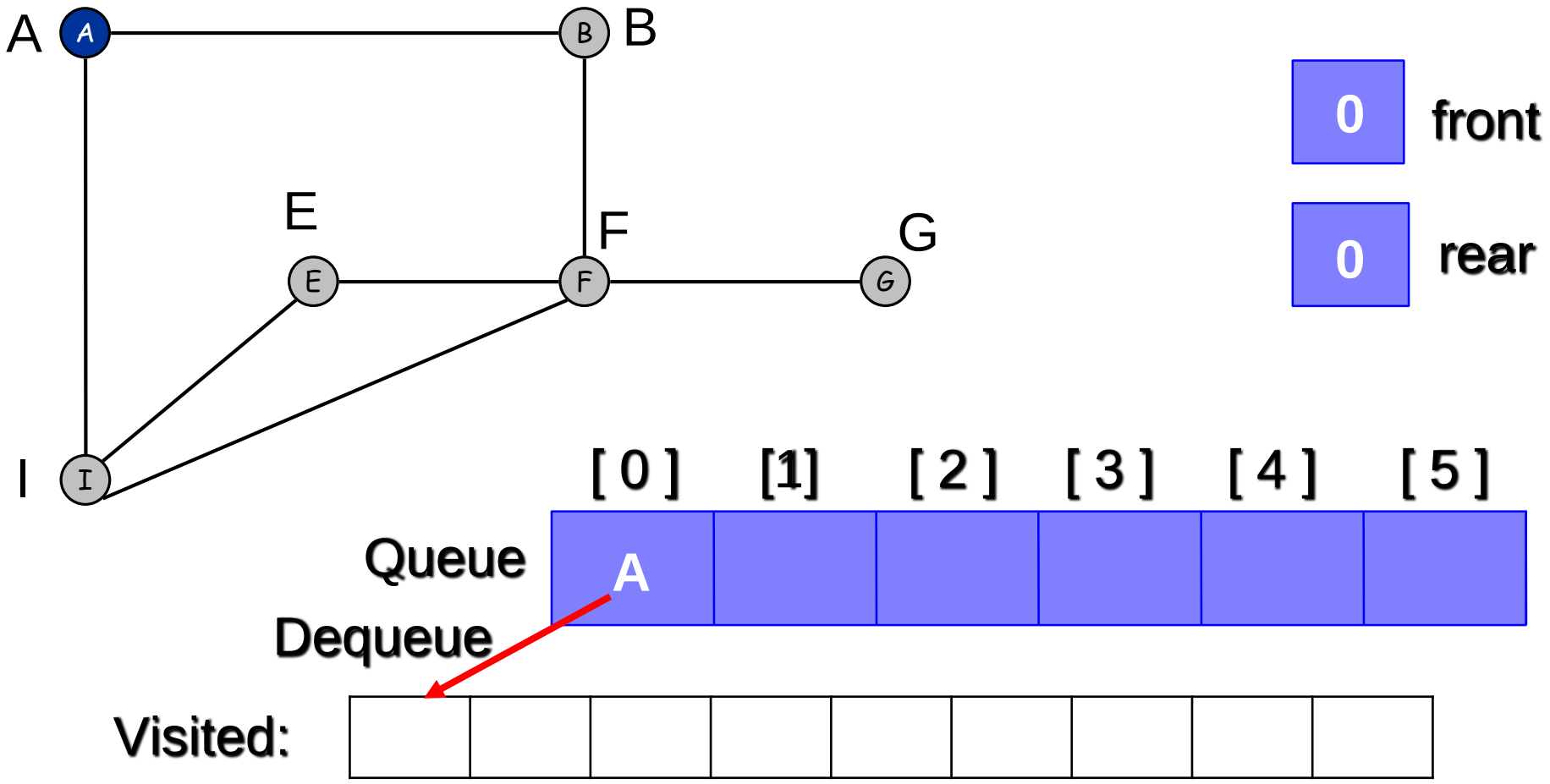
rear

[0] [1] [2] [3] [4] [5]

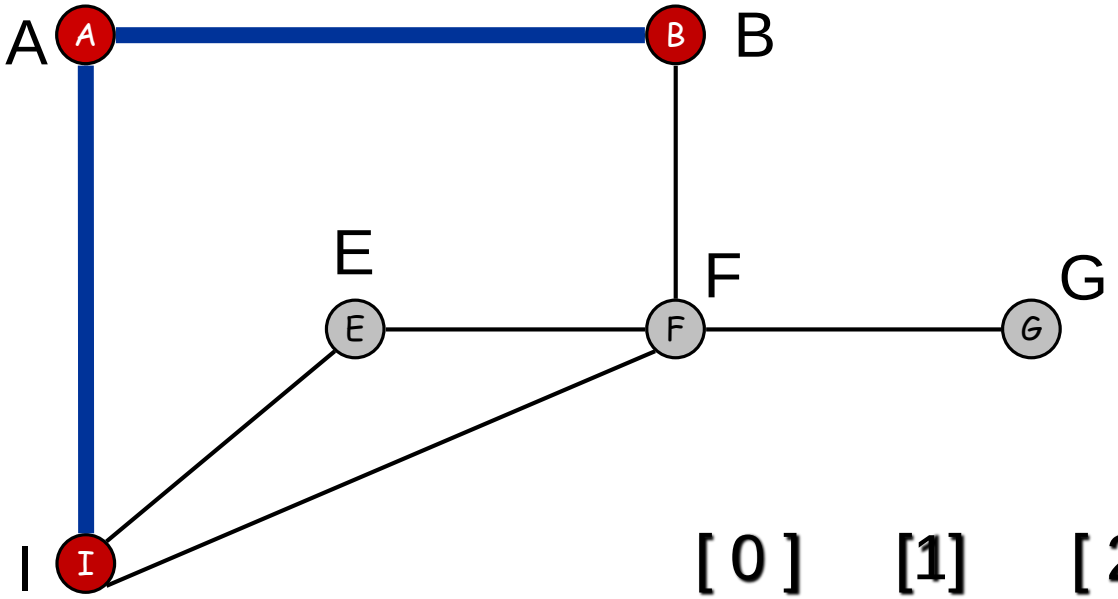
Queue

Visited:

Breadth First Search (BFS) Traversal



Breadth First Search (BFS) Traversal



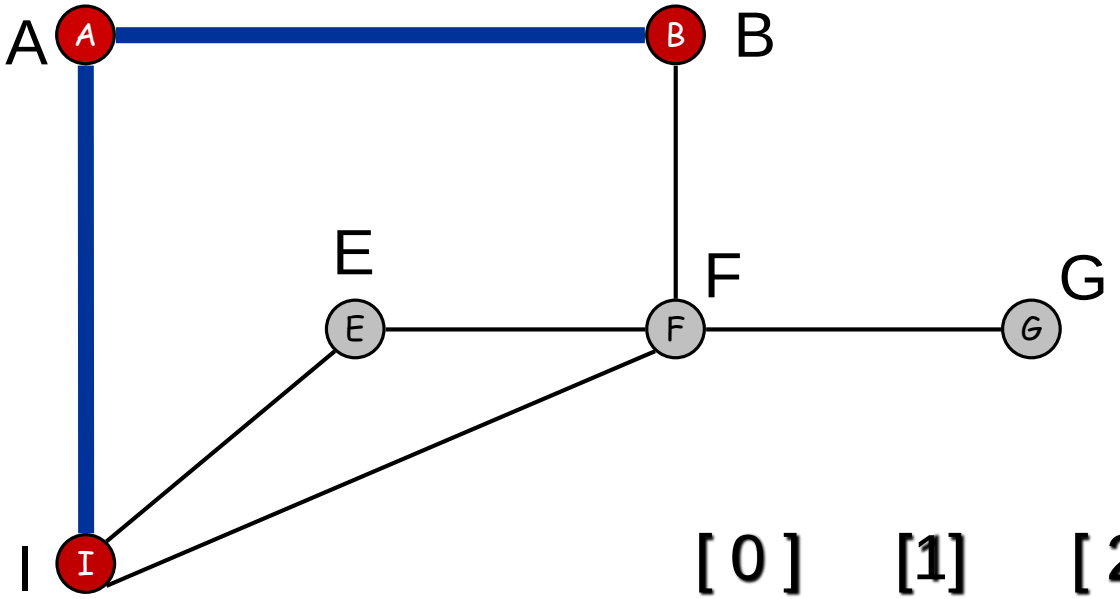
0 front

1 rear

	[0]	[1]	[2]	[3]	[4]	[5]
Queue	B	I				

Visited:	A								

Breadth First Search (BFS) Traversal



0 front

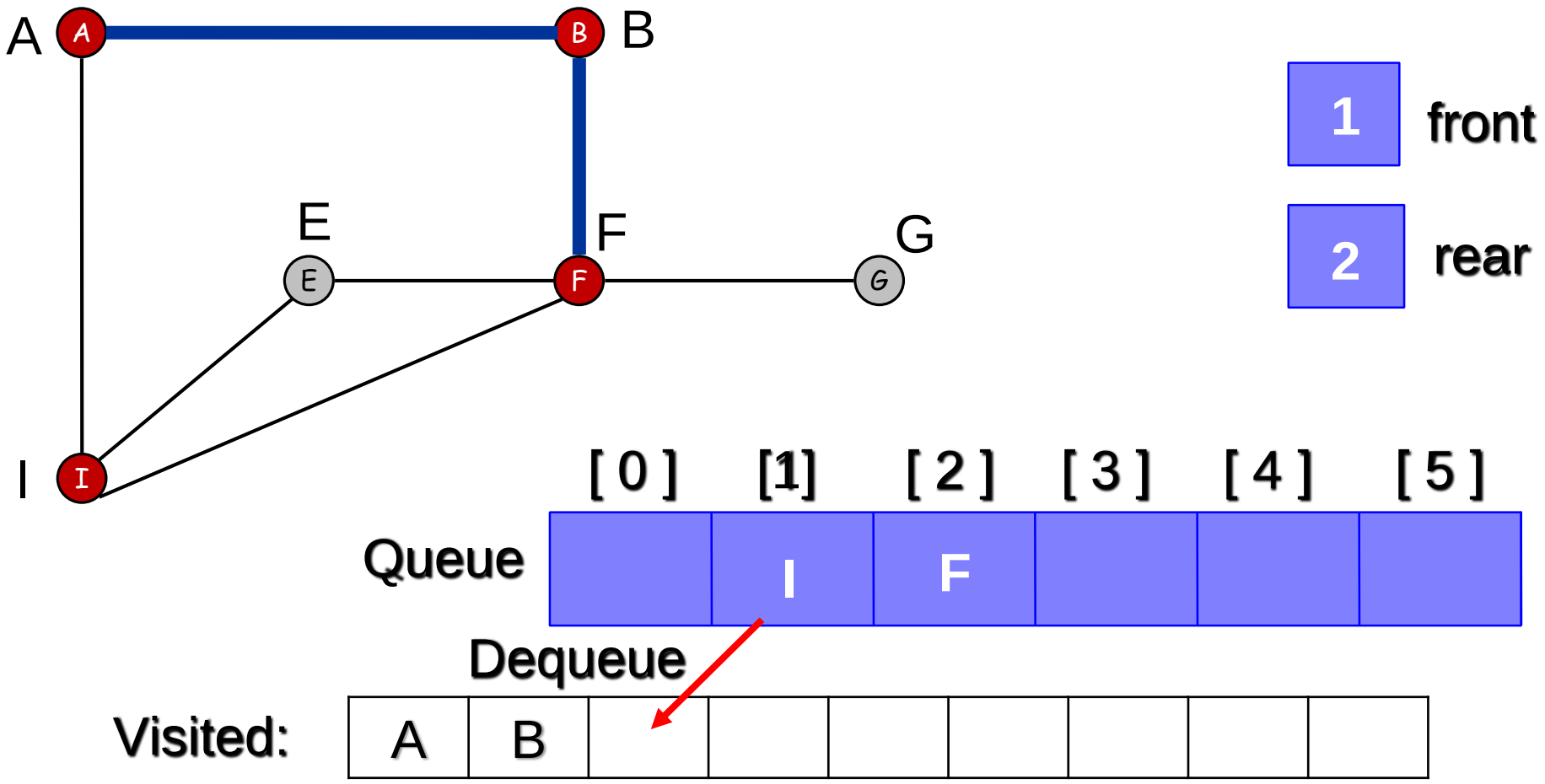
1 rear

	[0]	[1]	[2]	[3]	[4]	[5]
Queue	B	I				

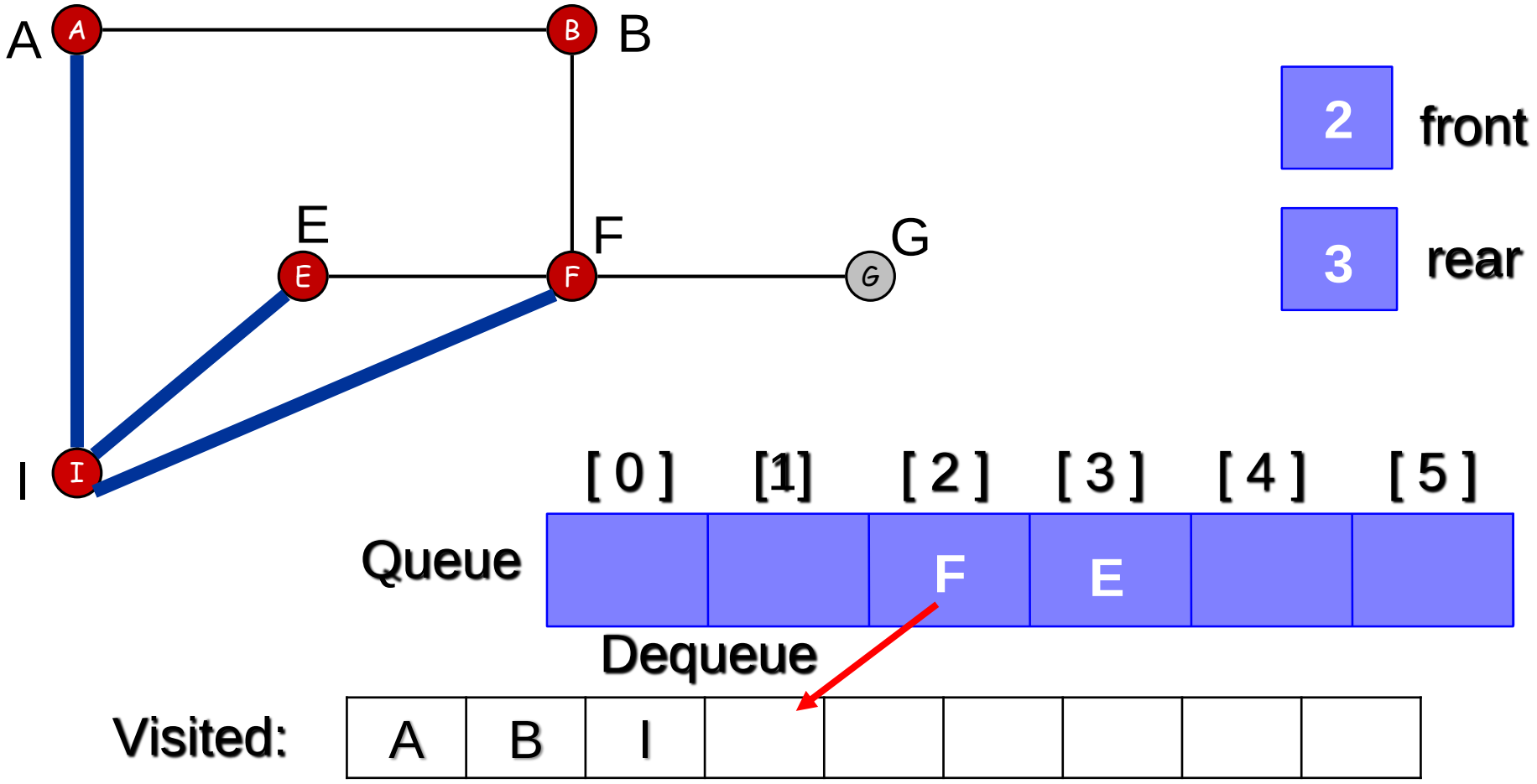
Dequeue

Visited:	A								

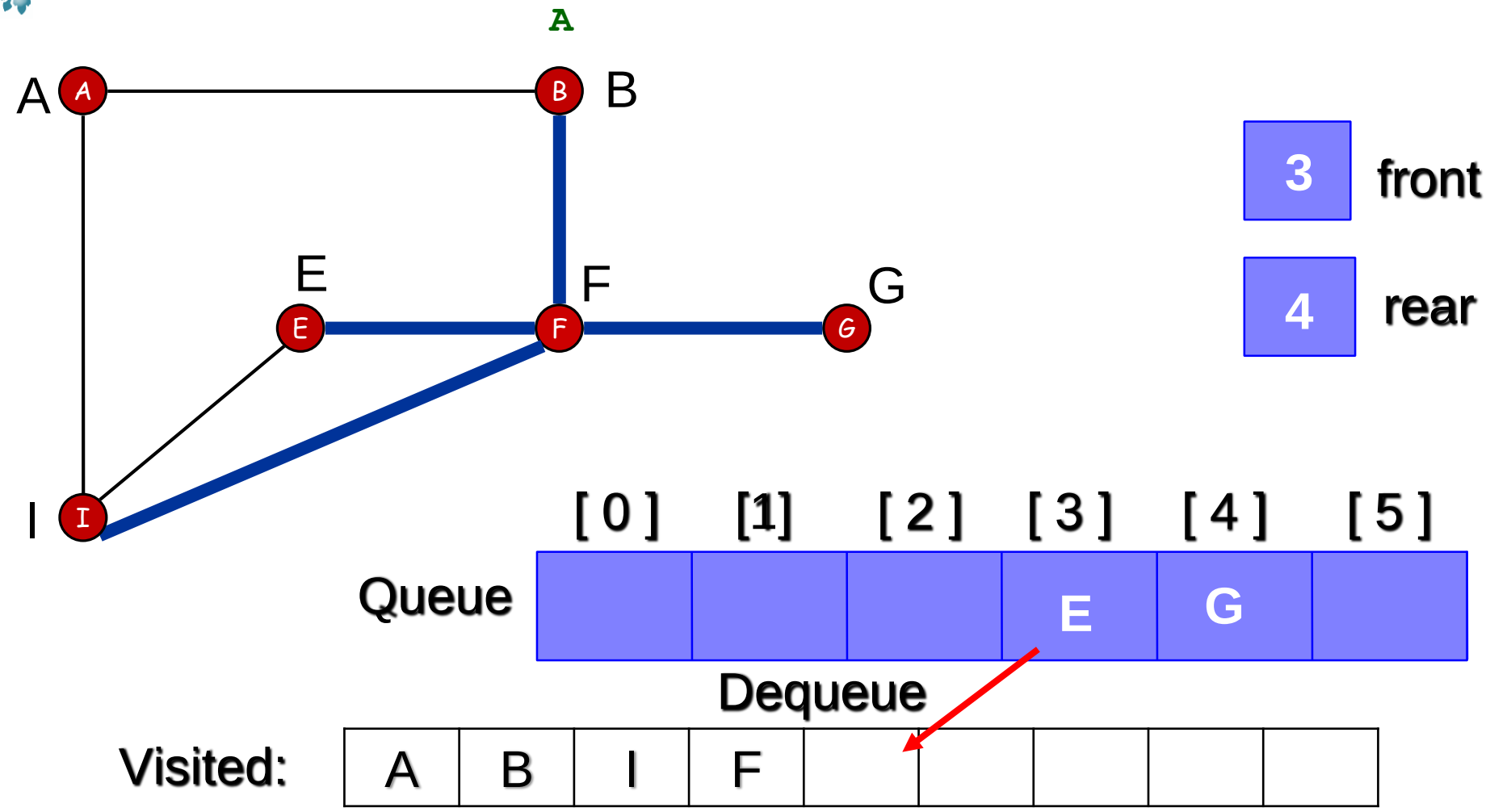
Breadth First Search (BFS) Traversal



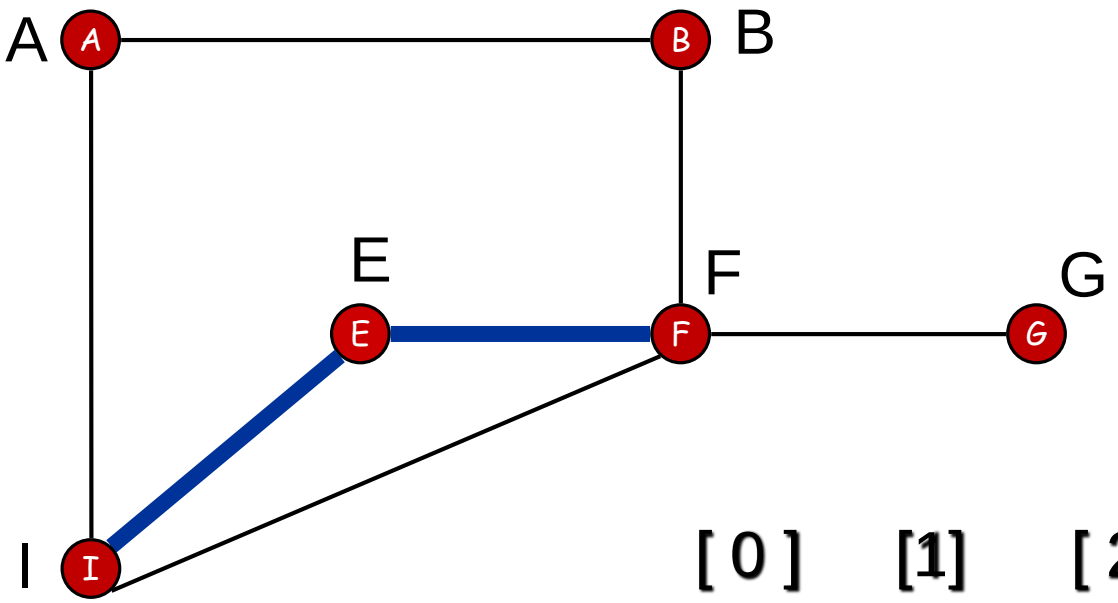
Breadth First Search (BFS) Traversal



Breadth First Search (BFS) Traversal



Breadth First Search (BFS) Traversal



4 front

4 rear

[0] [1] [2] [3] [4] [5]

Queue

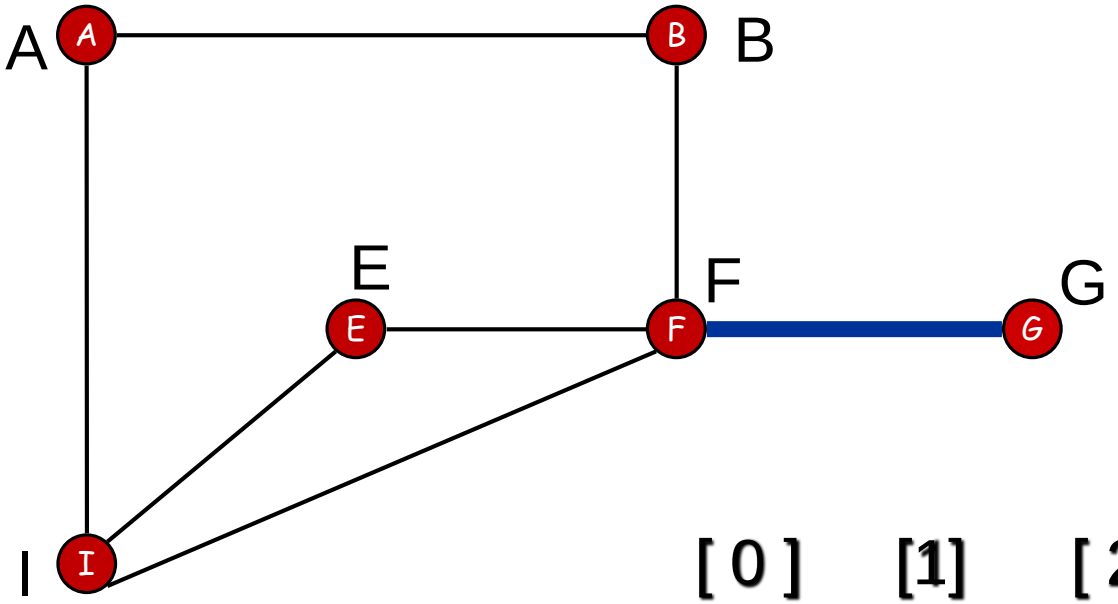
G

Dequeue

Visited:

A	B	I	F	E				

Breadth First Search (BFS) Traversal



0 front

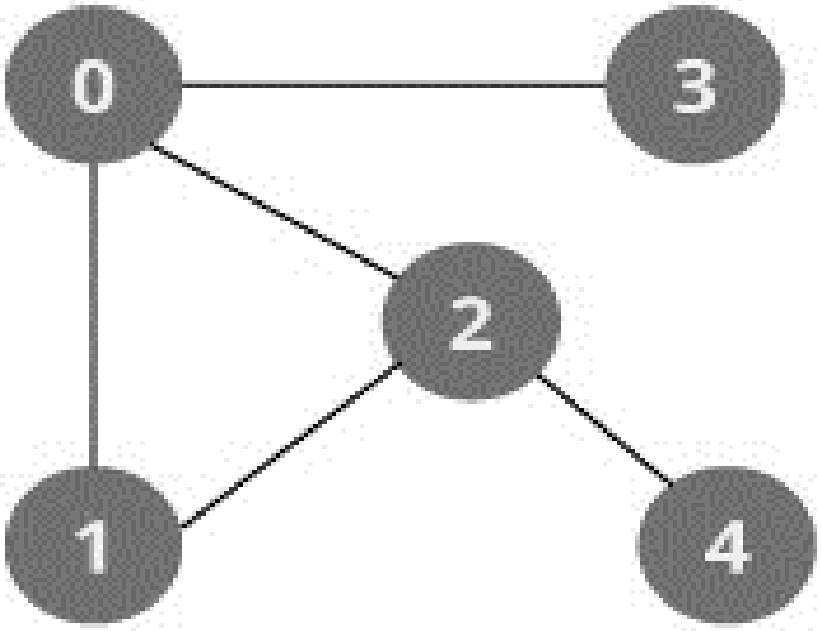
0 rear

[0] [1] [2] [3] [4] [5]

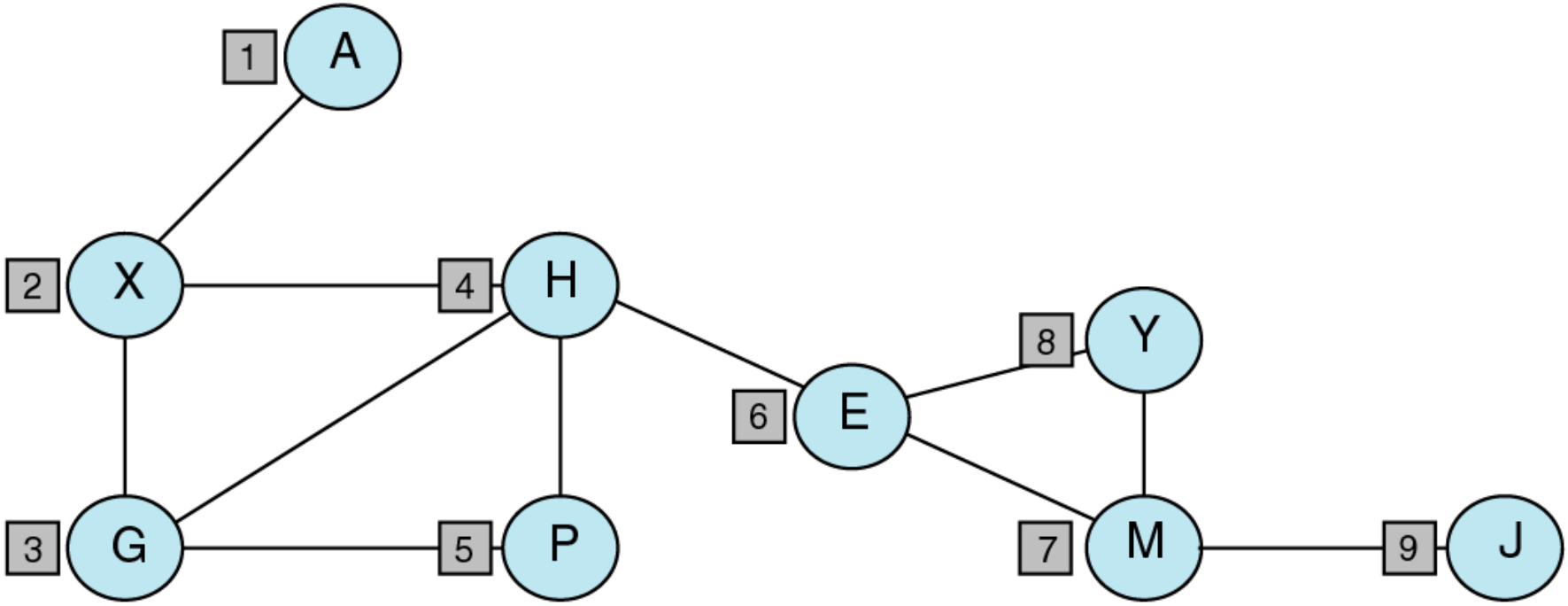
Queue

Visited:	A	B	I	F	E	G			

Breadth First Search (BFS) Traversal

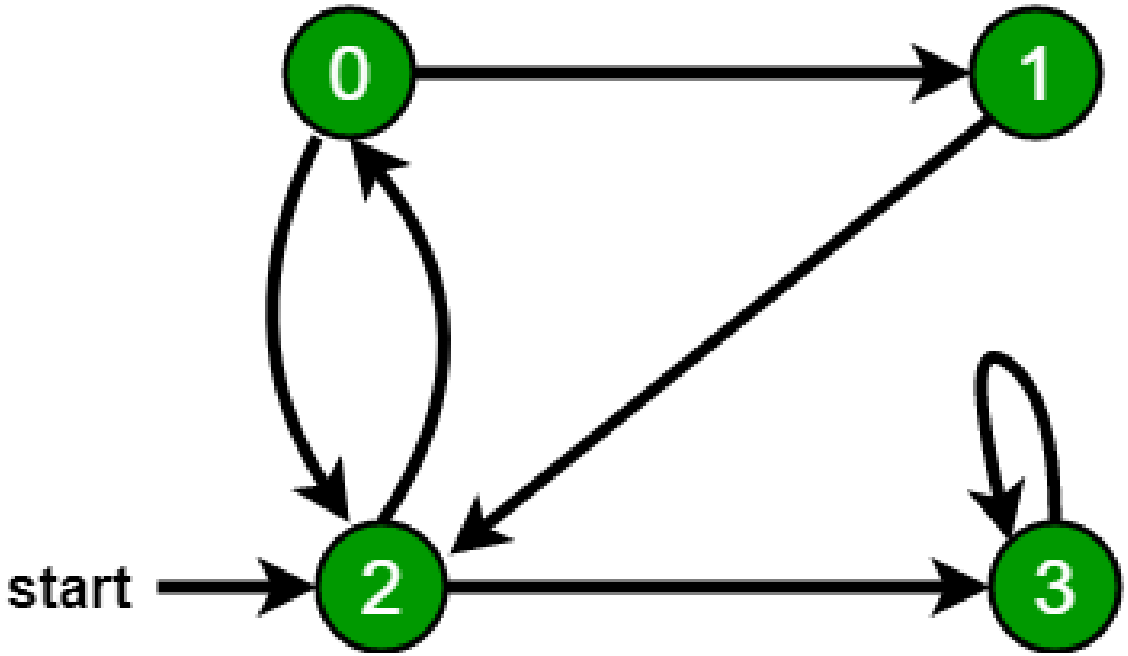


Breadth First Search (BFS) Traversal



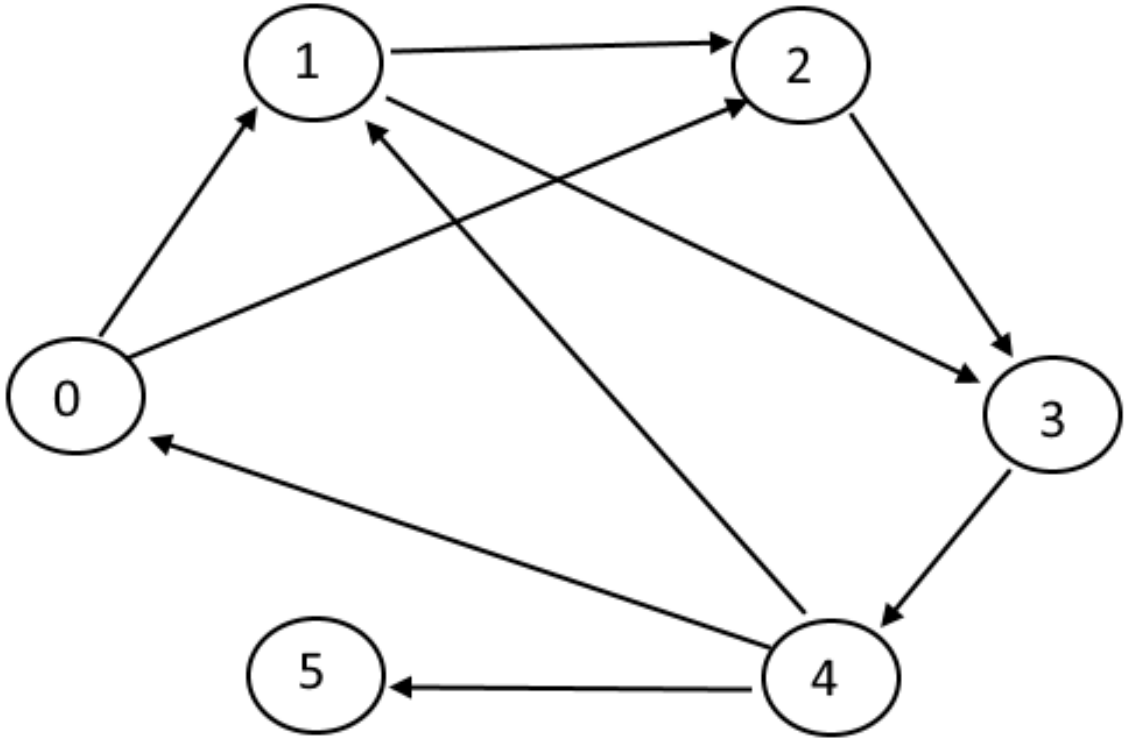
(a) The graph

BFS Practice Sample#1



BFS Traversal: 2, 0, 3, 1

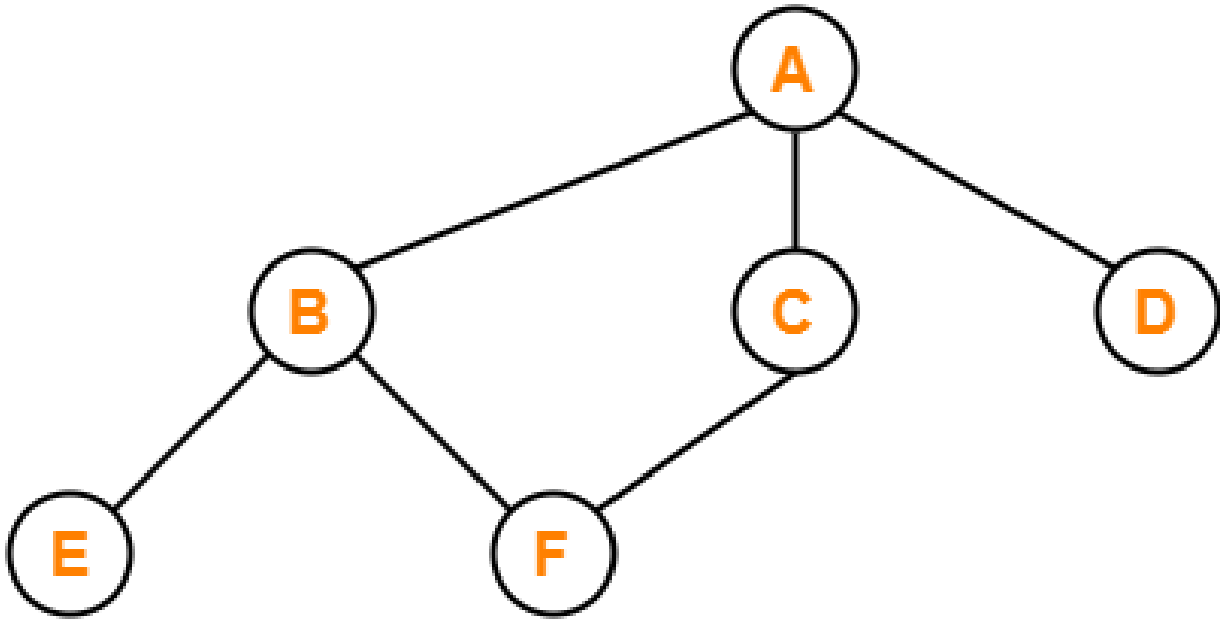
BFS Practice Sample#2



BFS starting from Node 0

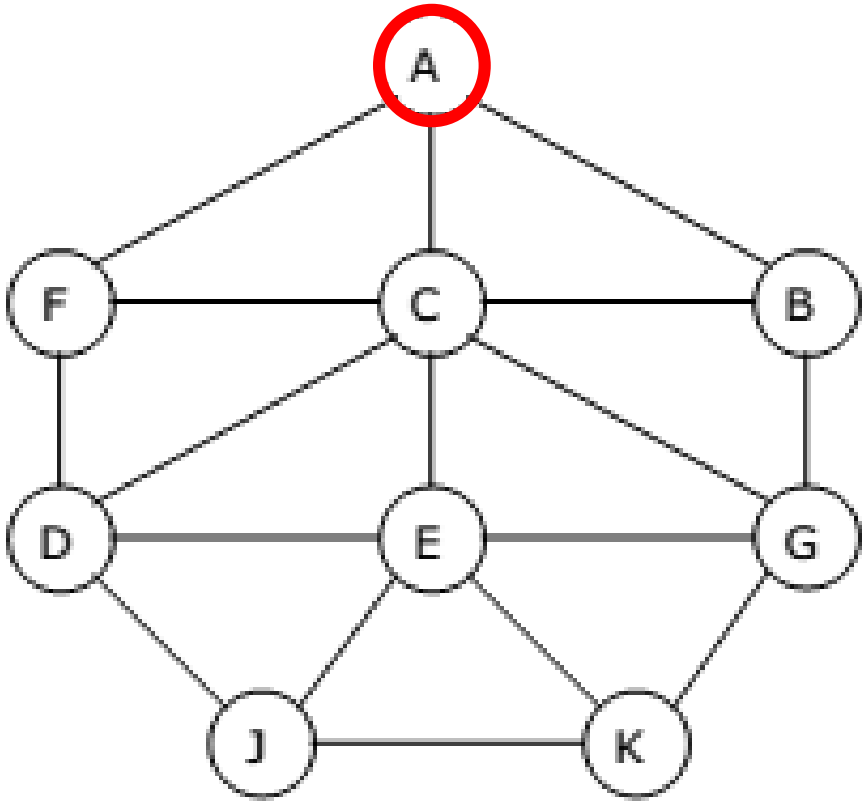
BFS Traversal: 0, 1, 2, 3, 4, 5

BFS Practice Sample#3



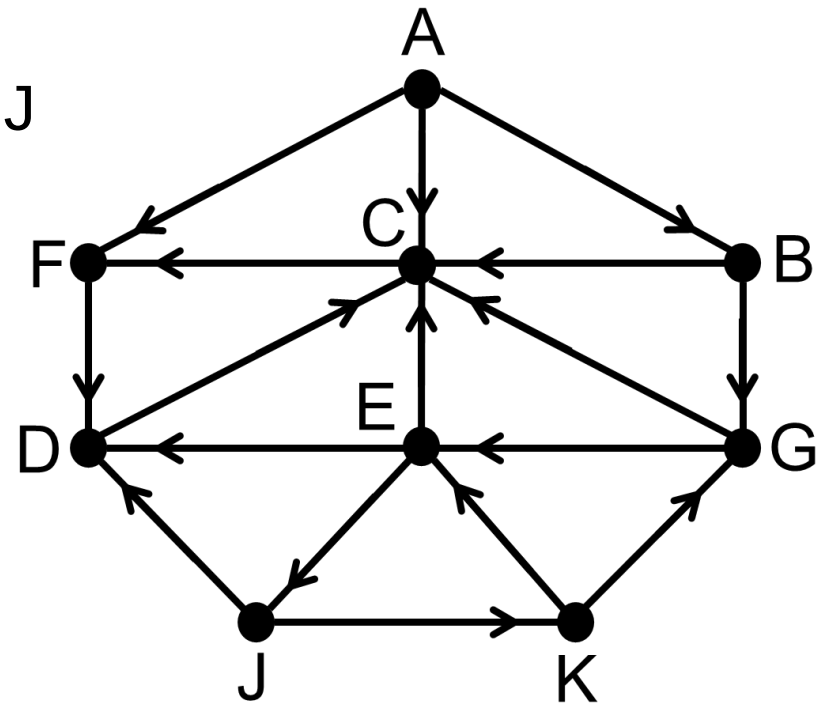
BFS Traversal: A, B, C, D, E, F

BFS Practice Sample#4



BFS Algorithm for Shortest Paths

Find the Shortest Path from A to J



0 **front**

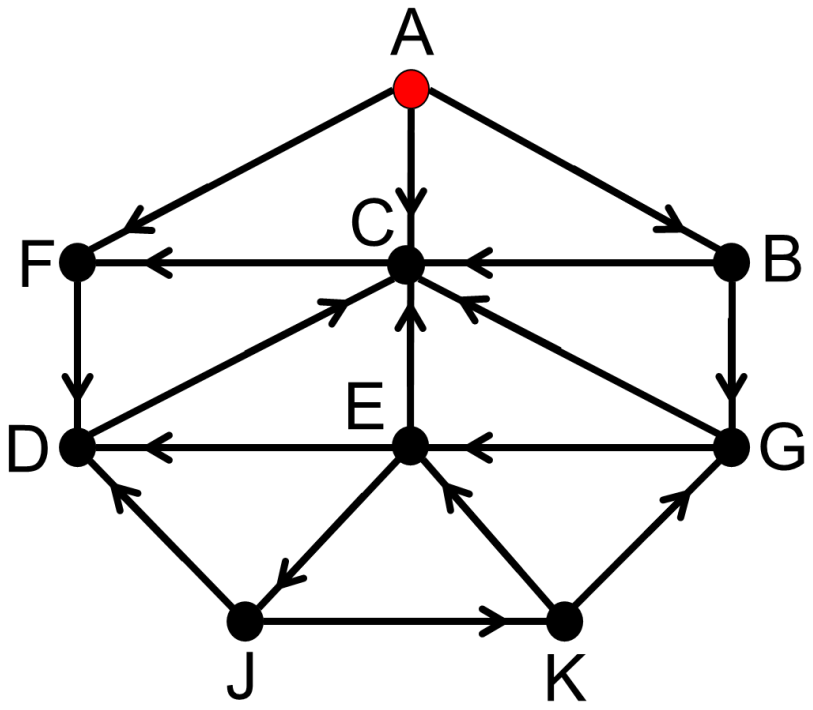
0 **rear**

	1	2	3	4	5	6	7	8	9
Queue									
Origin									

BFS Algorithm for Shortest Paths

1 front

1 rear

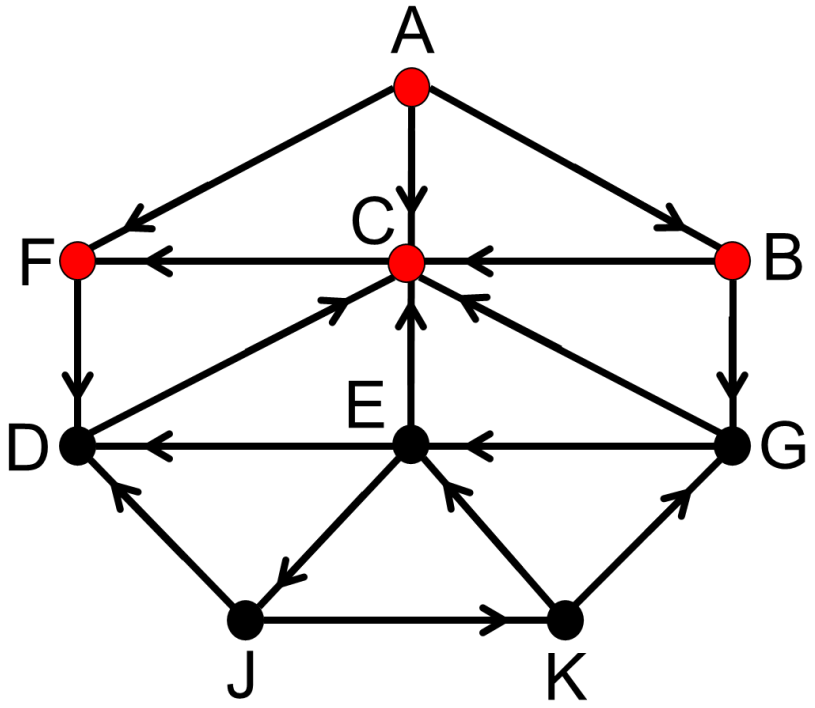


	1	2	3	4	5	6	7	8	9
Queue	A								
Origin	ø								

BFS Algorithm for Shortest Paths

2 front

4 rear

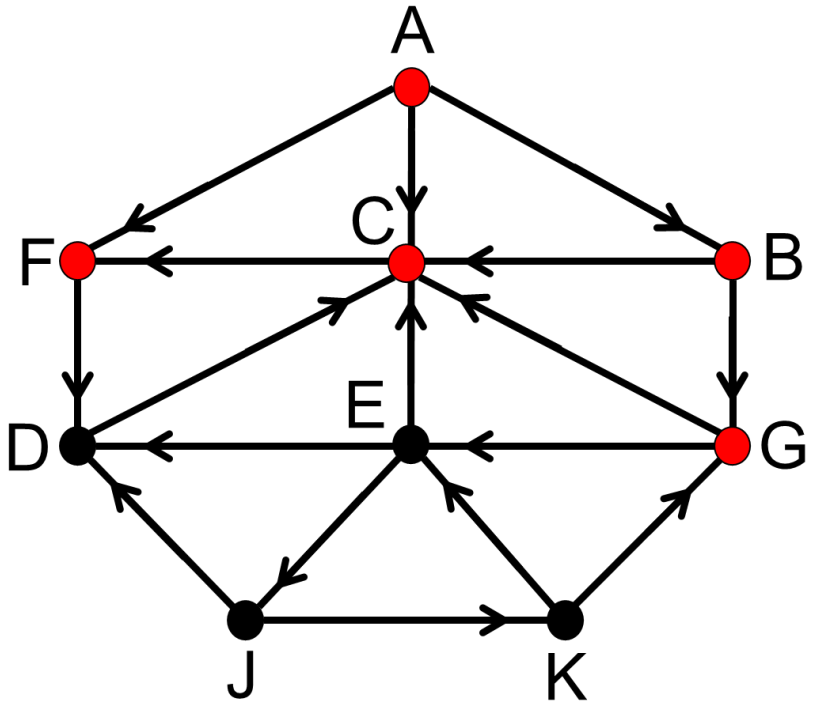


	1	2	3	4	5	6	7	8	9
Queue	A	B	C	F					
Origin	ø	A	A	A					

BFS Algorithm for Shortest Paths

3 front

5 rear

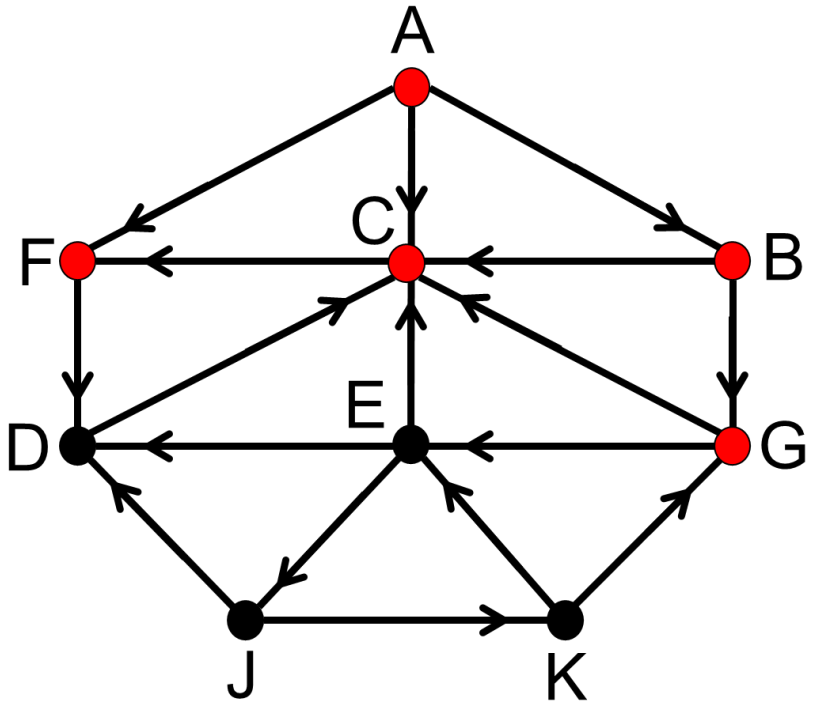


	1	2	3	4	5	6	7	8	9
Queue	A	B	C	F	G				
Origin	ø	A	A	A	B				

BFS Algorithm for Shortest Paths

4 front

5 rear

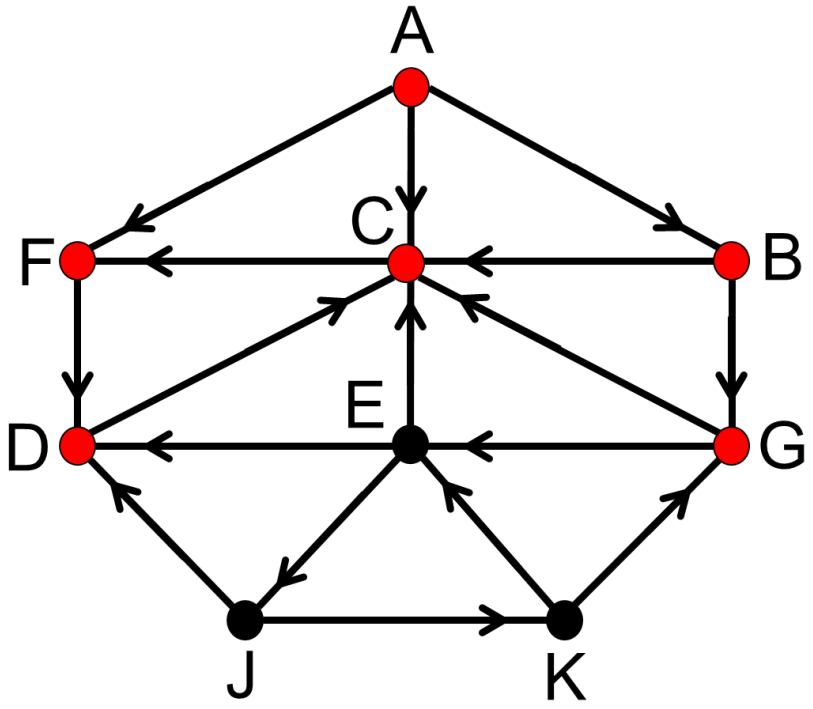


	1	2	3	4	5	6	7	8	9
Queue	A	B	C	F	G				
Origin	ø	A	A	A	B				

BFS Algorithm for Shortest Paths

5 front

6 rear

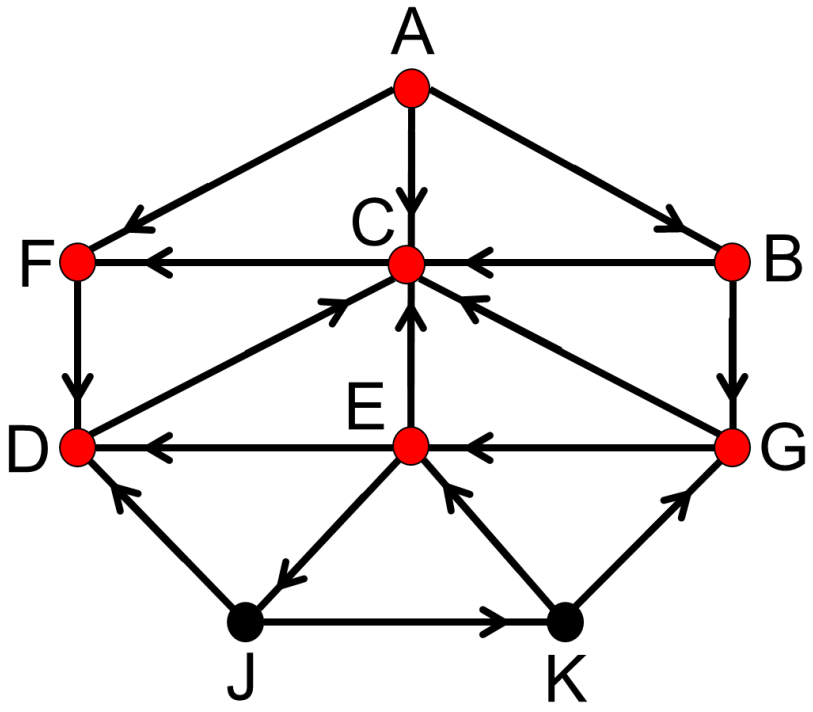


	1	2	3	4	5	6	7	8	9
Queue	A	B	C	F	G	D			
Origin	ø	A	A	A	B	F			

BFS Algorithm for Shortest Paths

6 front

7 rear

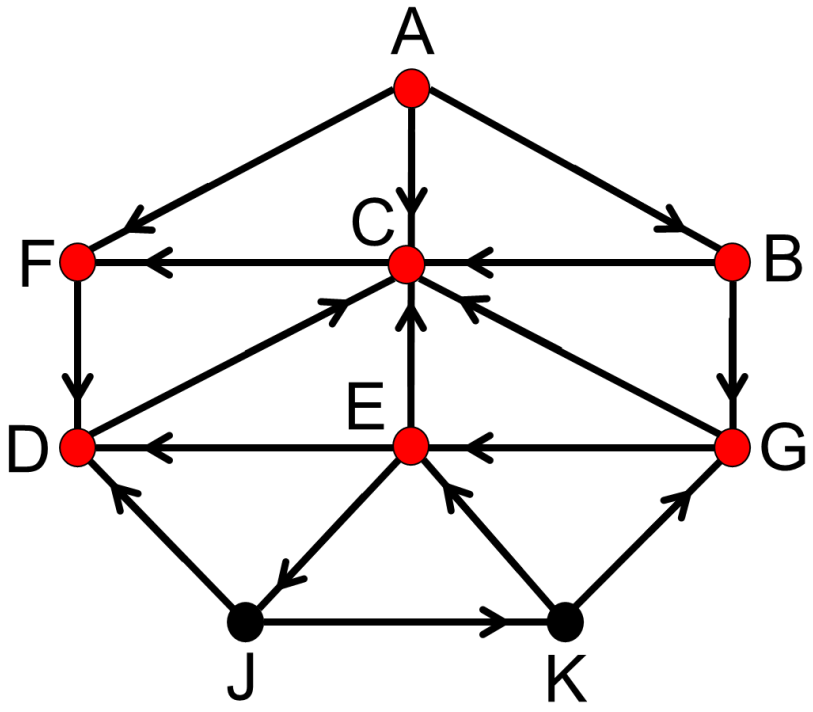


	1	2	3	4	5	6	7	8	9
Queue	A	B	C	F	G	D	E		
Origin	ø	A	A	A	B	F	G		

BFS Algorithm for Shortest Paths

7 **front**

7 **rear**

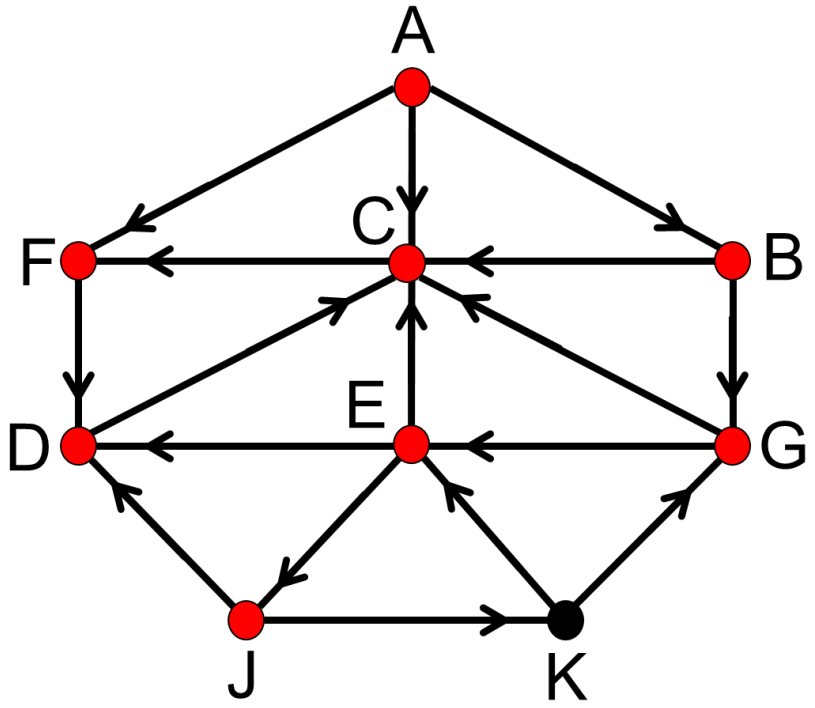


	1	2	3	4	5	6	7	8	9
Queue	A	B	C	F	G	D	E		
Origin	ø	A	A	A	B	F	G		

BFS Algorithm for Shortest Paths

8 front

8 rear



	1	2	3	4	5	6	7	8	9
Queue	A	B	C	F	G	D	E	J	
Origin	ø	A	A	A	B	F	G	E	

BFS Algorithm for Shortest Paths

8 **front**

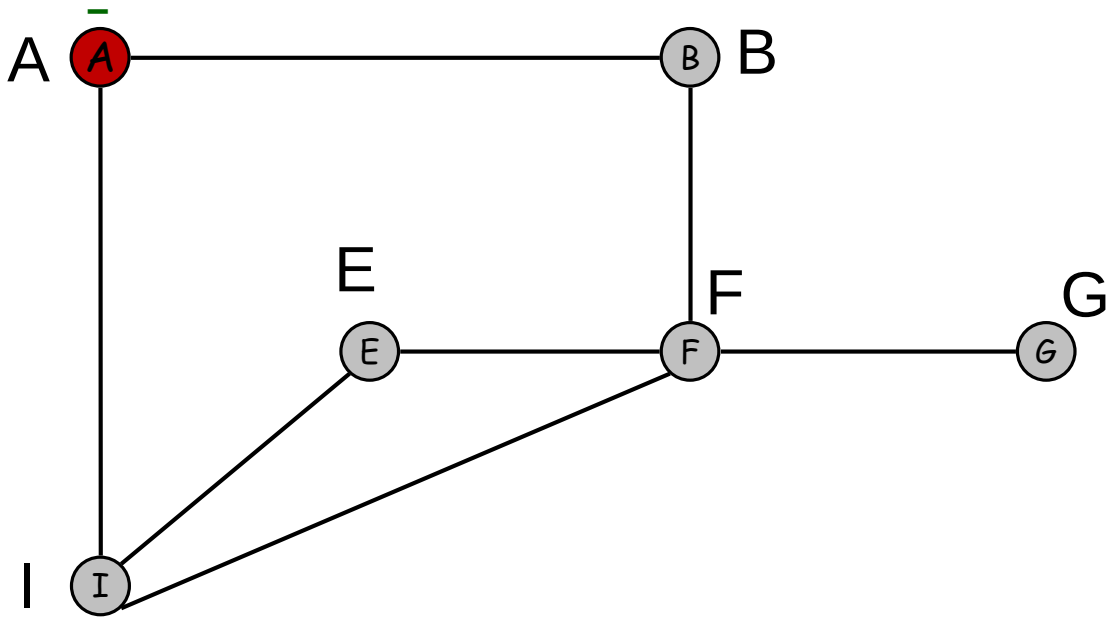
8 **rear**

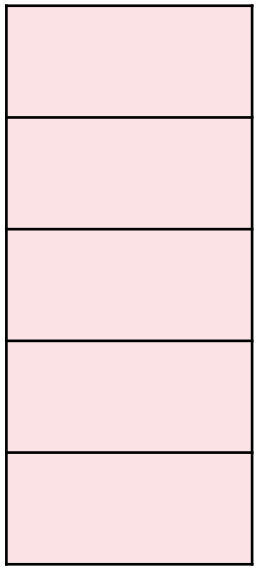
	1	2	3	4	5	6	7	8	9
Queue	A	B	C	F	G	D	E	J	
Origin	ø	A	A	A	B	F	G	E	

	1	2	3	4	5	6	7	8	9
Queue	A	B	C	F	G	D	E	J	
Origin	ø	A	A	A	B	F	G	E	

J ← E ← G ← B ← A

Depth First Search (DFS) Traversal

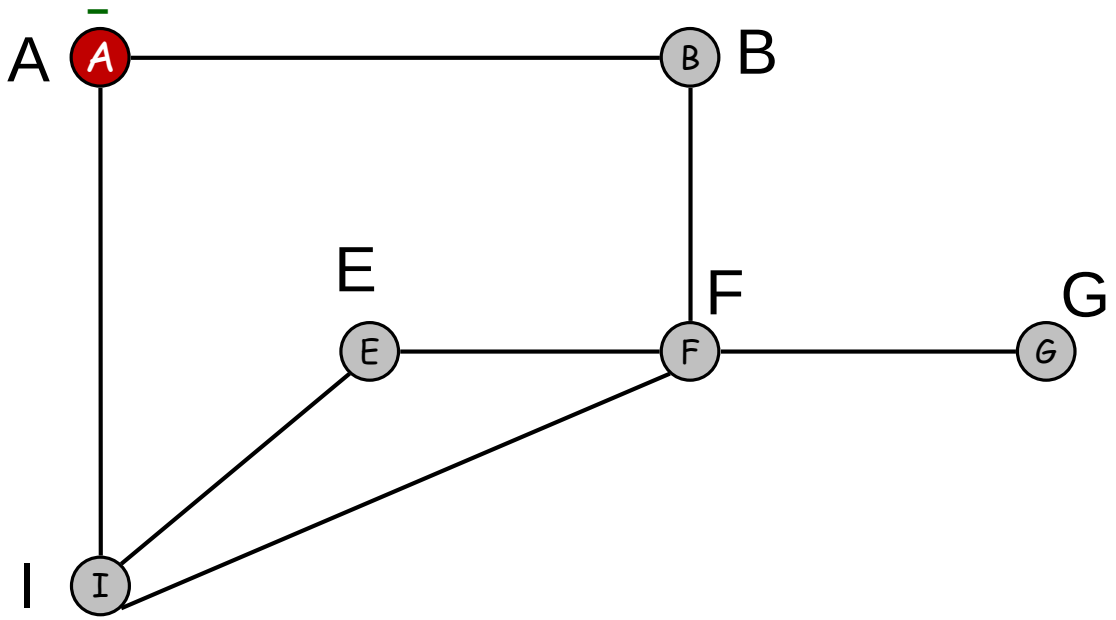


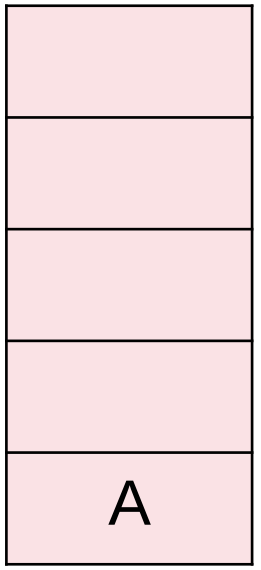


STACK

Visited:

Depth First Search (DFS) Traversal

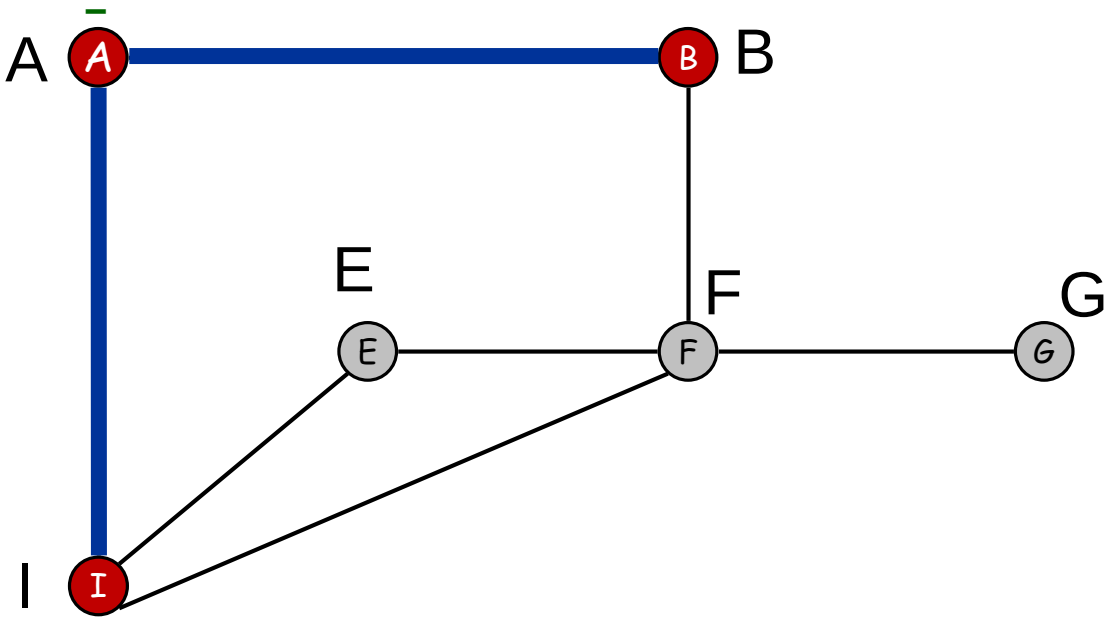


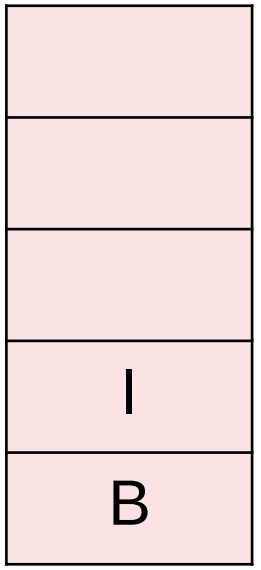


STACK

Visited:

Depth First Search (DFS) Traversal



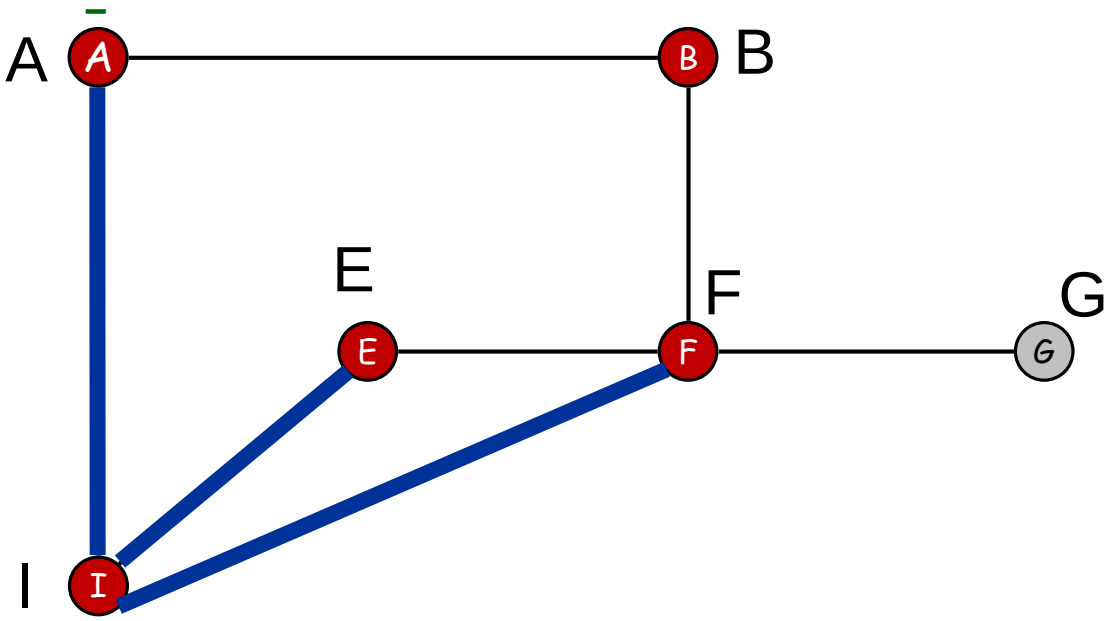


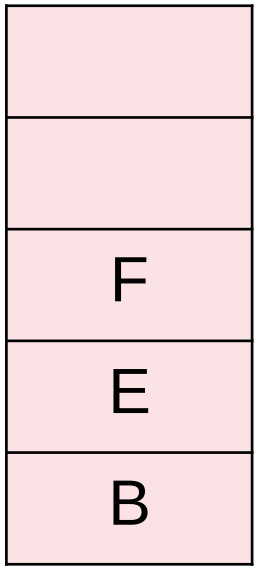
STACK

Visited:

A								

Depth First Search (DFS) Traversal



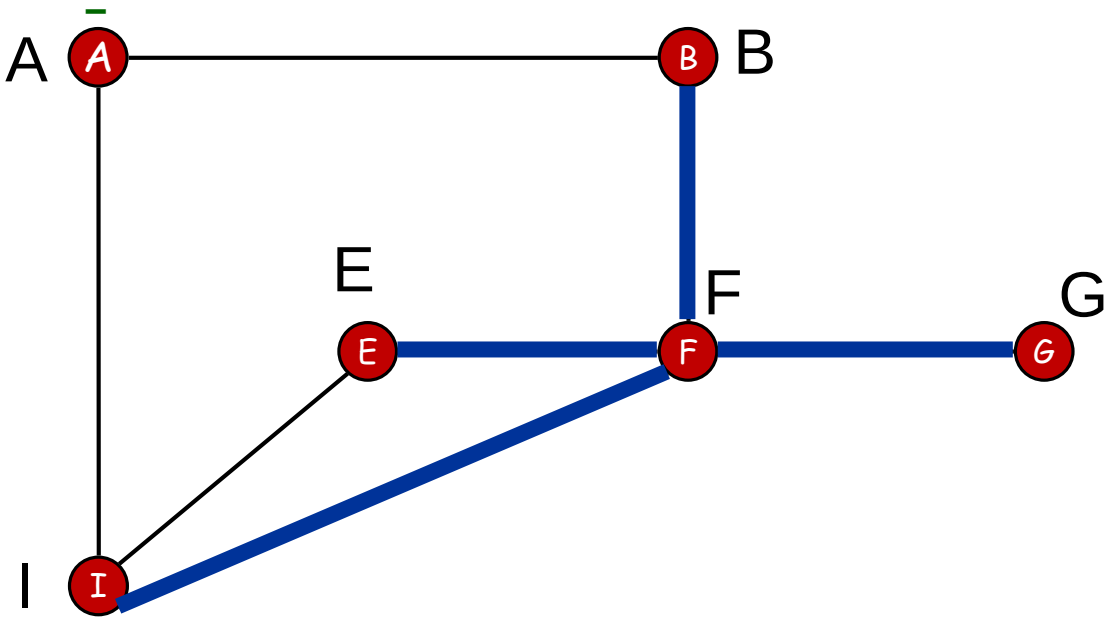


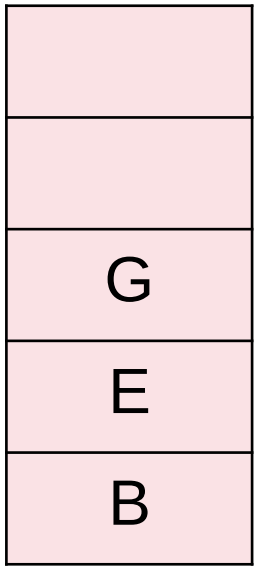
STACK

Visited:

A	I							

Depth First Search (DFS) Traversal



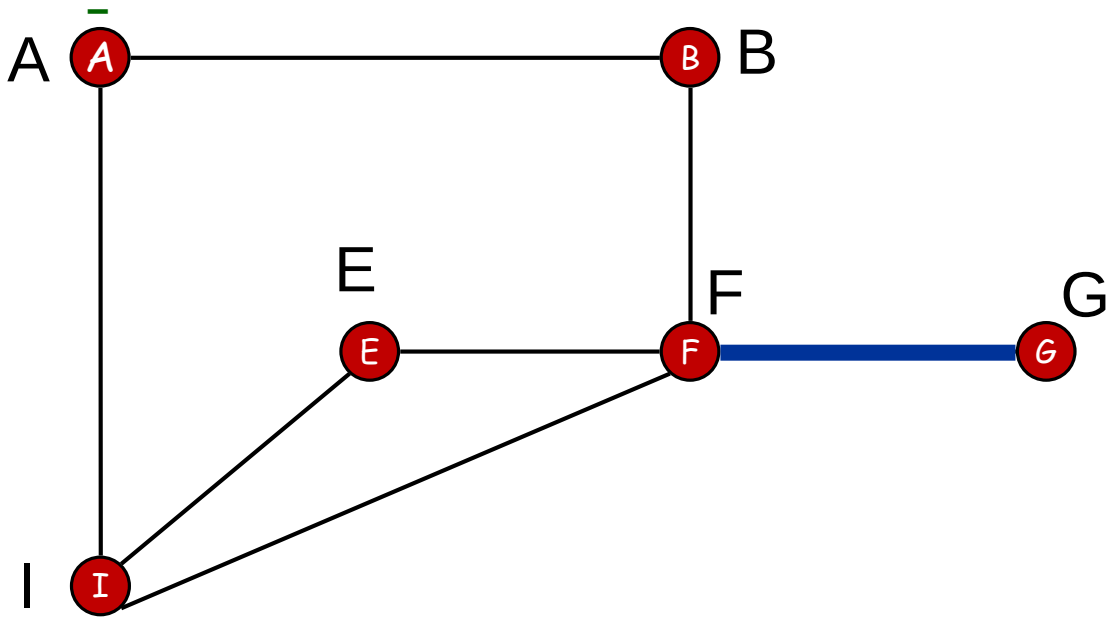


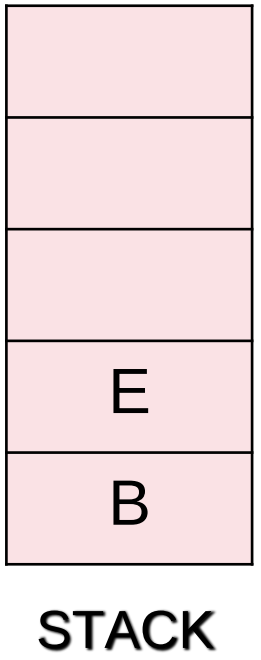
STACK

Visited:

A	I	F						

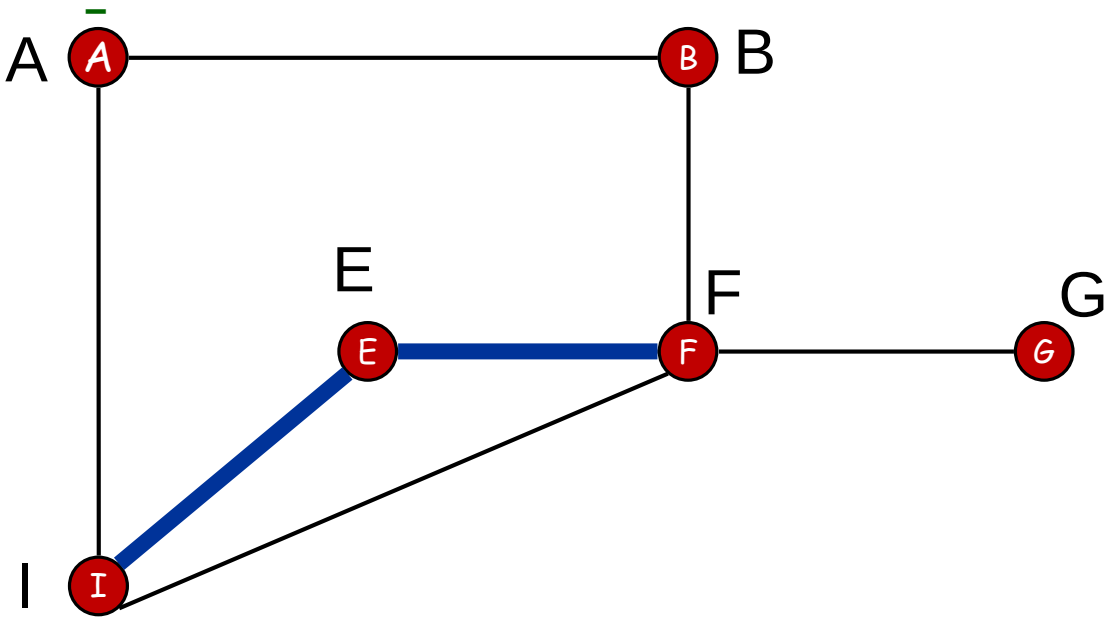
Depth First Search (DFS) Traversal

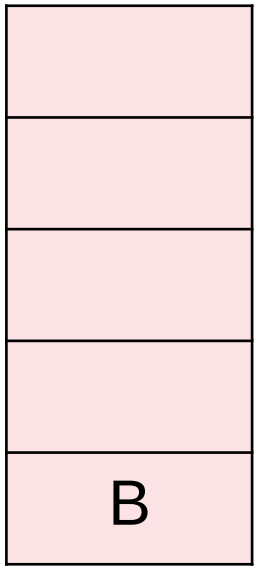




Visited:	A	I	F	G					

Depth First Search (DFS) Traversal

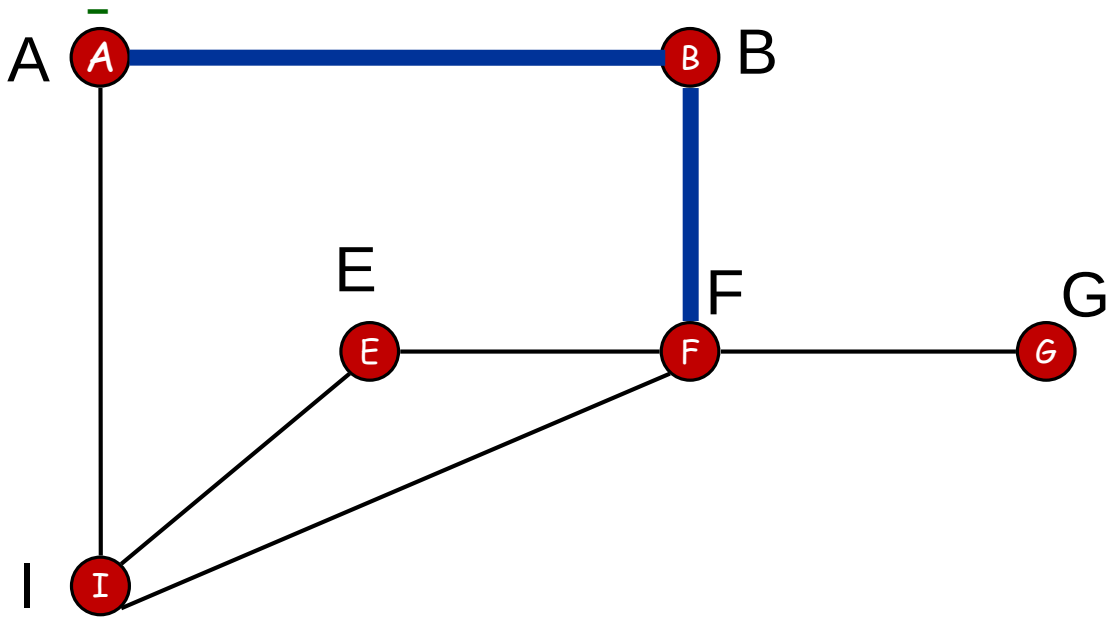


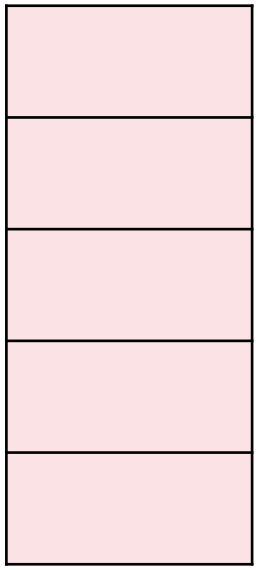


STACK

Visited:	A	I	F	G	E				

Depth First Search (DFS) Traversal

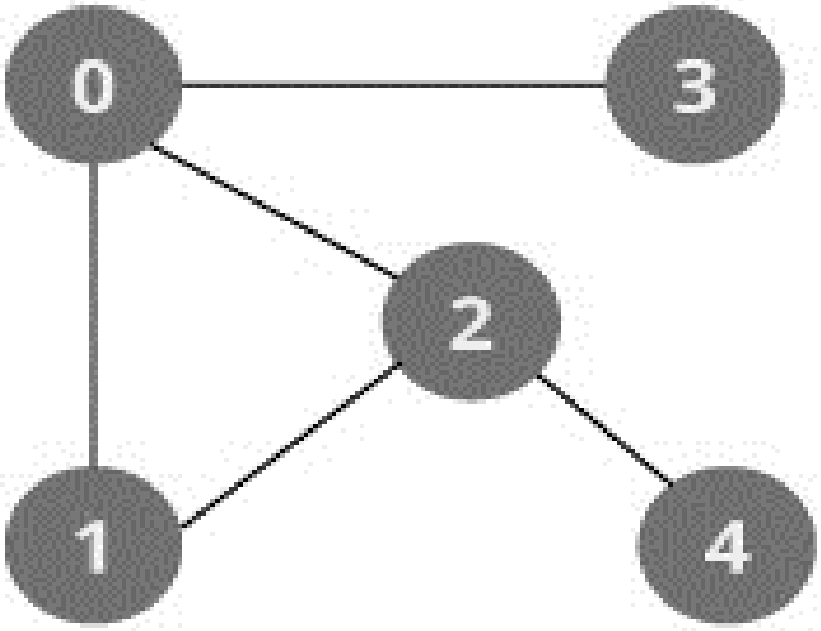




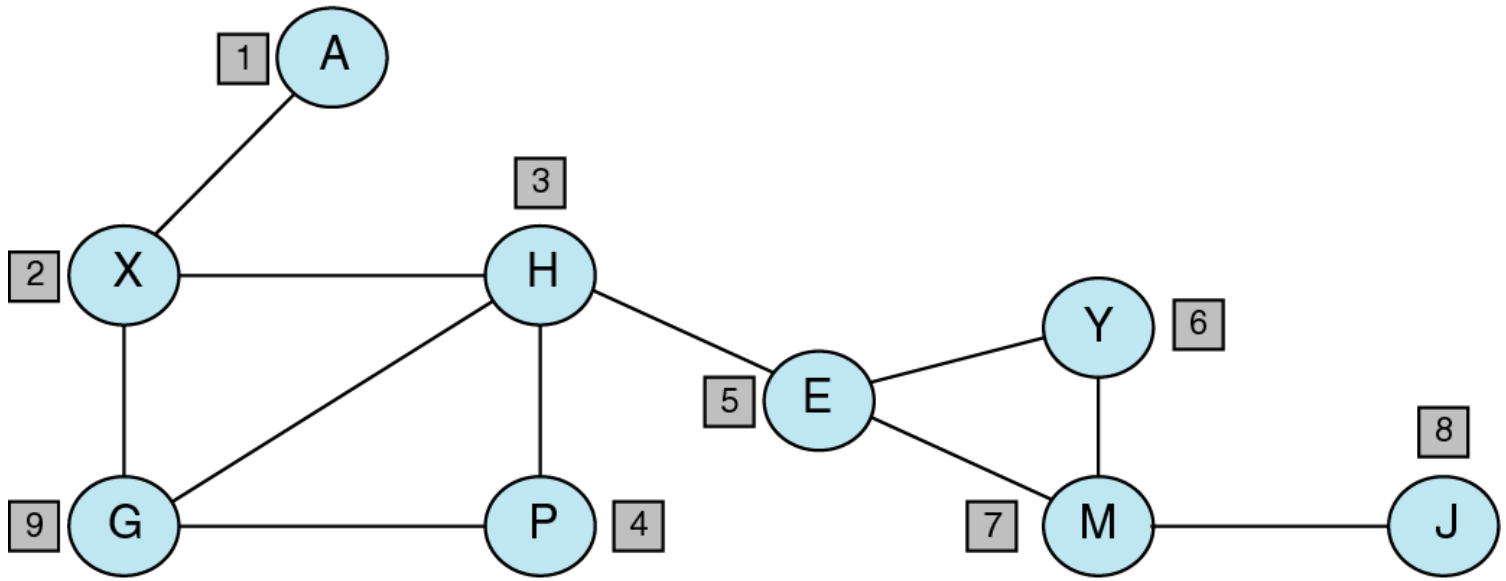
STACK

Visited:	A	I	F	G	E	B			

Depth First Search (DFS) Traversal

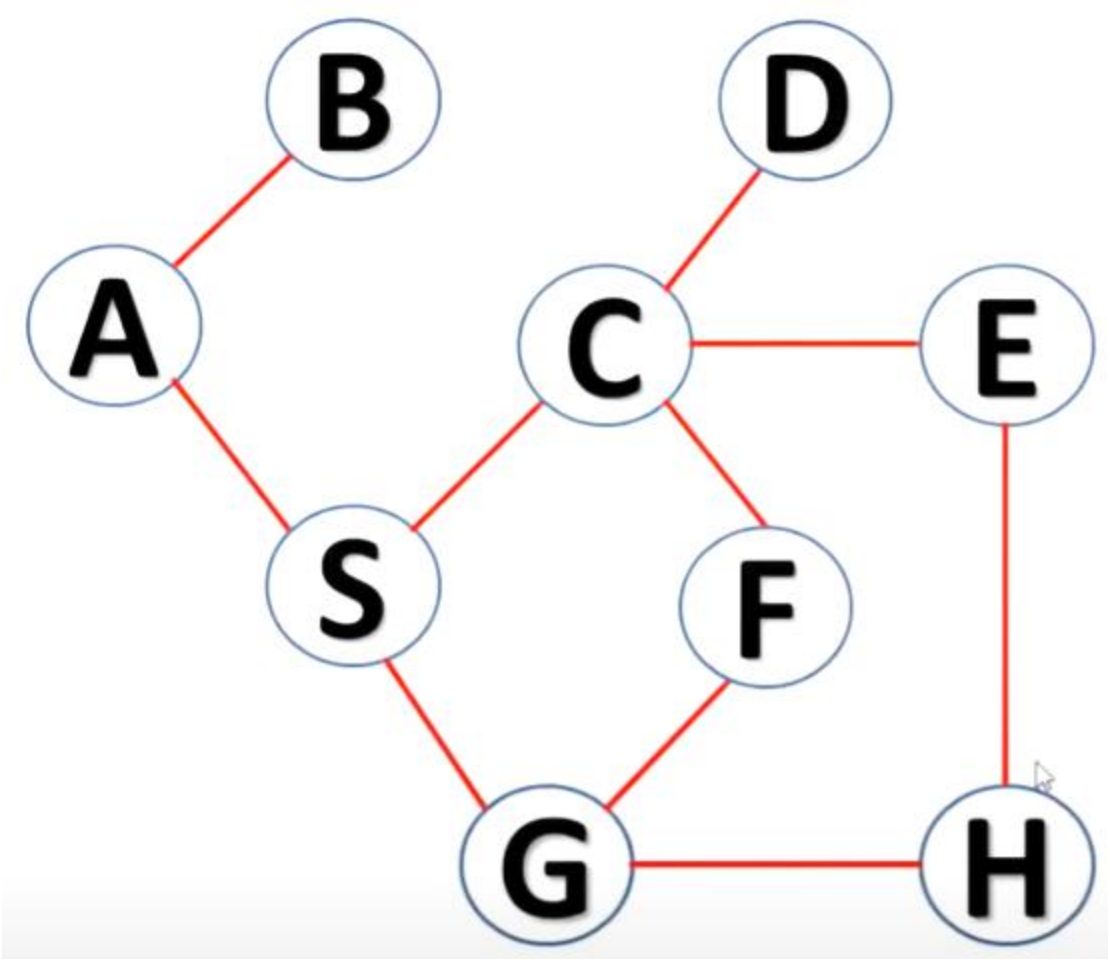


Depth First Search (DFS) Traversal

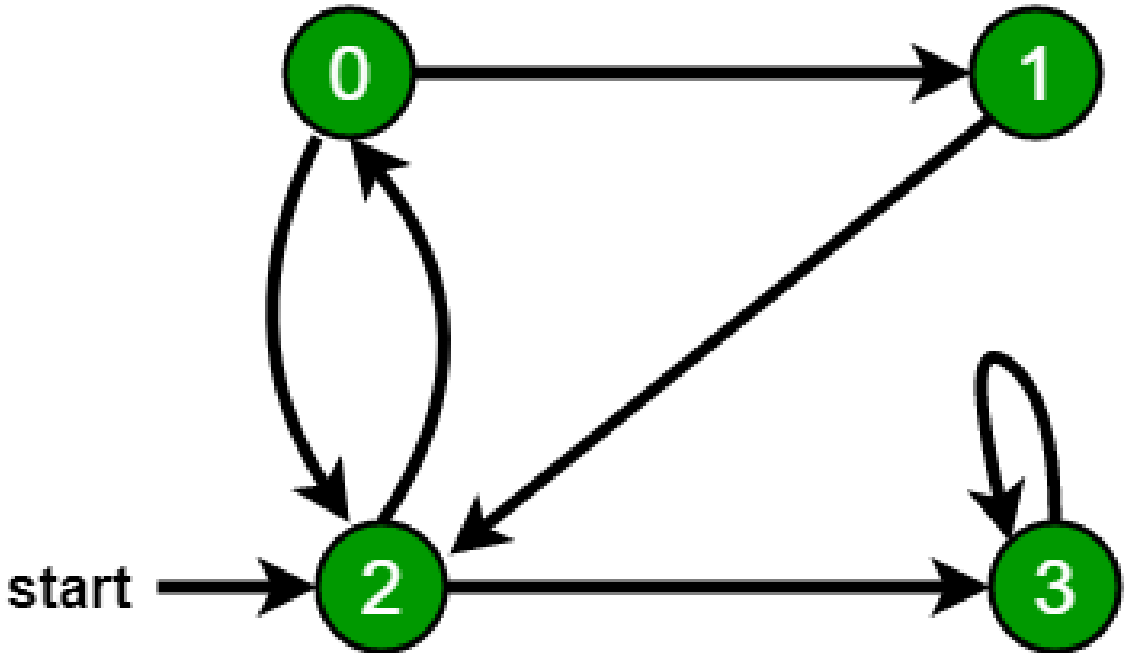


(a) The graph

Depth First Search (DFS) Traversal

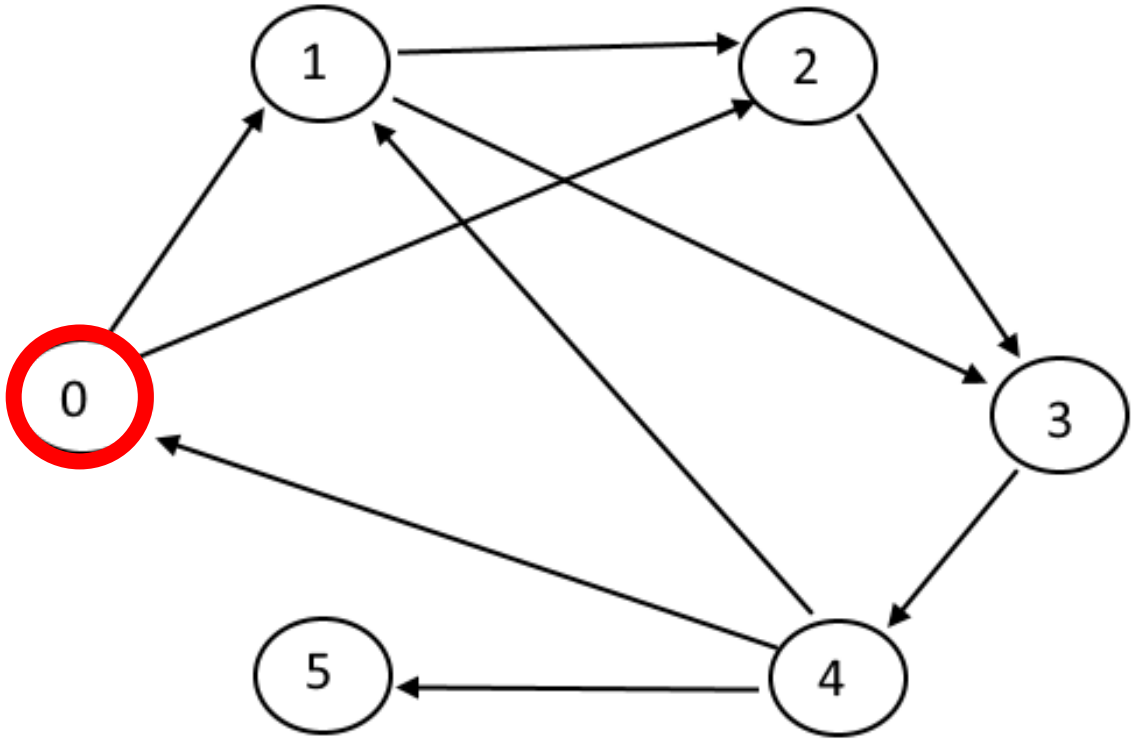


DFS Practice Sample#1



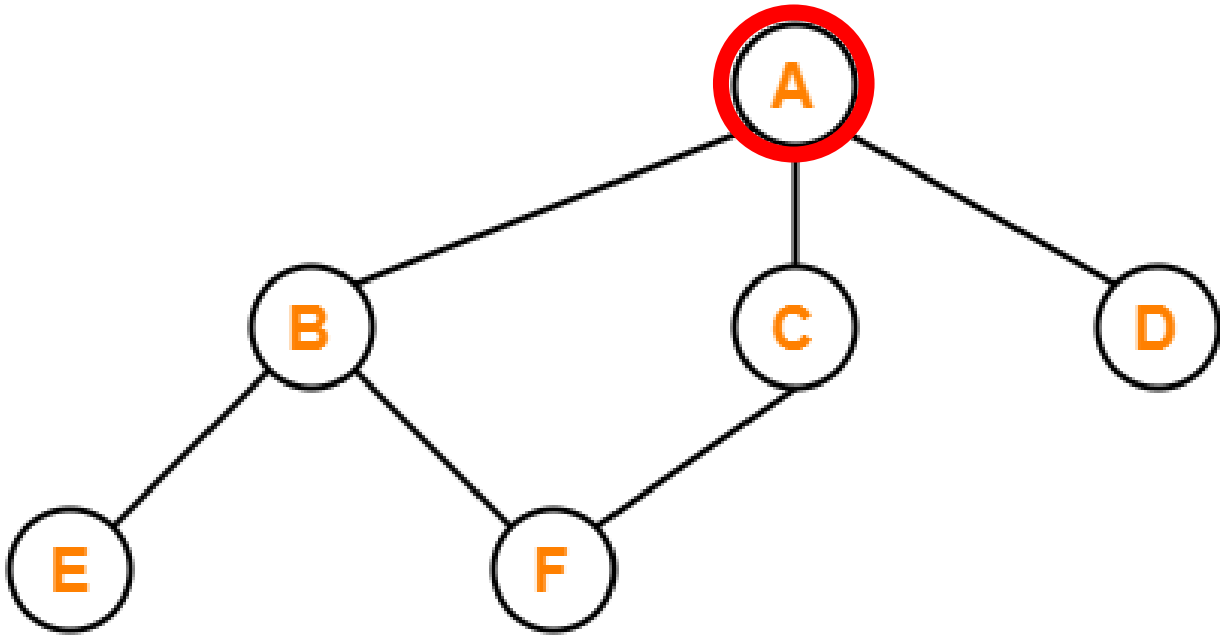
DFS Traversal: 2, 0, 1, 3

DFS Practice Sample#2



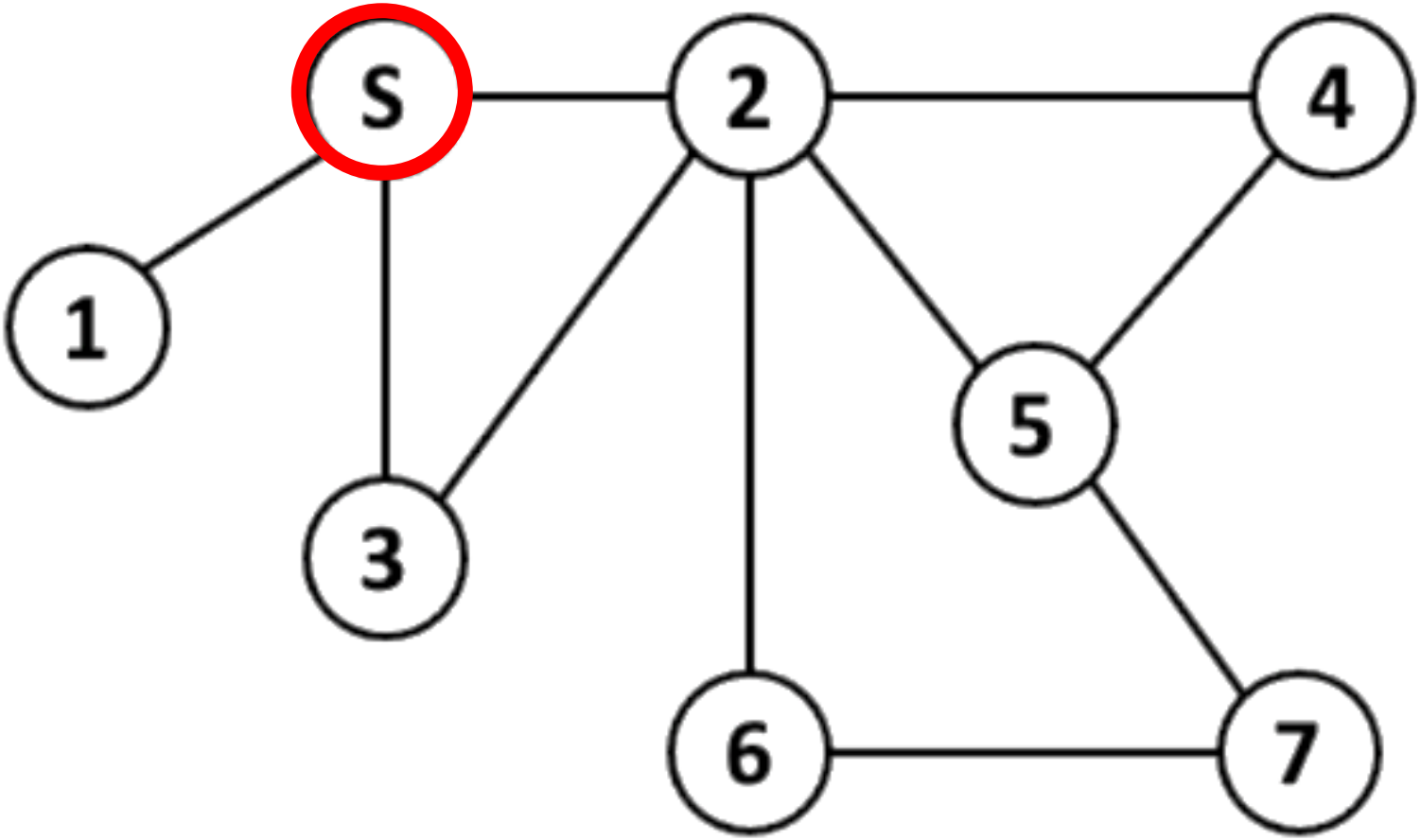
DFS Traversal: 0, 2, 3, 4, 5, 1

DFS Practice Sample#3

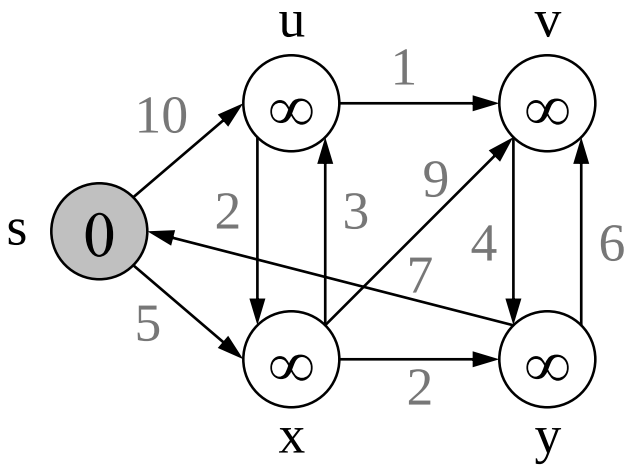


DFS Traversal: A, D, C, F, B, E

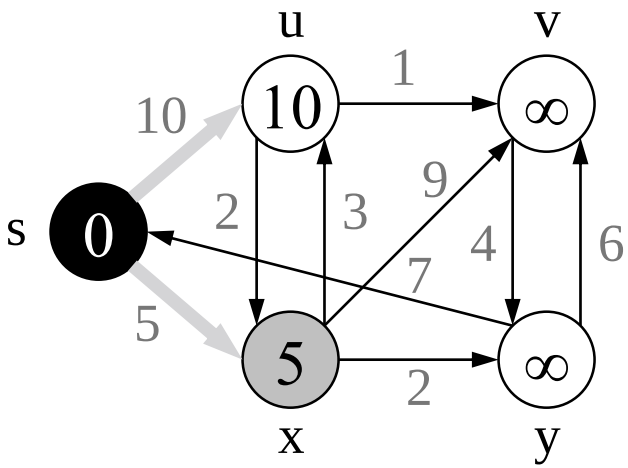
DFS Practice Sample#4



Dijkstra's Algorithm for Shortest Paths

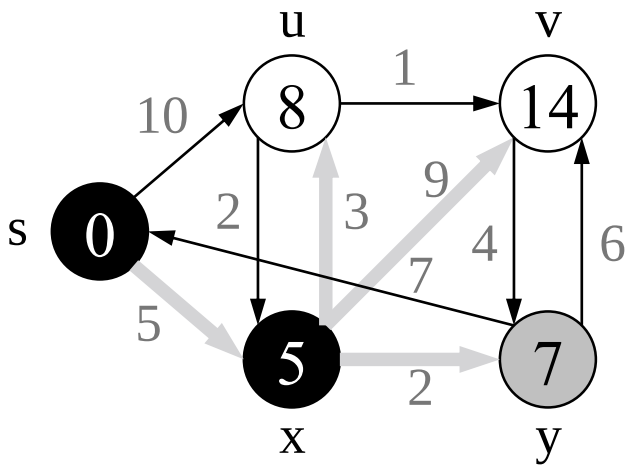


(a)

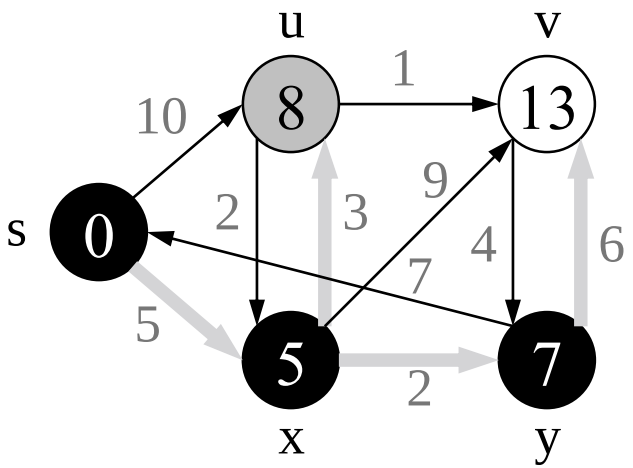


(b)

Dijkstra's Algorithm for Shortest Paths

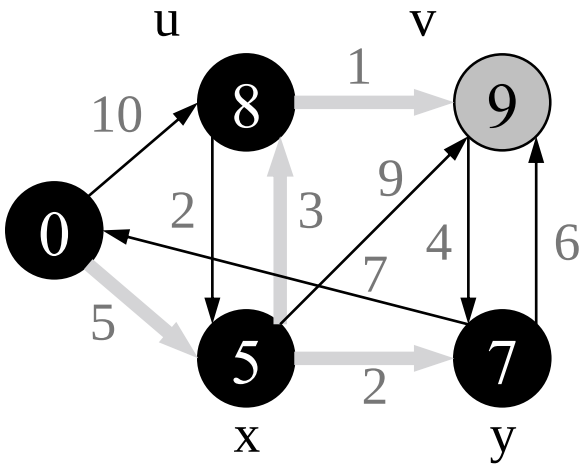


(c)

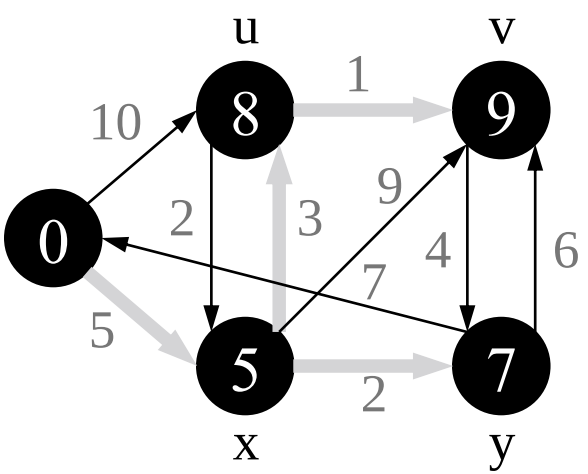


(d)

Dijkstra's Algorithm for Shortest Paths

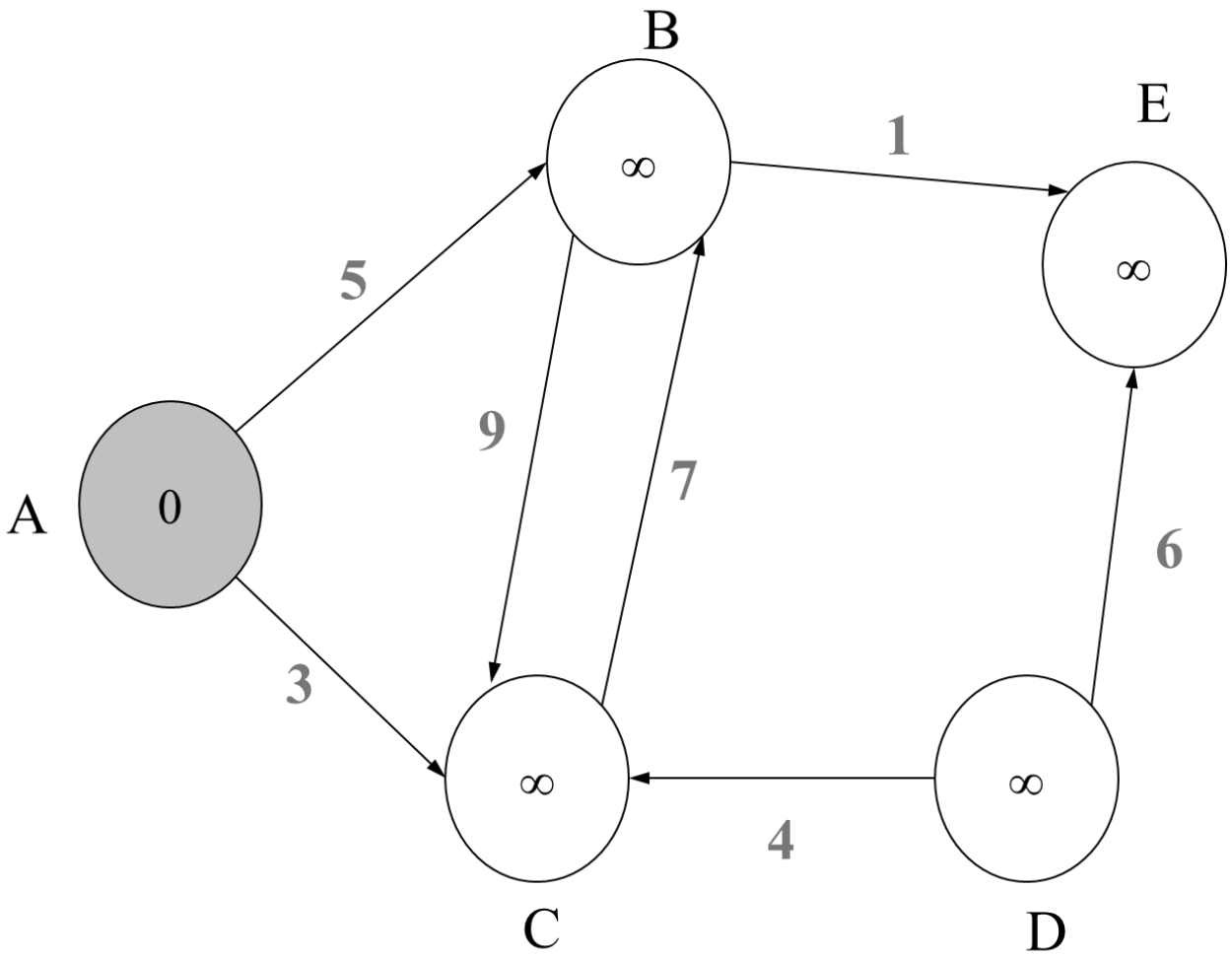


(e)

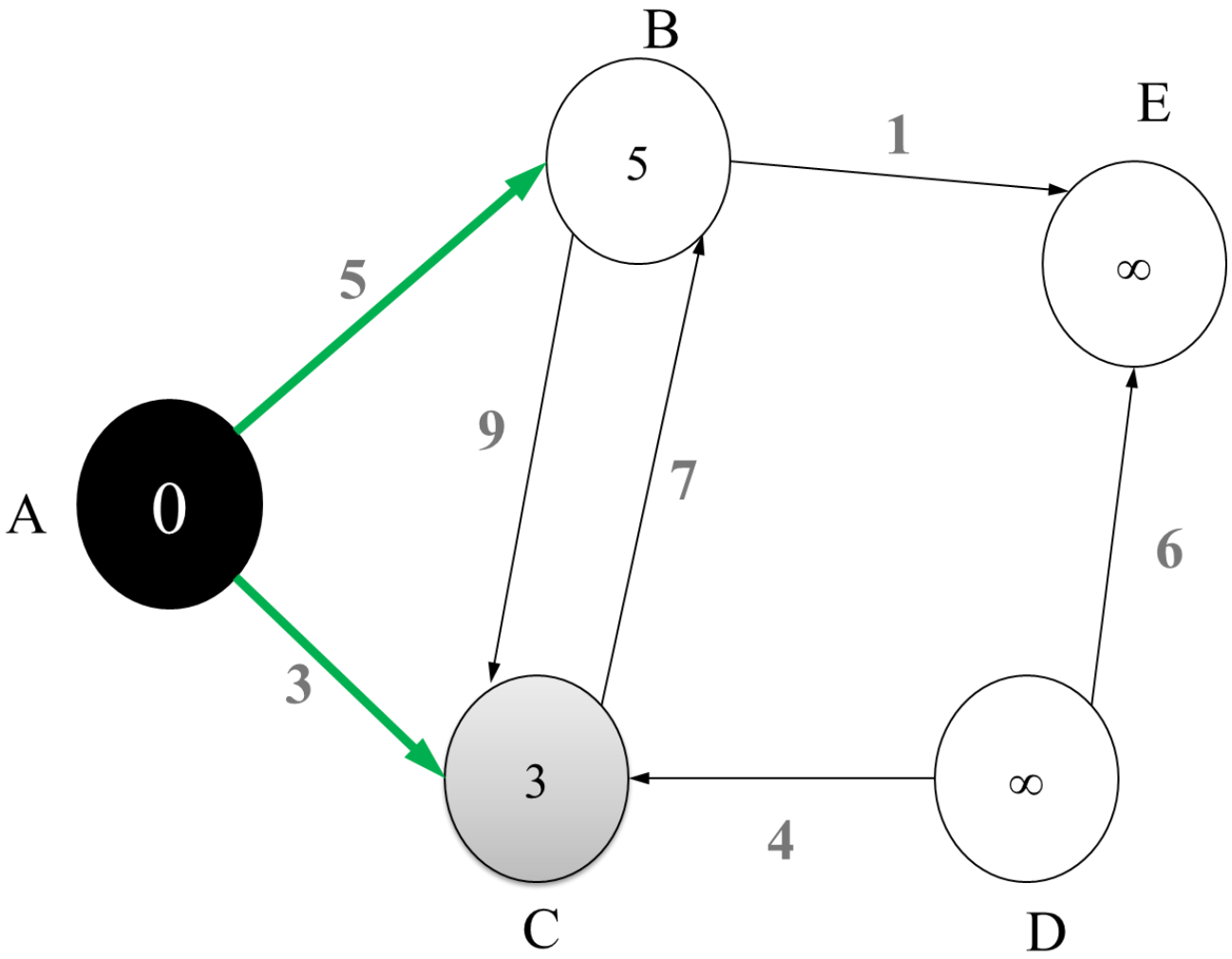


(f)

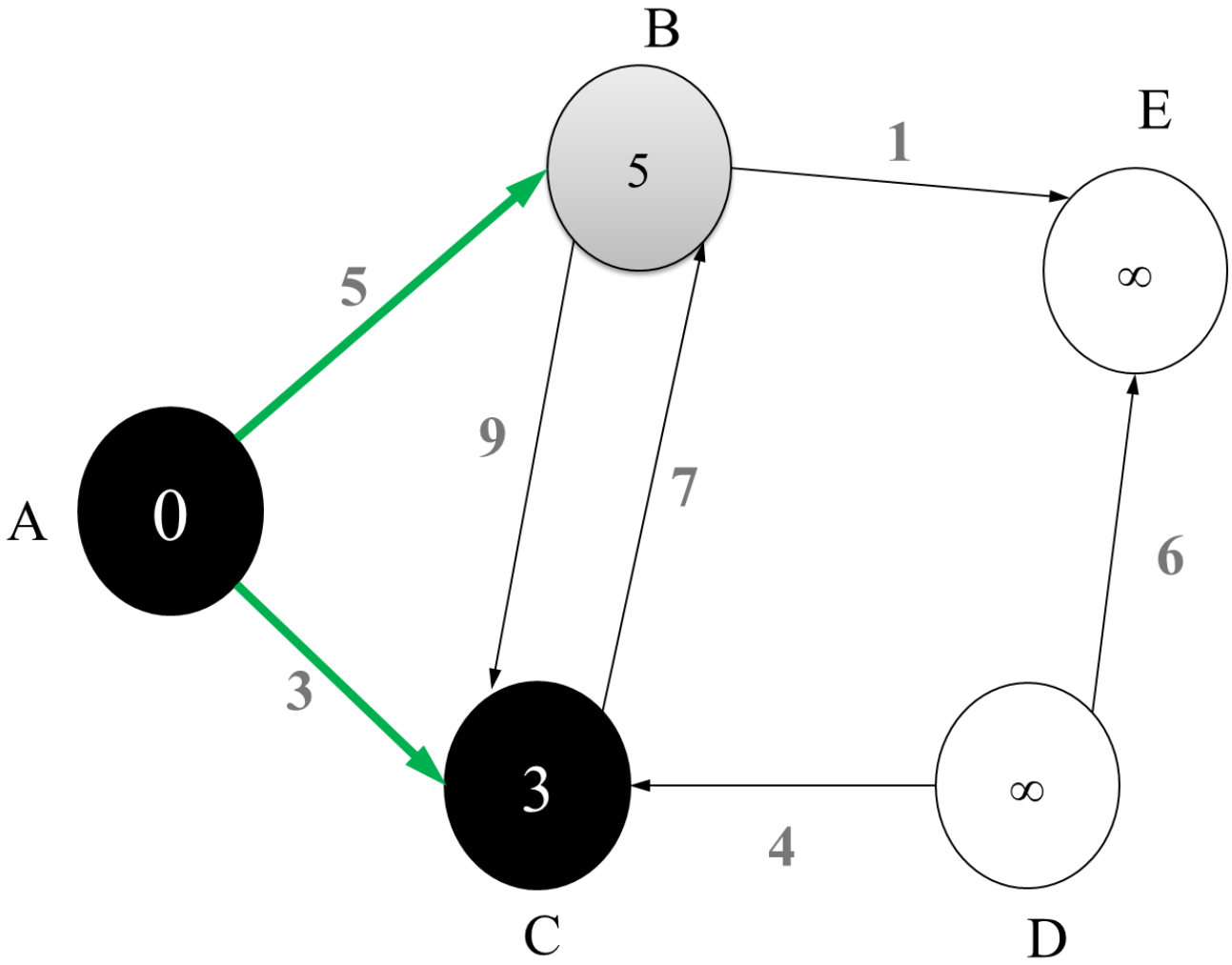
Dijkstra's Algorithm for Shortest Paths



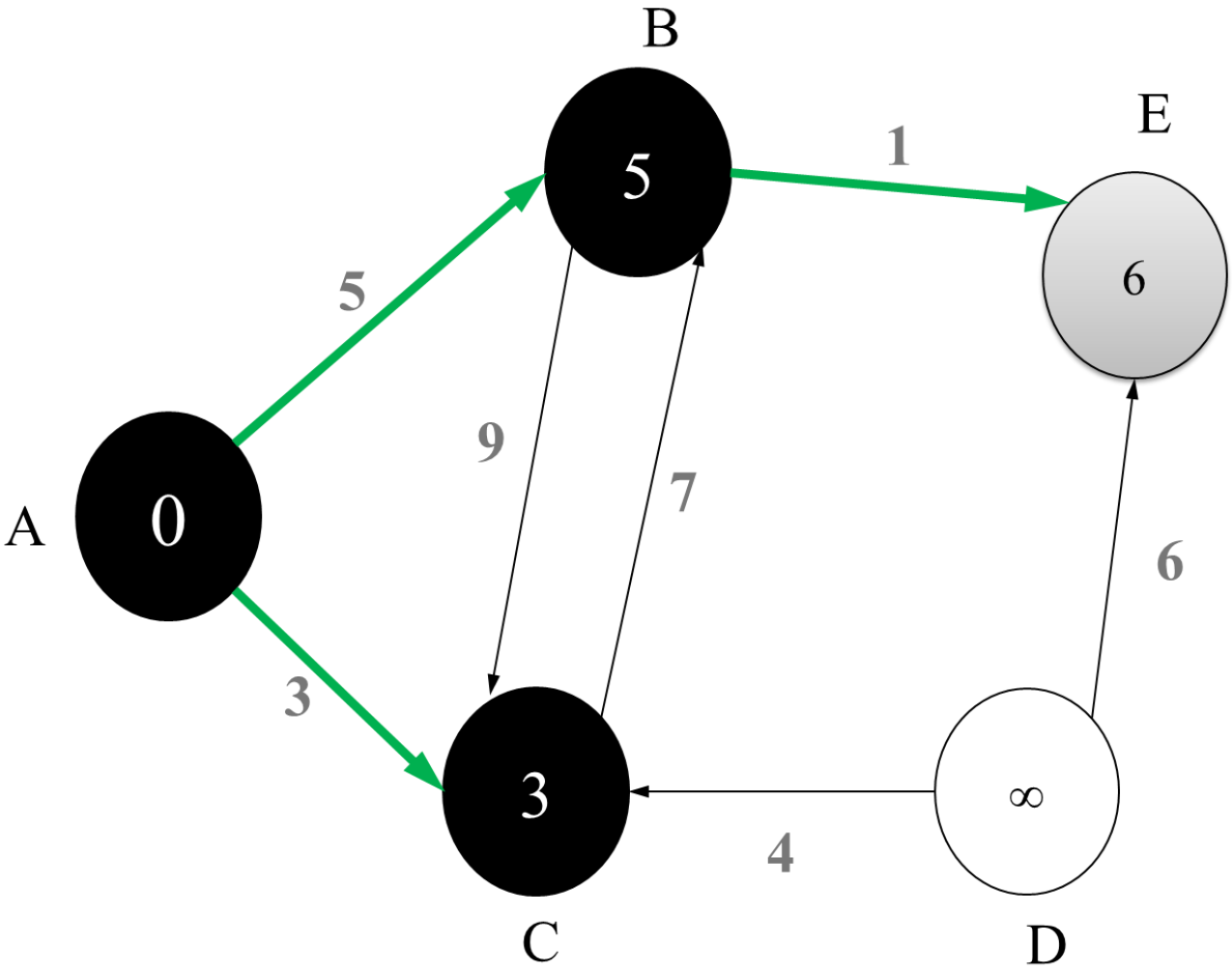
Dijkstra's Algorithm for Shortest Paths



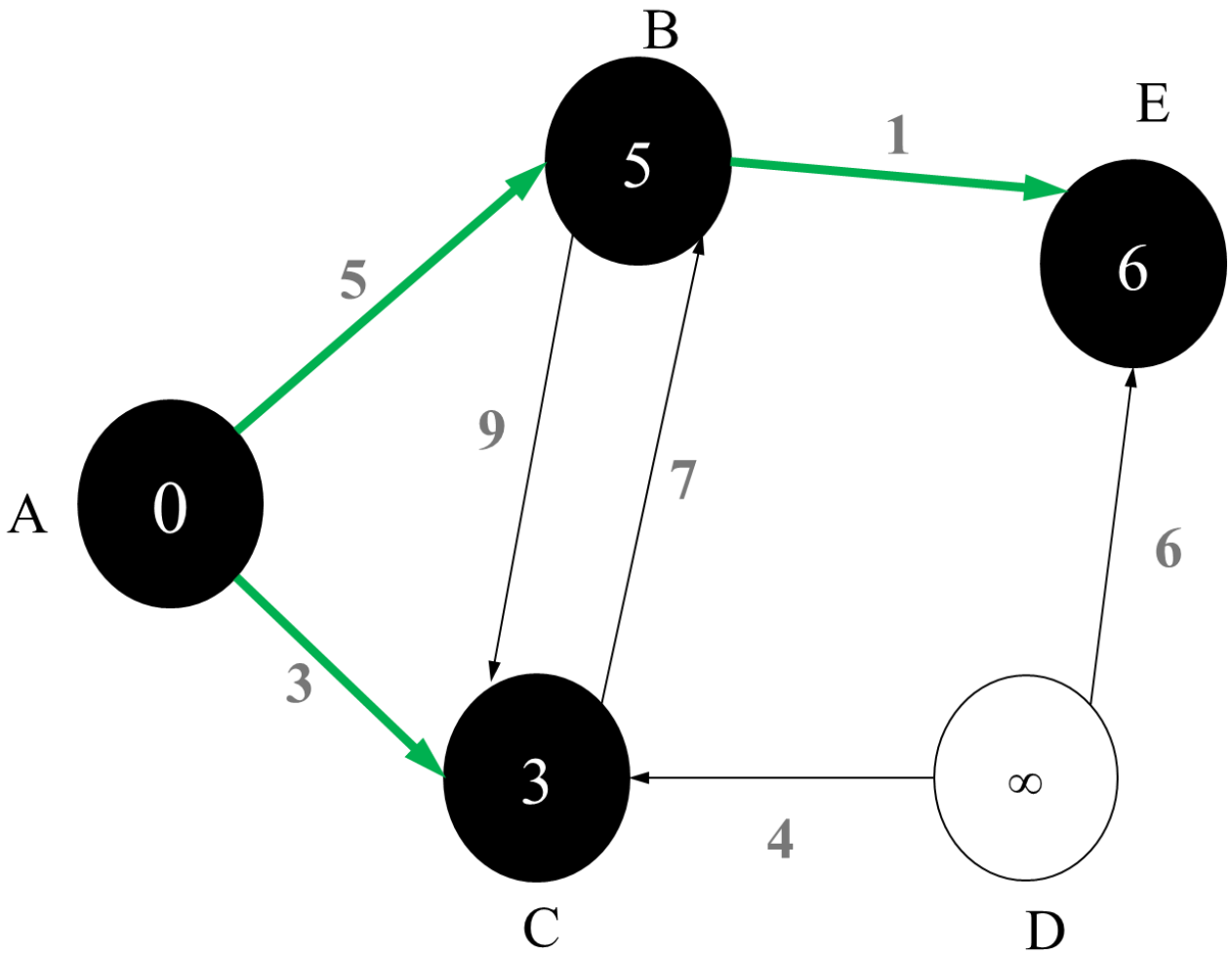
Dijkstra's Algorithm For Shortest Paths



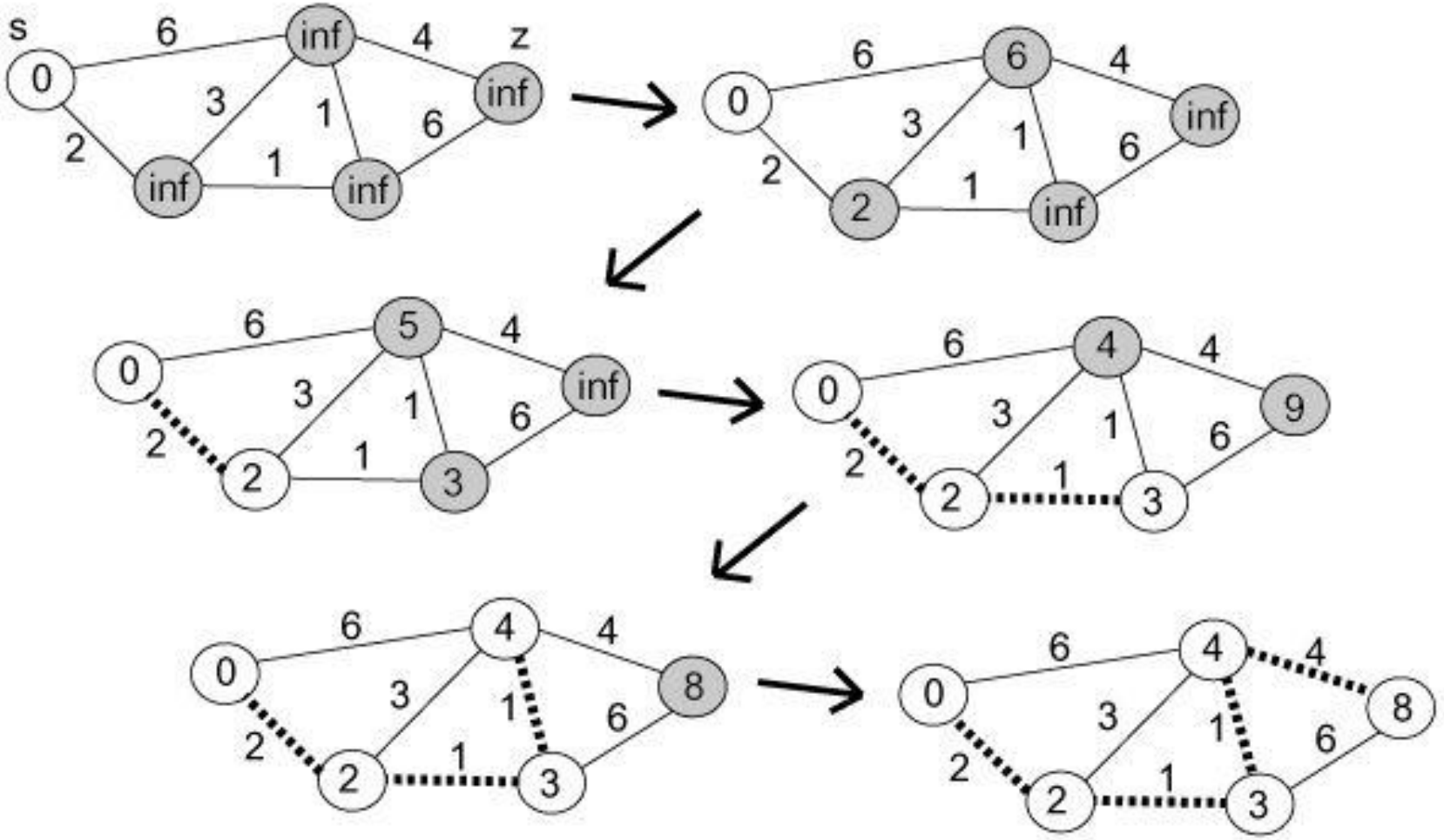
Dijkstra's Algorithm for Shortest Paths



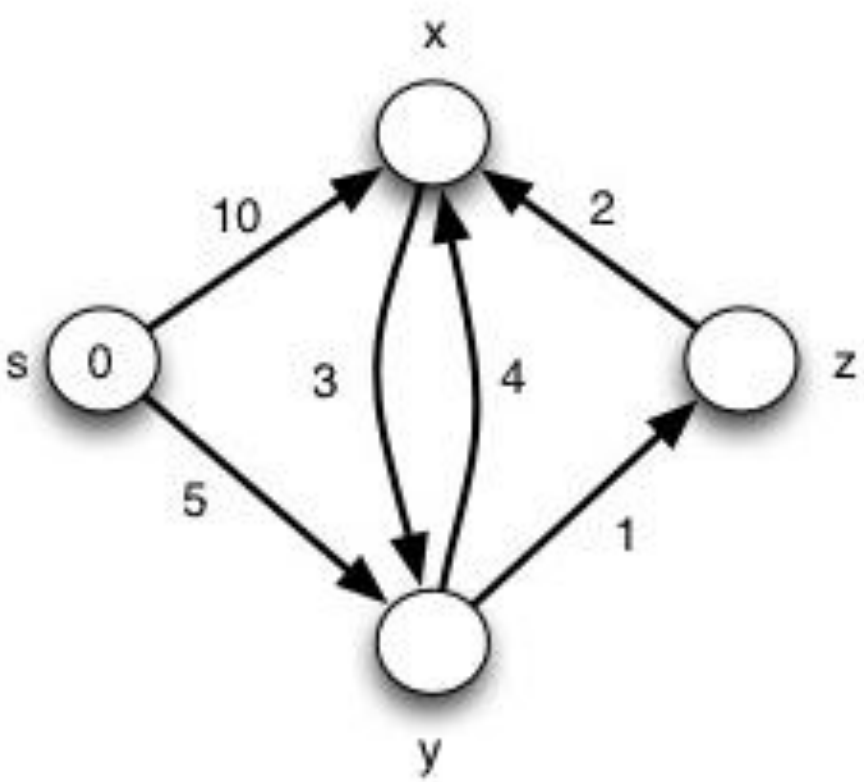
Dijkstra's Algorithm for Shortest Paths

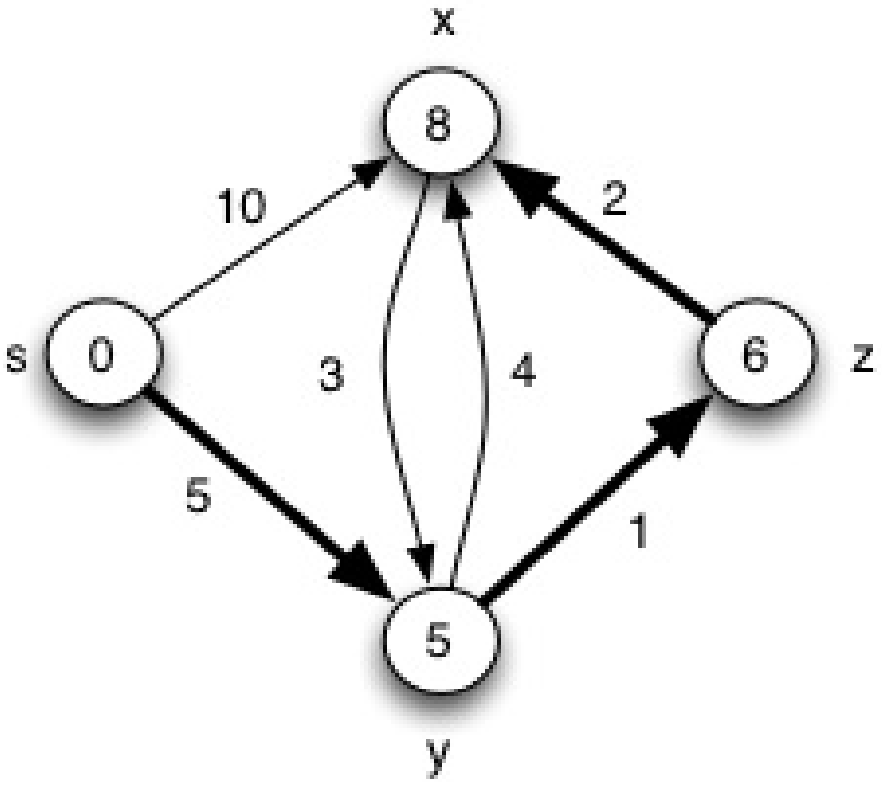


Dijkstra's Algorithm for Shortest Paths

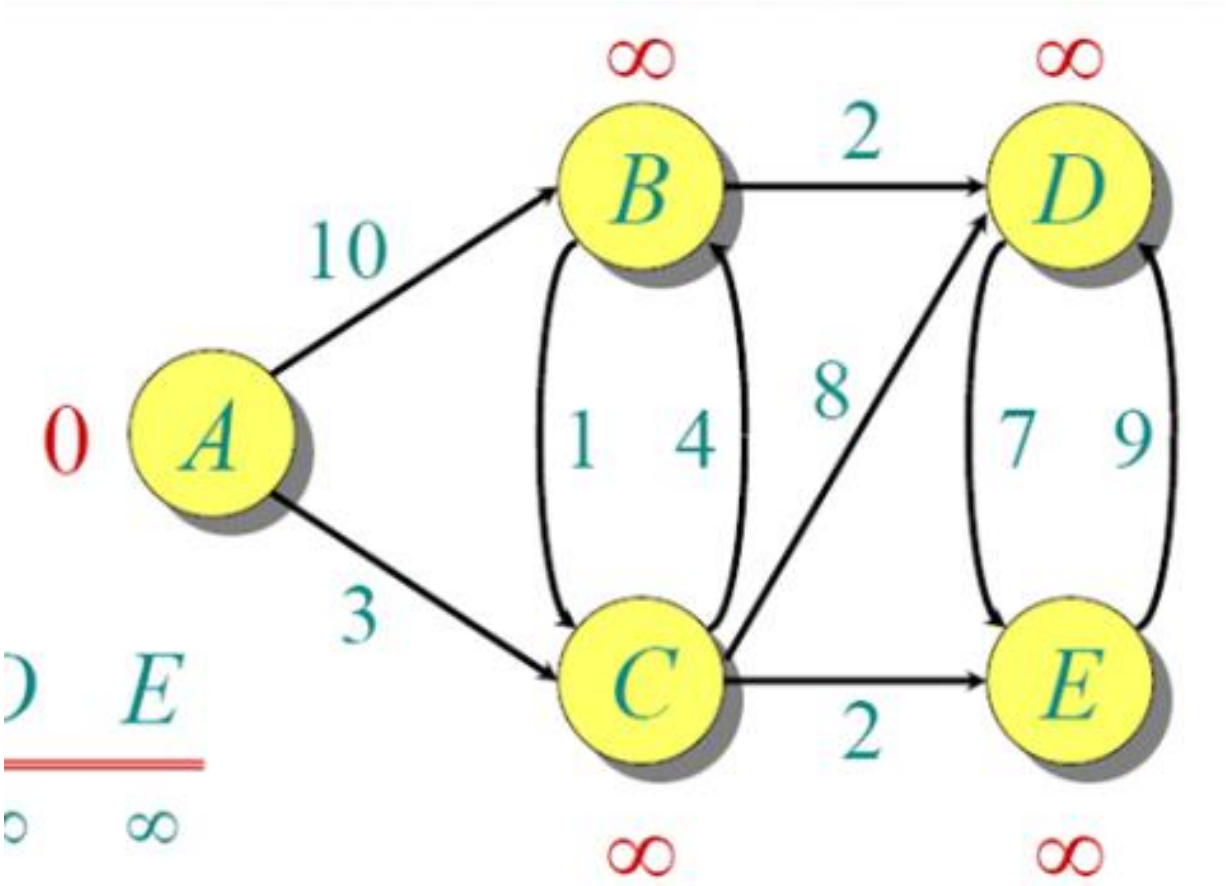


Dijkstra's Algorithm for Shortest Paths

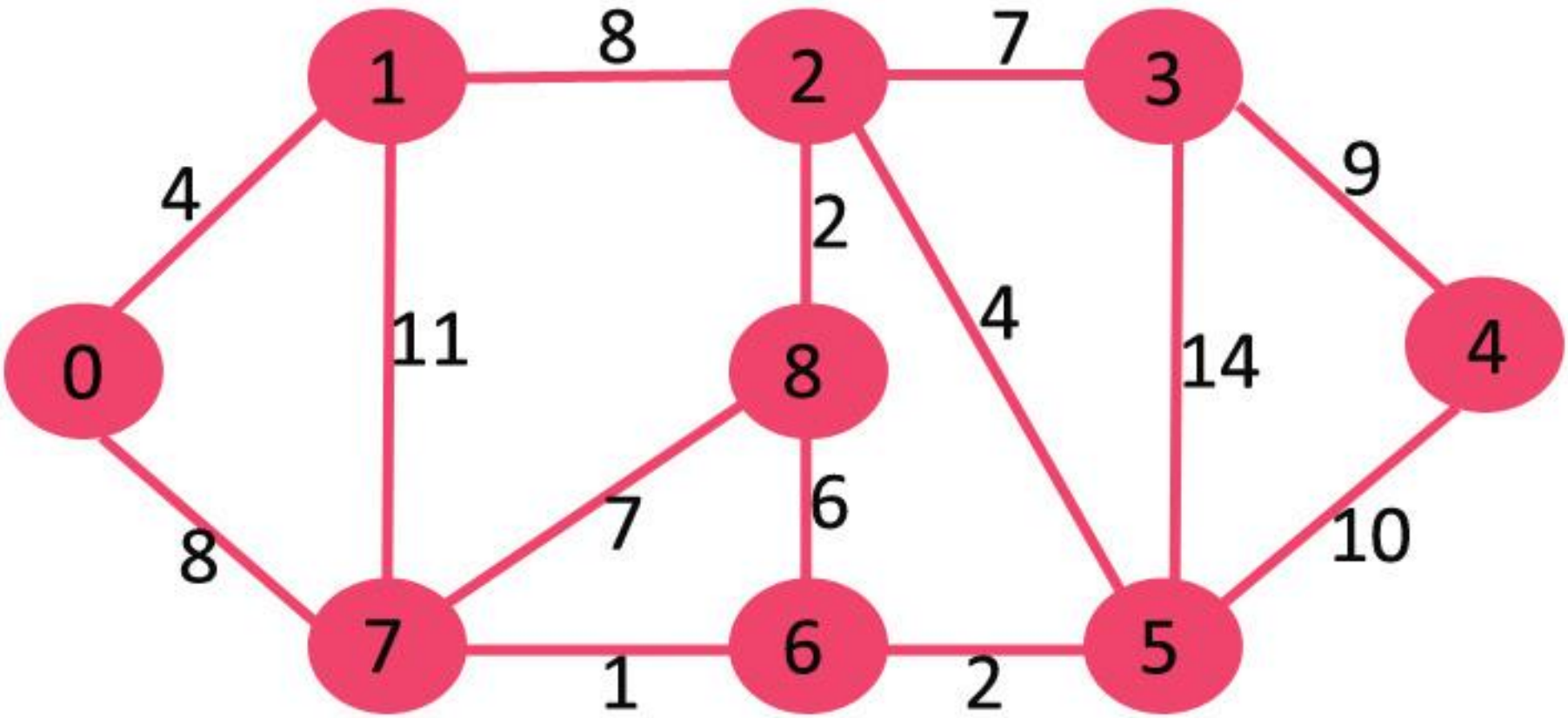




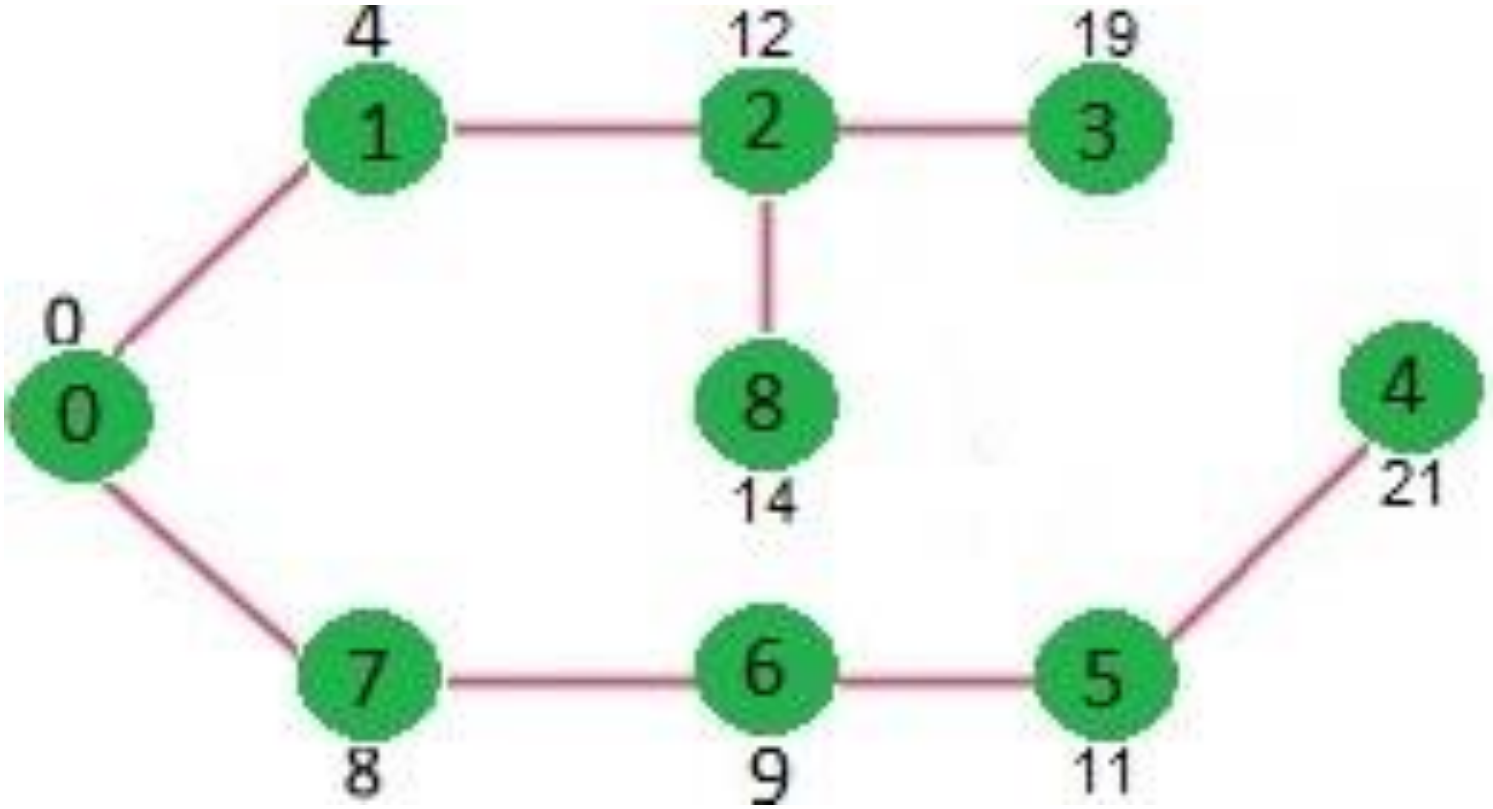
Dijkstra's Algorithm for Shortest Paths



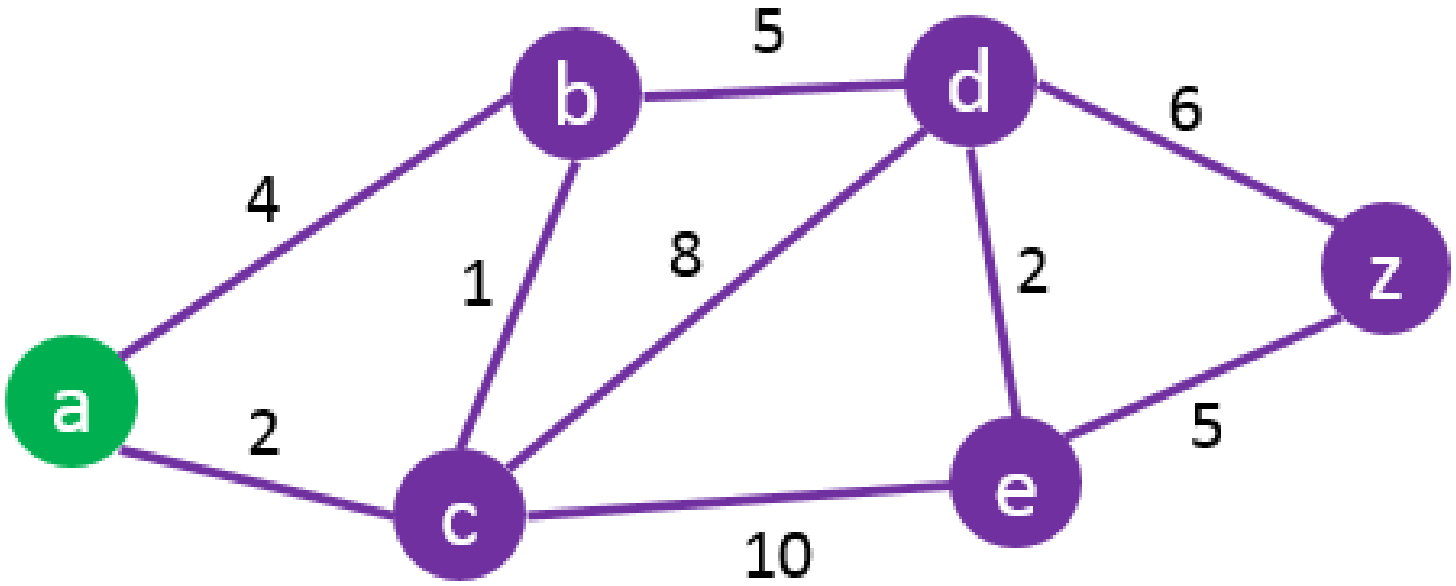
Dijkstra's Algorithm for Shortest Paths



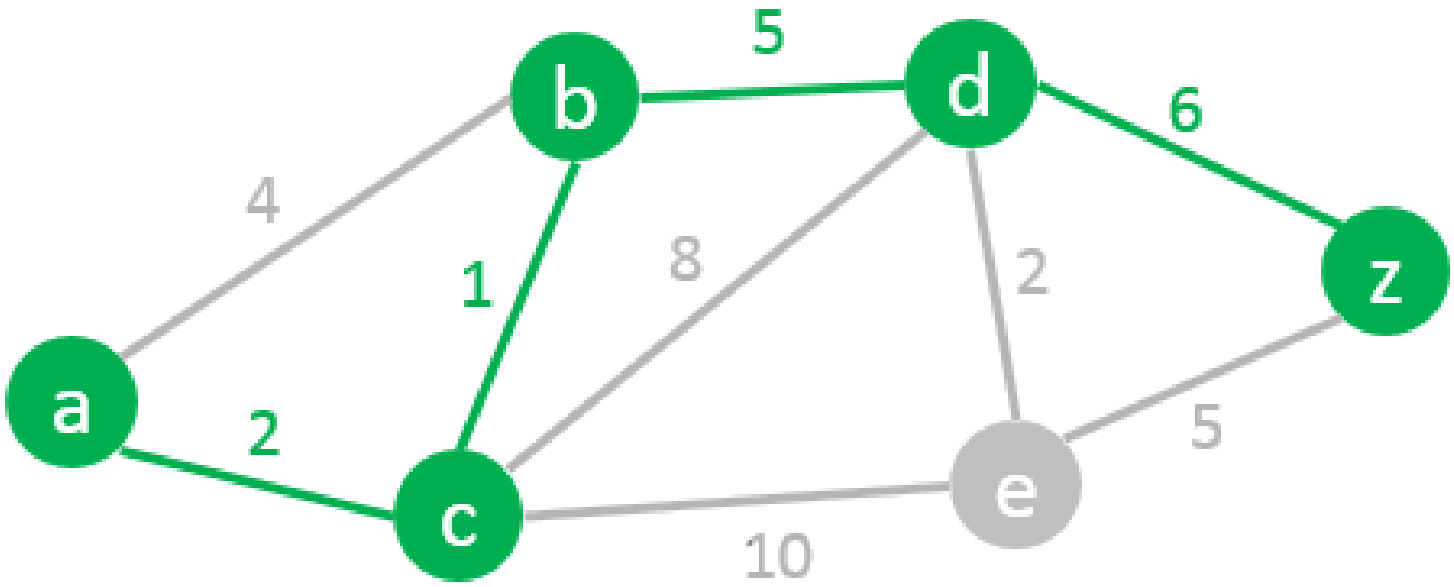
Dijkstra's Algorithm for Shortest Paths



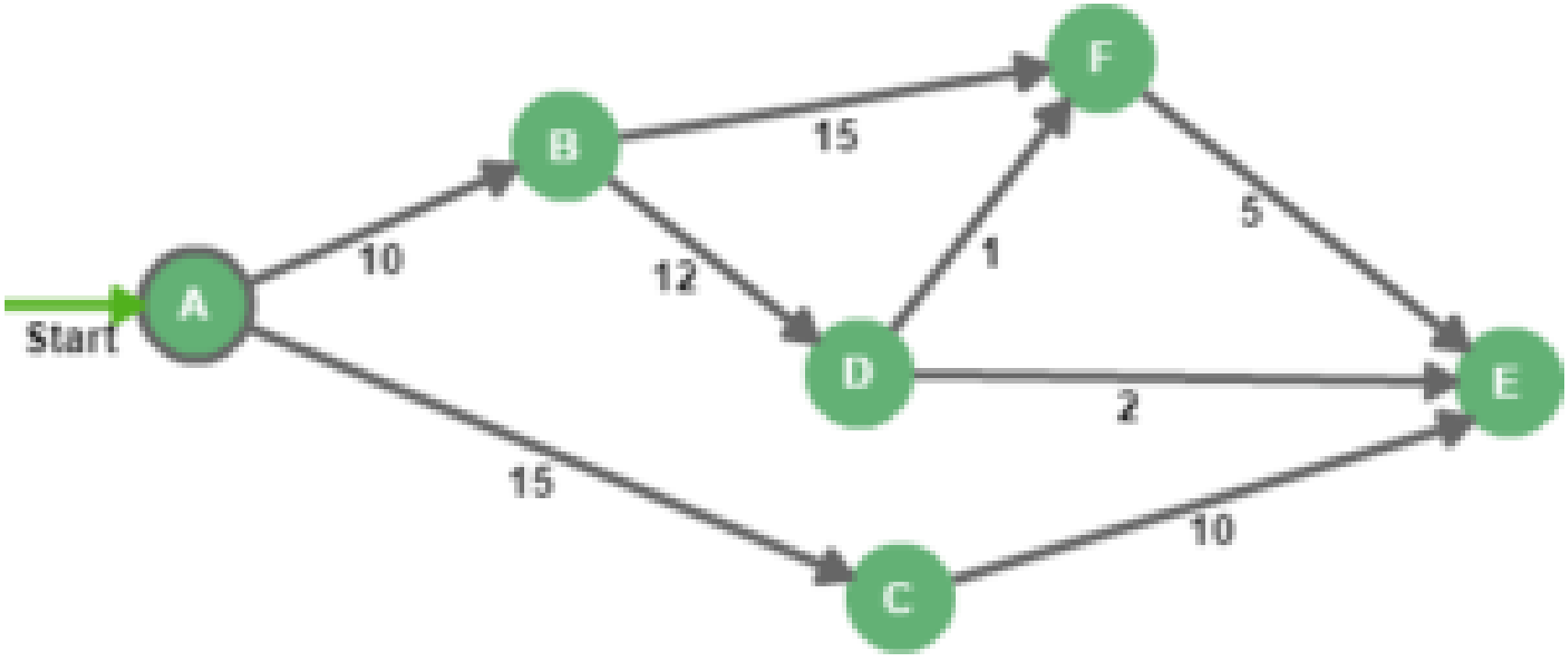
Dijkstra's Algorithm for Shortest Paths



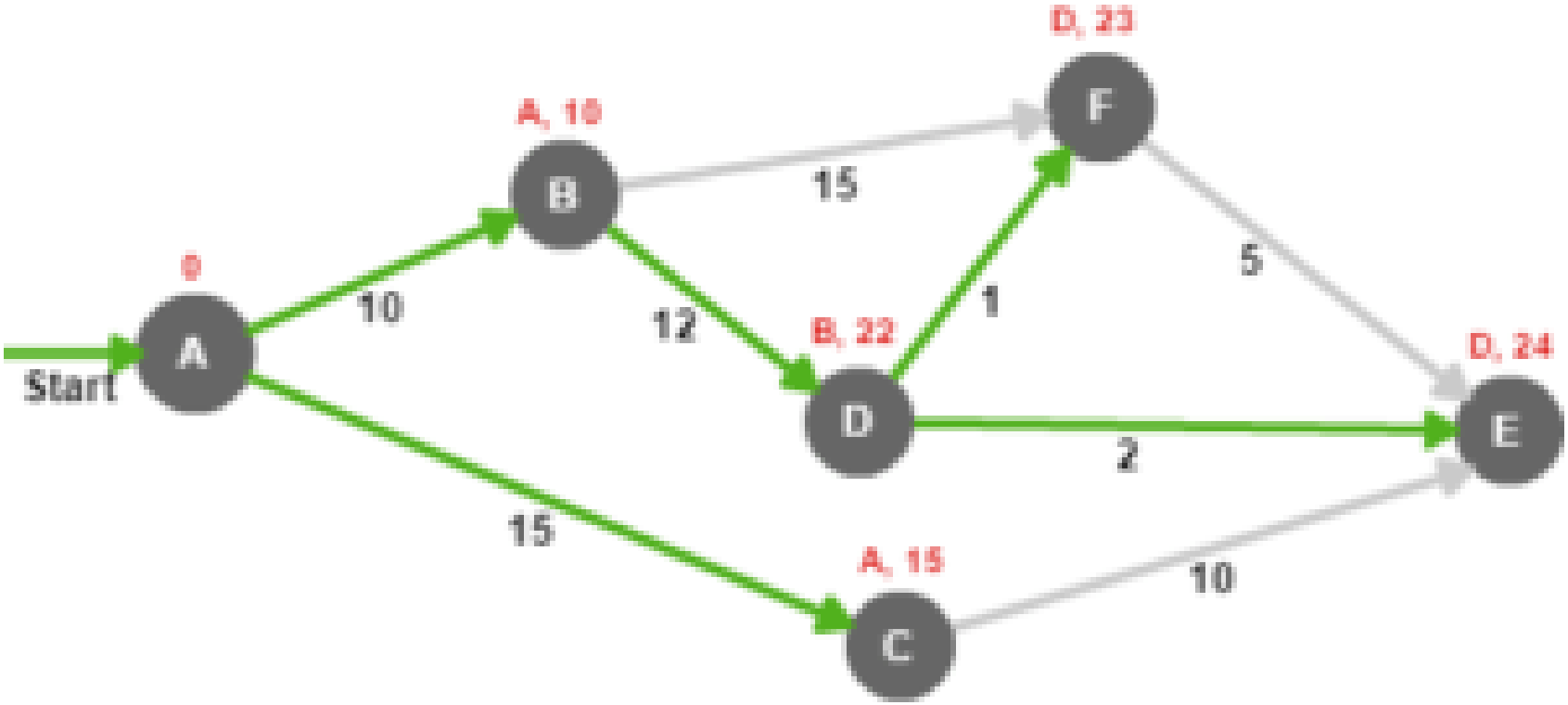
Dijkstra's Algorithm for Shortest Paths



Dijkstra's Algorithm for Shortest Paths



Dijkstra's Algorithm for Shortest Paths




Thank You!