Social Networking

الشبكات الاجتماعية

ITMC 413

إعداد

أ.منار سامي عريف

Graph Metrics : Path length

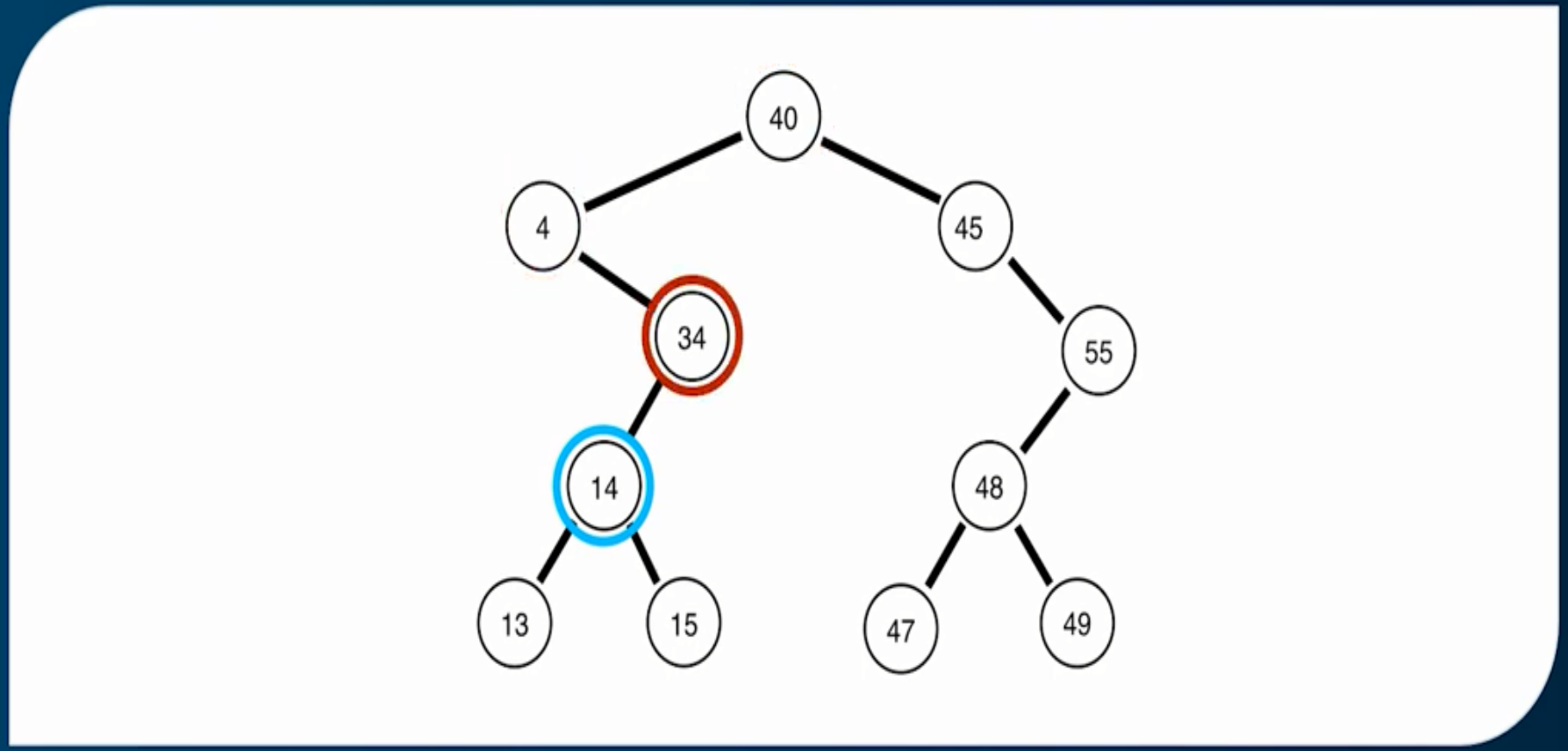
The characteristic path length of a graph is the median of the means of the shortest path lengths connecting each vertex of a graph to all other vertices.

In social networks, networks with low characteristic path lengths can have information spread much more rapidly to the entire population within the network. On the other hand, high characteristic path lengths can make information take longer to spread.

Distance and Breadth-First Search

In addition to simply asking whether two nodes are connected by a path, it is also interesting in most settings to ask how *long* such a path is. In transportation, Internet communication, or the spread of news, it is often important whether something flowing through a network has to travel just a few hops or many.

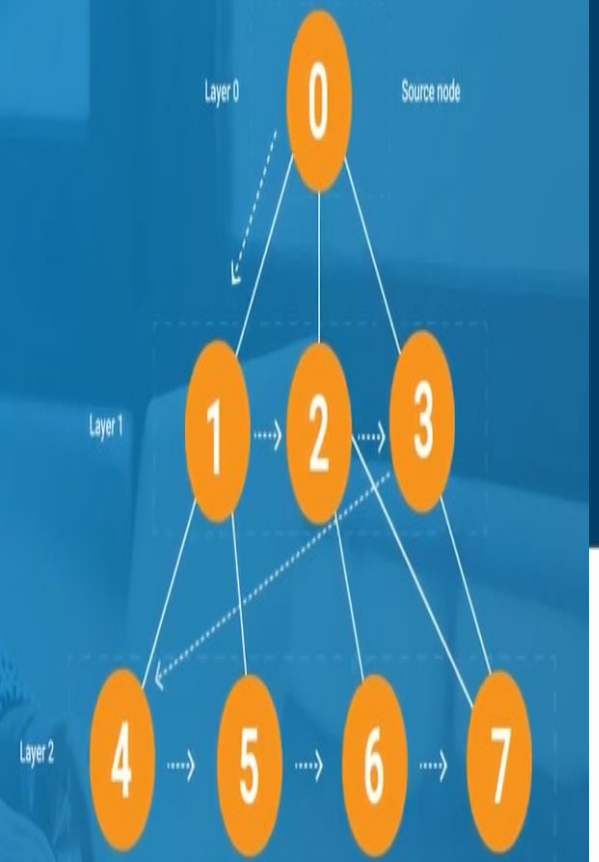
To be able to talk about this notion precisely, we define the *length* of a path to be the number of steps it contains from beginning to end. Using the notion of a path's length, we can talk about whether two nodes are close together or far apart in a graph. In particular, the *distance* between two nodes in a graph is defined as the length of the shortest path between them.

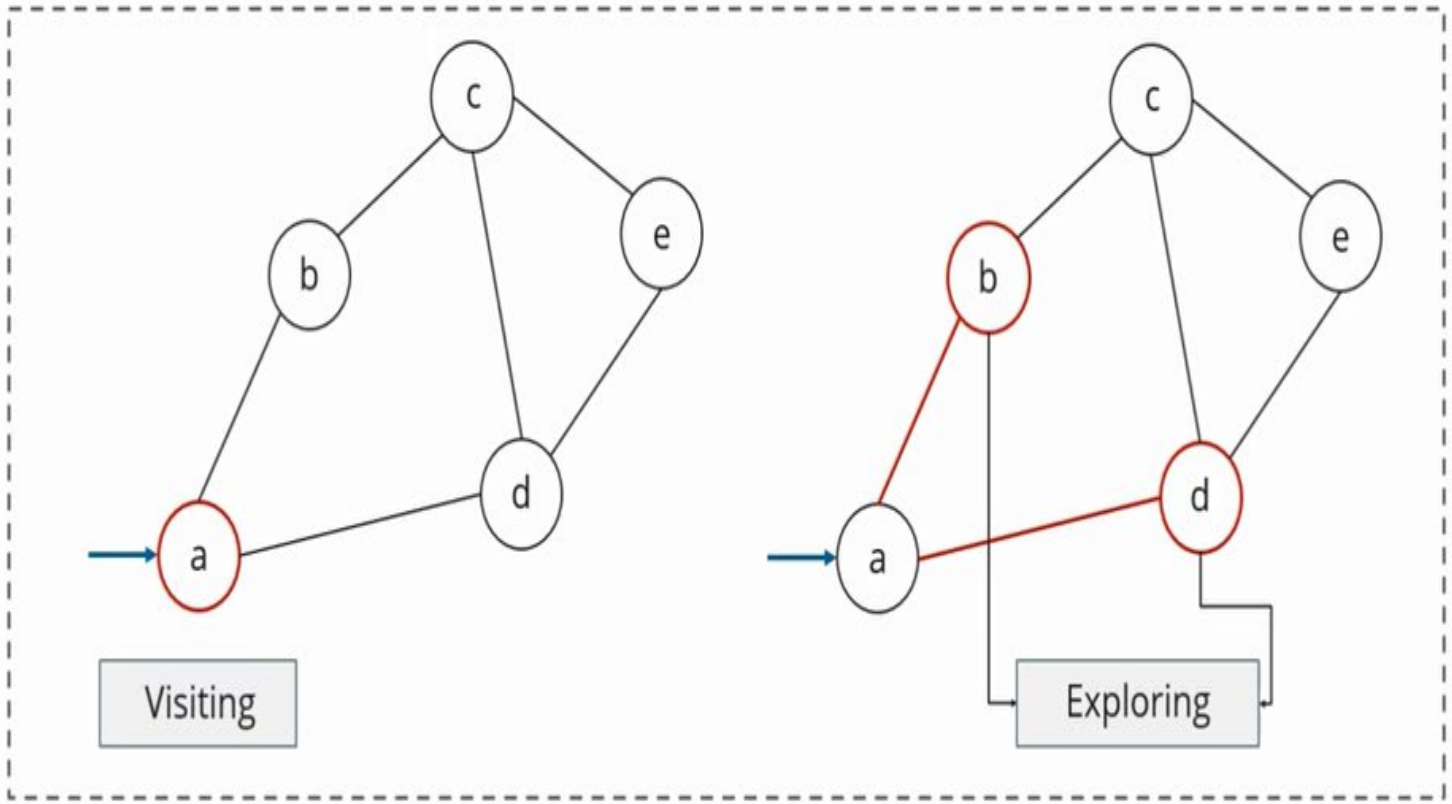


Introduction To Graph Traversal

The process of visiting and exploring a graph for processing is called graph traversal. To be more specific it is all about visiting and exploring each vertex and edge in a graph such that all the vertices are explored exactly once.

Breadth First Search Algorithm

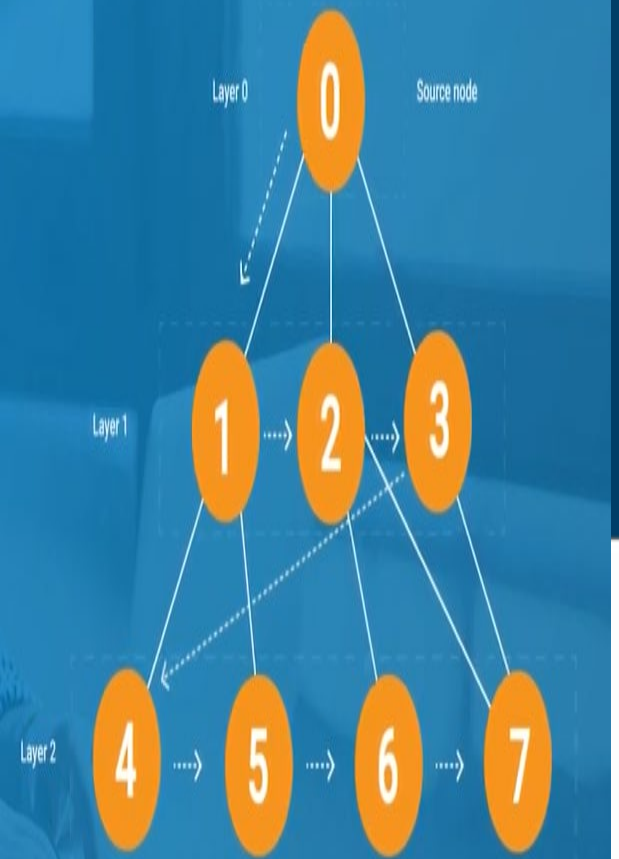


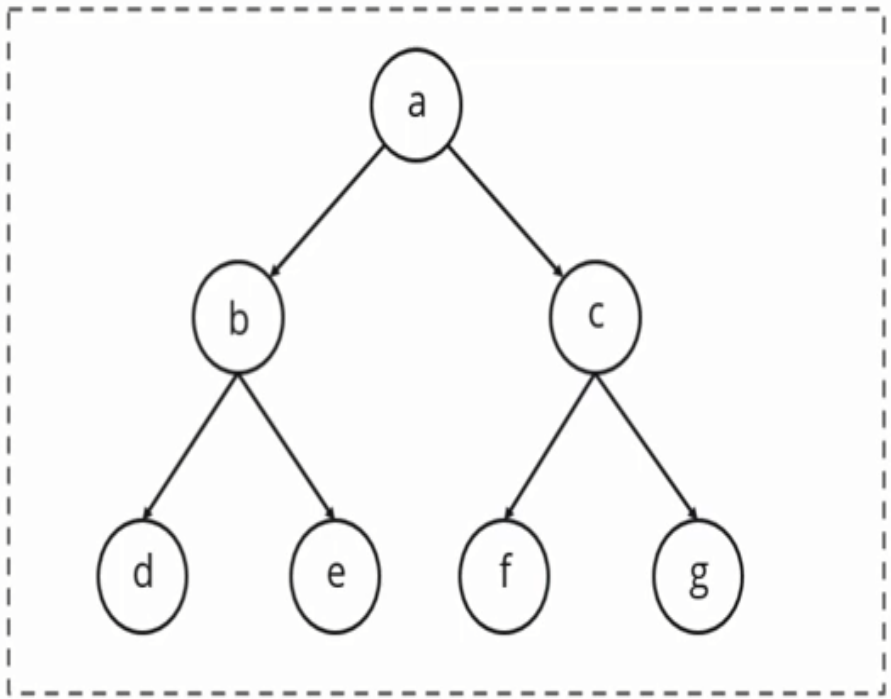


What is the Breadth-First Search Algorithm?

Breadth-First Search algorithm is a graph traversing technique, where you select a random initial node (source or root node) and start traversing the graph layer-wise in such a way that all the nodes and their respective children nodes are visited and explored.

Breadth First Search Algorithm



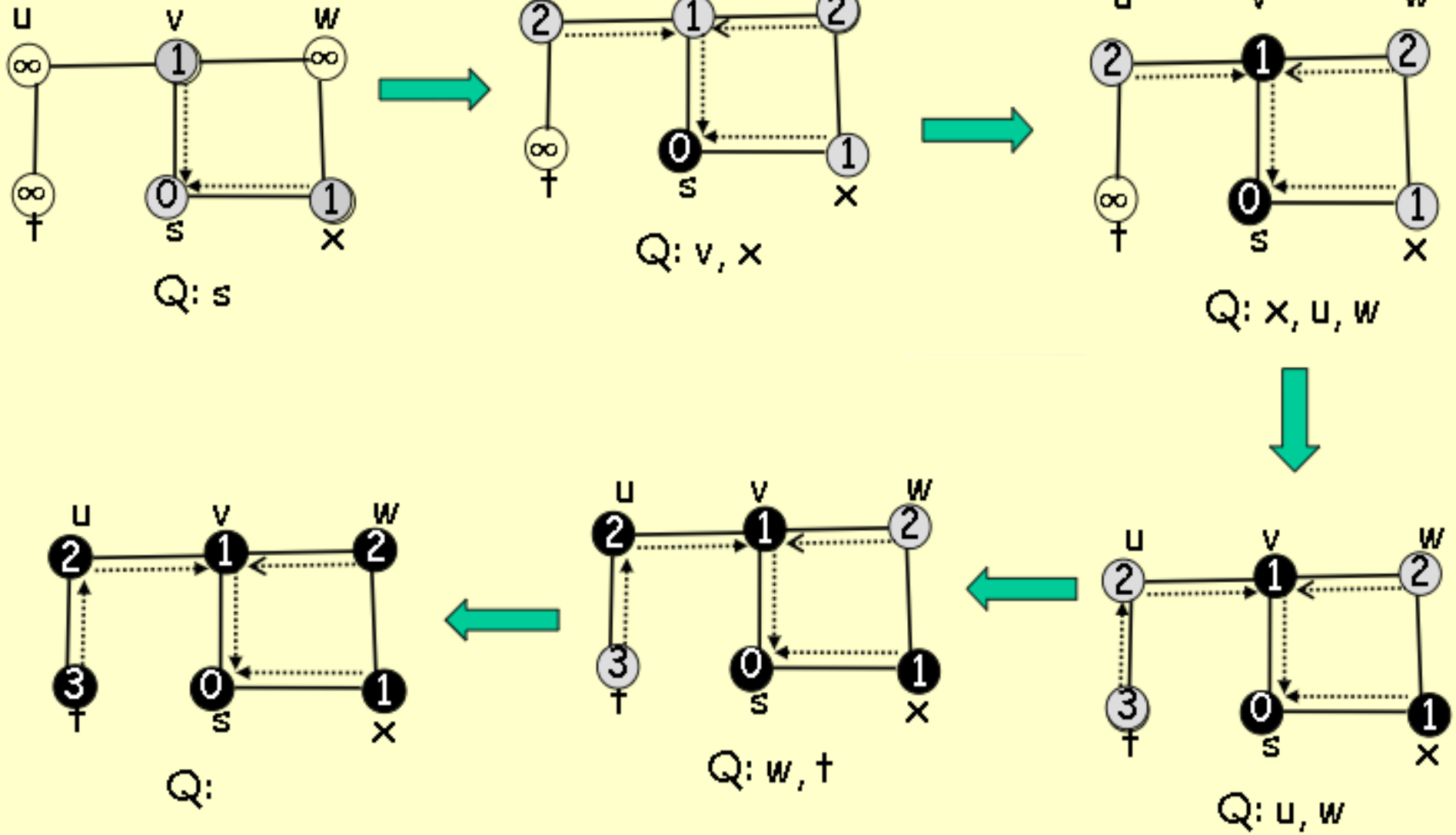


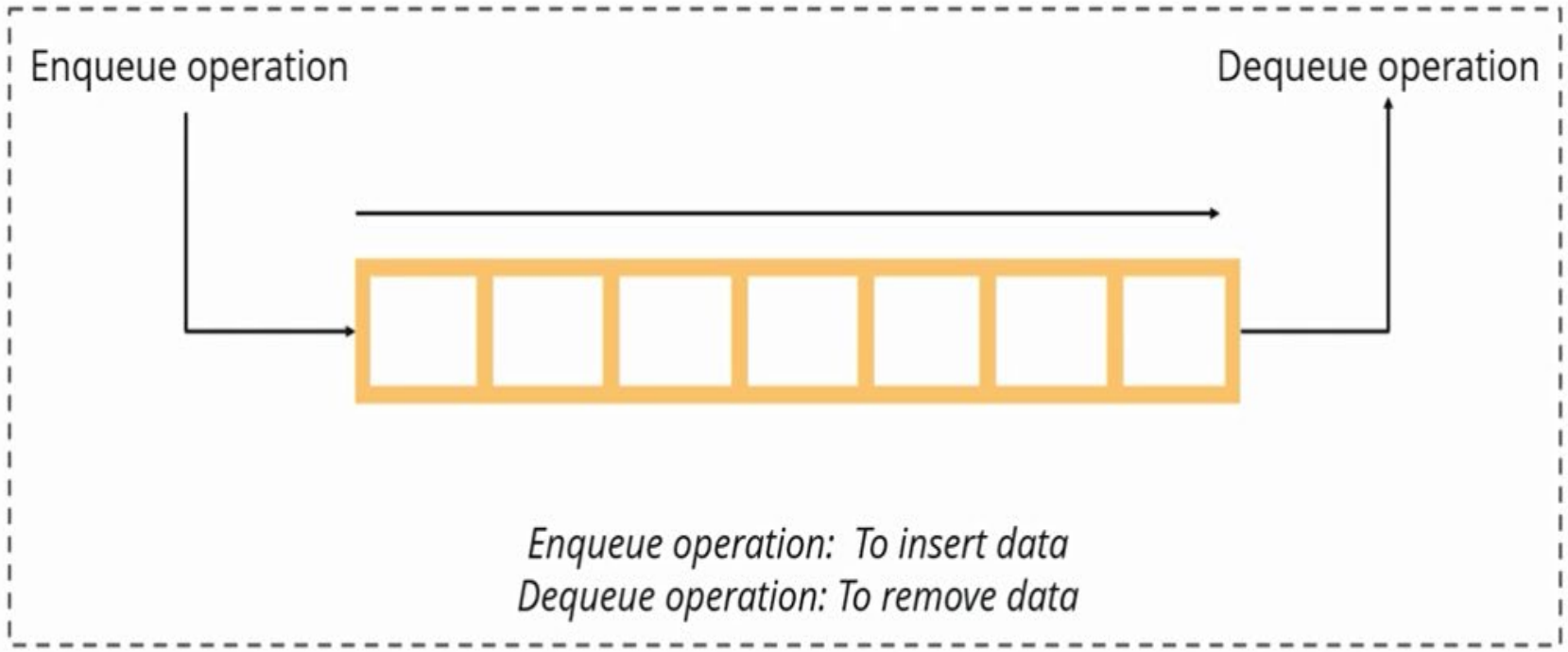
Binary Tree

Understanding the Breadth-First Search Algorithm with an example

Breadth-First Search algorithm follows a simple, level-based approach to solve a problem. Consider the above binary tree (which is a graph). Our aim is to traverse the graph by using the Breadth-First Search Algorithm.

BREADTH-FIRST SEARCH EXAMPLE



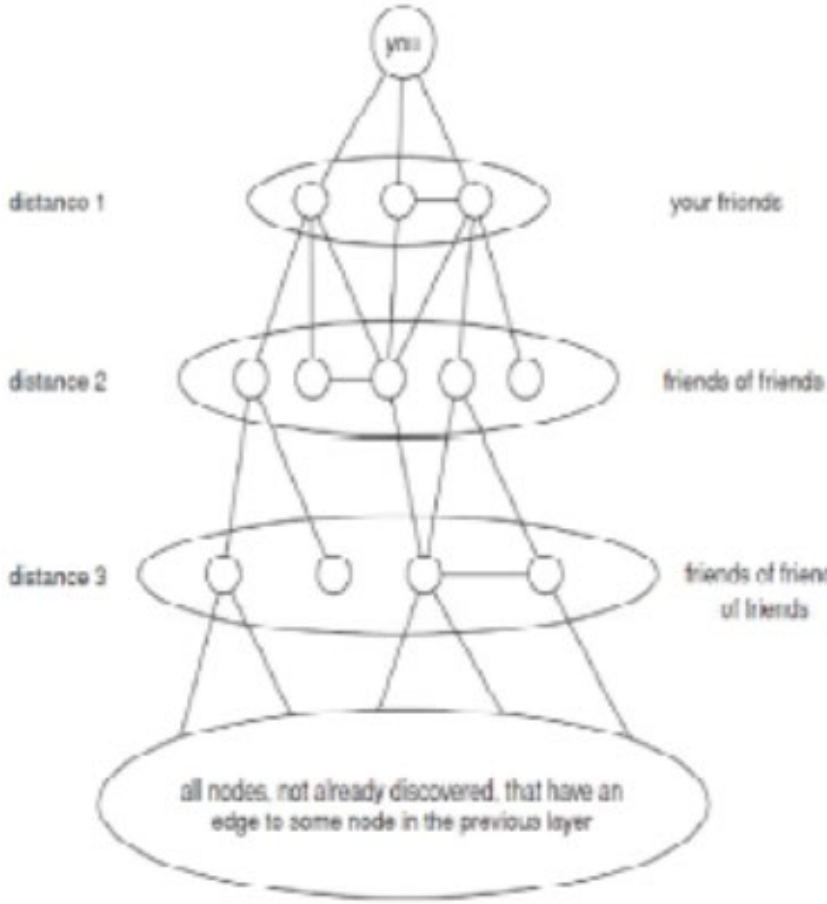


What is a Queue?

A queue is an abstract data structure that follows the First-In-First-Out methodology (data inserted first will be accessed first). It is open on both ends, where one end is always used to insert data and the other is used to remove data.

BREADTH-FIRST SEARCH

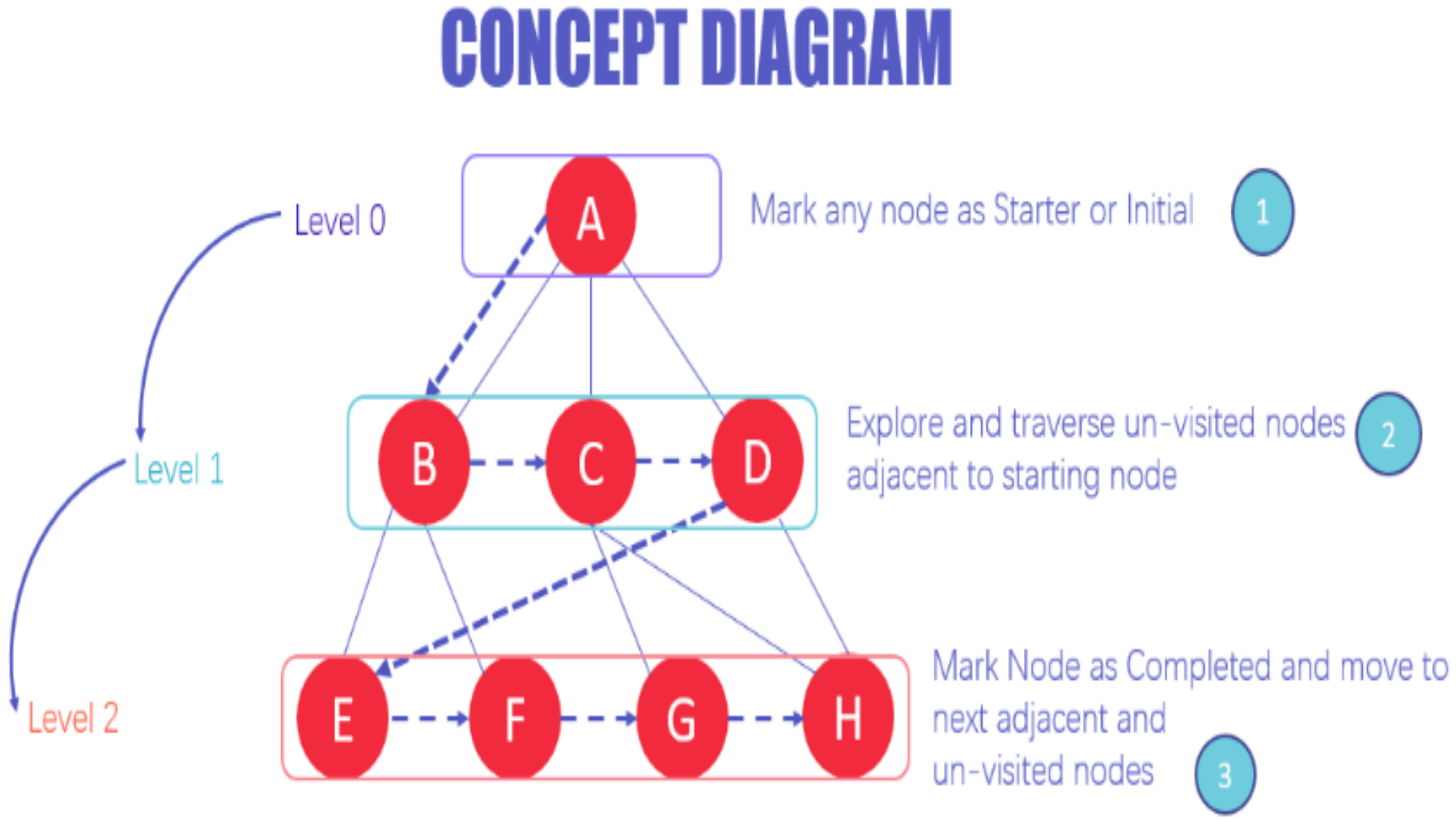
- **Breadth-first search** (**BFS**) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph and explores the neighbor nodes first, before moving to the next level neighbors.
- Breadth-first search discovers the distances to nodes one "layer" at a time; each layer is built of nodes that have an edge to at least one node in the previous layer.
- This technique is called *breadth-first search*, since it searches the graph outward from a starting node, reaching the closest nodes first.



BREADTH-FIRST SEARCH

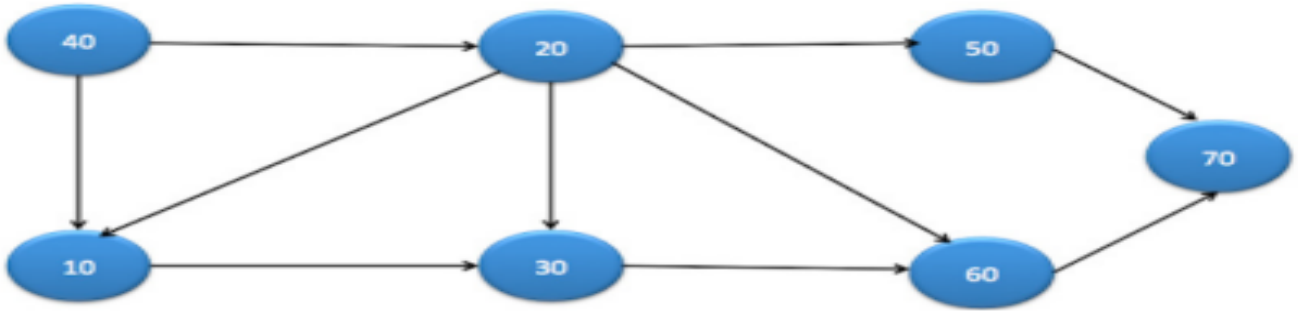
- Breadth first search is graph traversal algorithm. In this algorithm, lets say we start with node i, then we will visit neighbours of i, then neighbours of neighbours of i and so on.
- We use queue to traverse graph. We put first node in queue . It repeatedly extracts node and put its neighbours in the queue. We need to track if node have been visited before or not. It can be easily done with help of boolean variable visited in the node. If node have been already visited then we won't visit it again.

The architecture of BFS algorithm



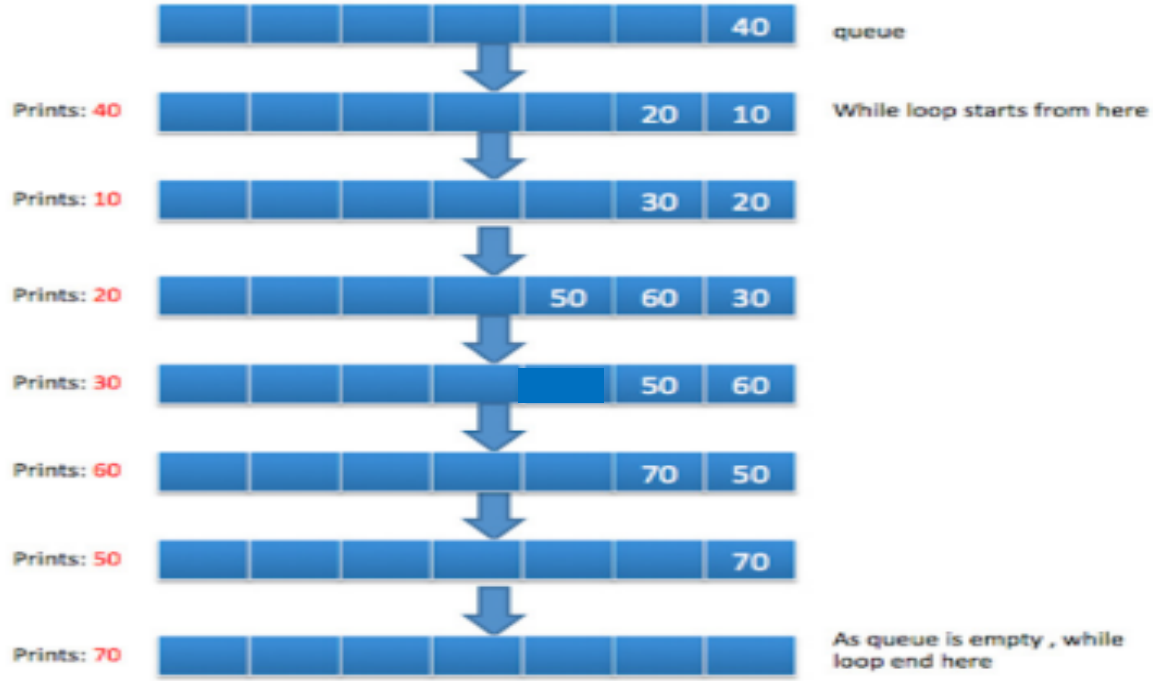
BREADTH-FIRST SEARCH EXAMPLE

BFS example



Steps for BFS:

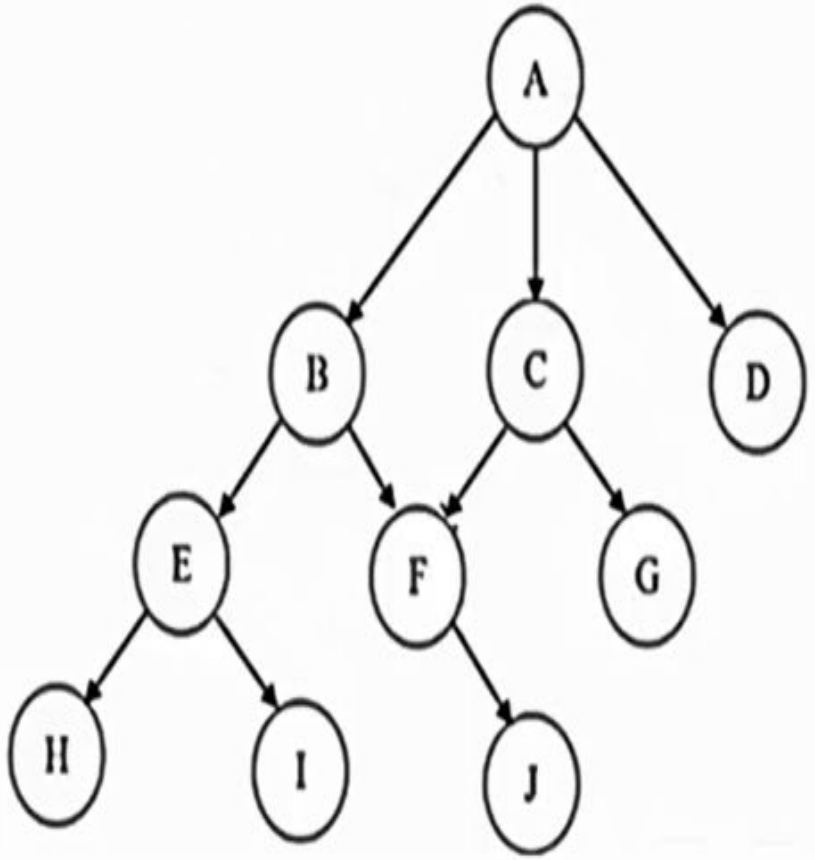
- Create empty queue and push root node to it.
- Do the following when queue is not empty
  - Pop a node from queue and print it.
  - Find neighbours of node with the help of adjacency matrix and check if node is already visited or not.
  - Push neighbours of node into queue if not null



Breadth first traversal is :40,10,20,30,60,50,70

Apply the breadth first search (BFS) algorithm on the following graph, where the start state is (A) and the goal state is (G).

Iteration #	X	open	closed
0	–	[A]	[]
1	A	[BCD]	[A]
2	B	[CDEF]	[BA]
3	C	[DEFG]	[CBA]
4	D	[EFG]	[DCBA]
5	E	[FGHI]	[EDCBA]
6	F	[GHIJ]	[FEDCBA]
7	G	G is the goal	

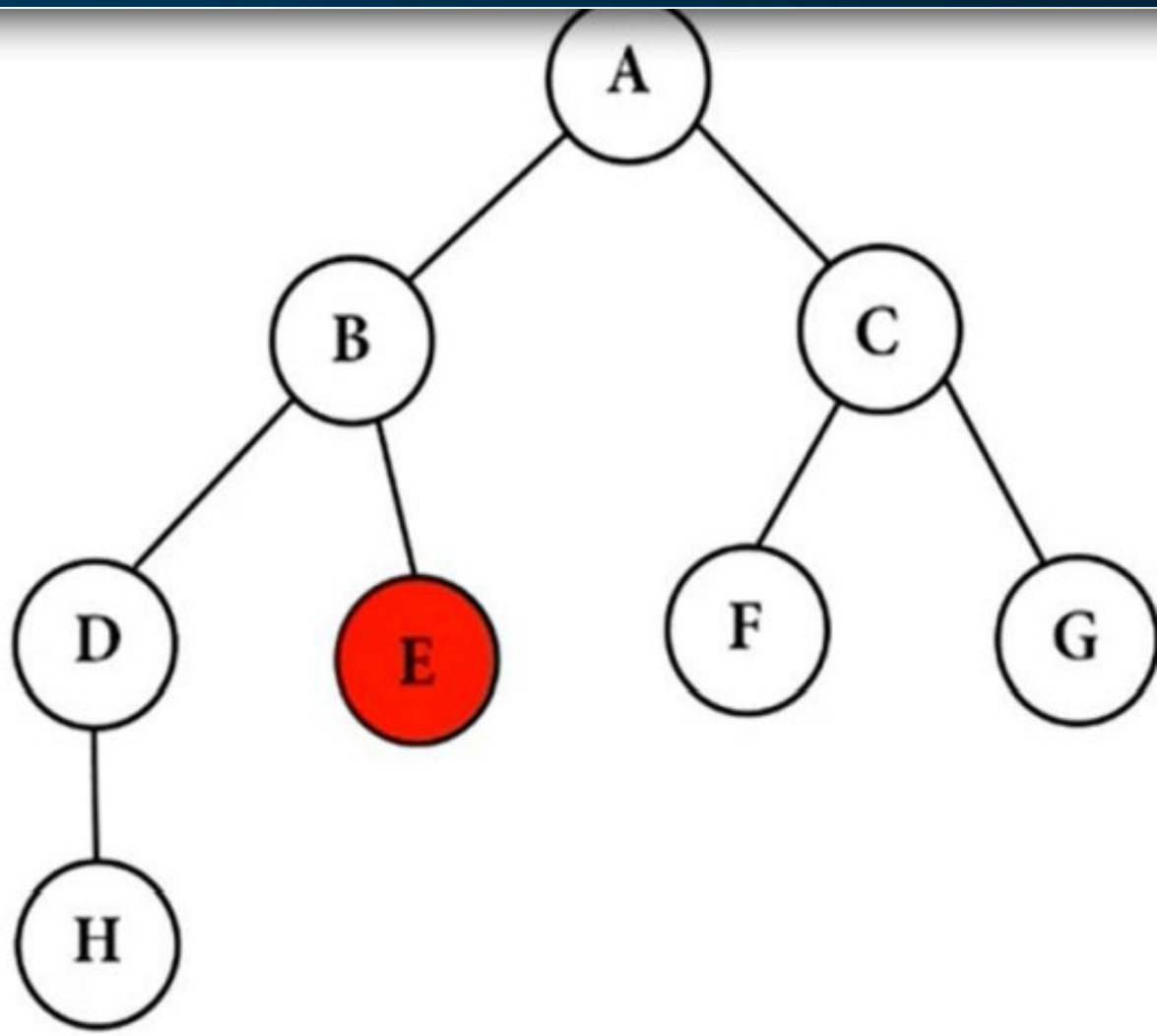


BREADTH-FIRST SEARCH EXAMPLE

Breadth First Search

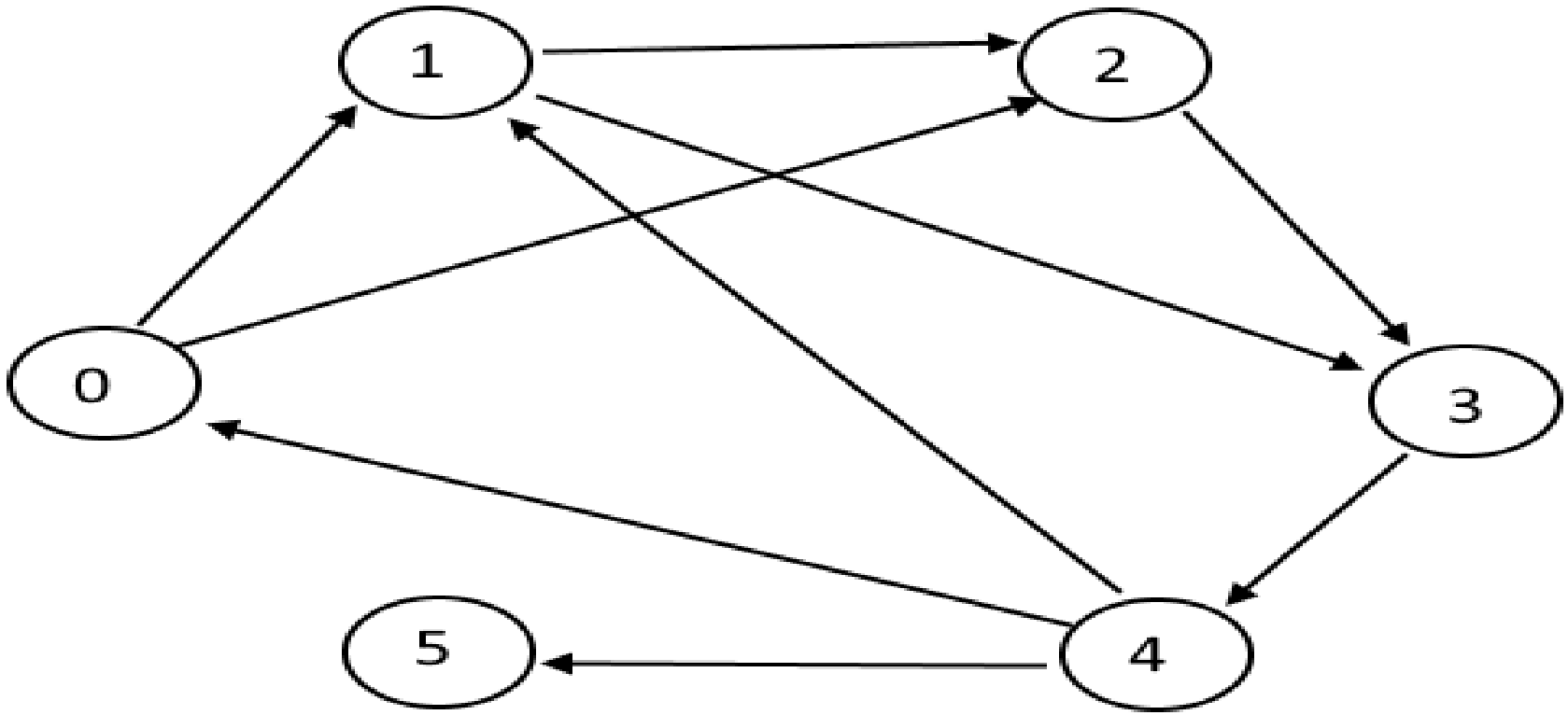
(BFS)

—	A
A	B C
B	C D E
C	D E F G
D	E F G H
E	Goal



Quiz ?

Graph



BFS starting from Node 0

Betweenness centrality (Measure Based on Network Structure)

Betweenness centrality is an indicator of a node's centrality in a network. It is equal to the number of shortest paths from all vertices to all others that pass through that node.

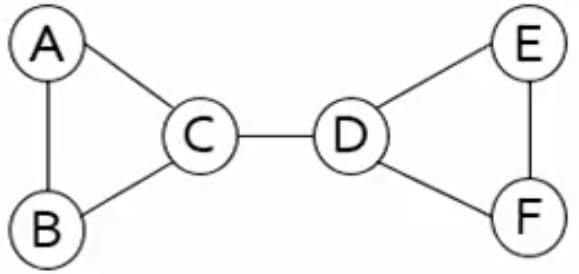
- The vertex betweenness centrality BC(v) of a vertex v is defined as follow

$$BC(v) = \sum_{u,v \in V} \left(\frac{\sigma_{uw}(v)}{\sigma_{uw}} \right)$$

σ_{uw} = Total number of shortest paths between node u and w.

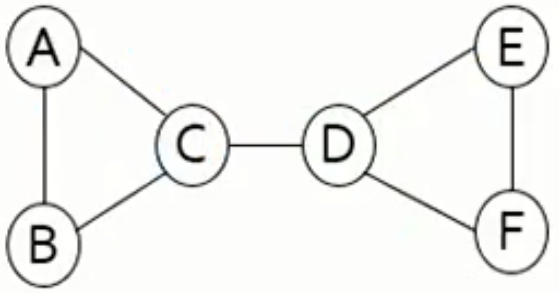
$\sigma_{uw}(v)$ = Total number of shortest paths between node u and w that pass through v.

Calculate the between centrality for node C



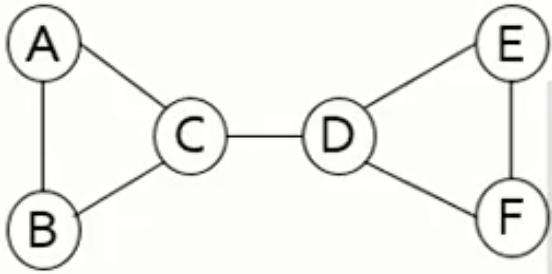
	σ_{uw}	$\sigma_{uw}(v)$	$\sigma_{uw}(v) / \sigma_{uw}$
(A,B)	1	0	0
(A,D)	1	1	1
(A,E)	1	1	1
(A,F)	1	1	1
(B,D)	1	1	1
(B,E)	1	1	1
(B,F)	1	1	1

Calculate the between centrality for node C



	σ_{uw}	$\sigma_{uw}(v)$	$\sigma_{uw}(v) / \sigma_{uw}$
(D,E)	1	0	0
(D,F)	1	0	0
(E,F)	1	0	0

Betweenness Centrality for C =



	σ_{uw}	$\sigma_{uw}(v)$	$\sigma_{uw}(v) / \sigma_{uw}$
(A,B)	1	0	0
(A,D)	1	1	1
(A,E)	1	1	1
(A,F)	1	1	1
(B,D)	1	1	1
(B,E)	1	1	1
(B,F)	1	1	1
(D,E)	1	0	0
(D,F)	1	0	0
(E,F)	1	0	0

Betweenness Centrality for C = 6

Betweenness Centrality for A = ?

BC = 0/1 = 0

BD = 0/1 = 0

BE = 1/1 = 1

BF = 0/1 = 0

CD = 0/1 = 0

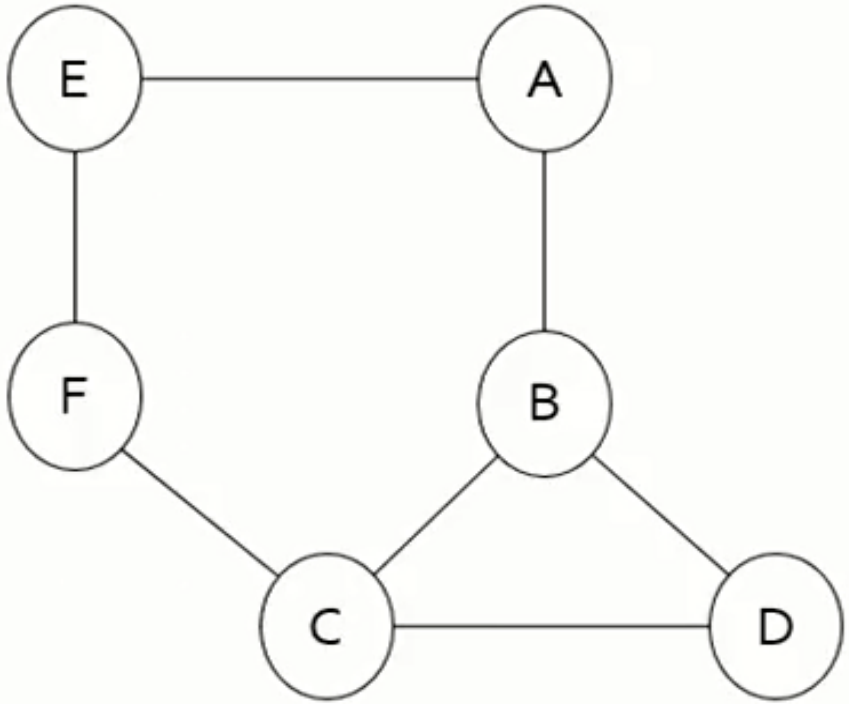
CE = 0/1 = 0

CF = 0/1 = 0

DE = 1/2 = 0.5

DF = 0/1 = 0

EF = 0/1 = 0



Betweenness Centrality for A = ?

Betweenness Centrality for A = ?

BC = 0/1 = 0

BD = 0/1 = 0

BE = 1/1 = 1

BF = 0/1 = 0

CD = 0/1 = 0

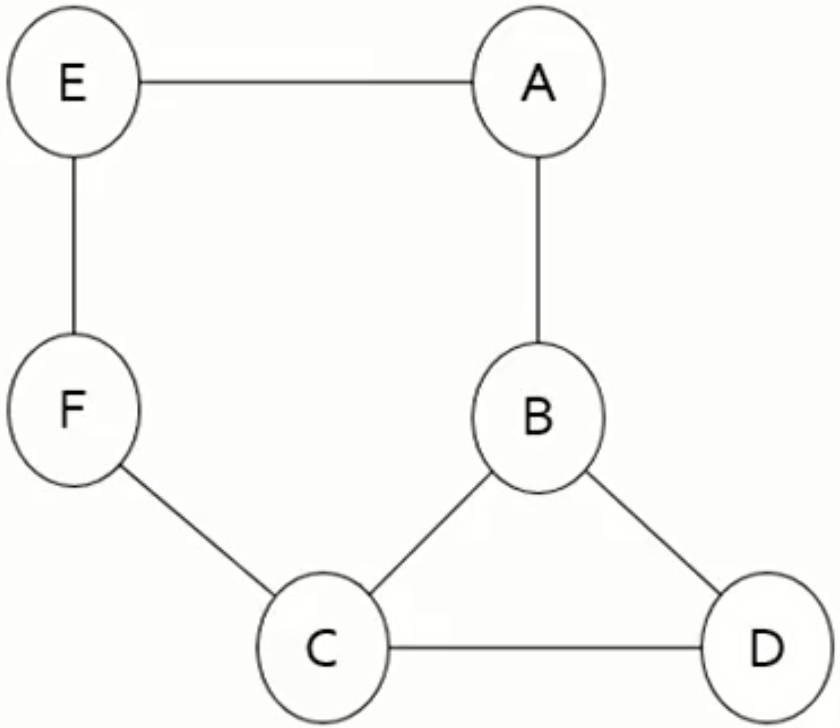
CE = 0/1 = 0

CF = 0/1 = 0

DE = 1/2 = 0.5

DF = 0/1 = 0

EF = 0/1 = 0



Betweenness Centrality for A = **1.5**

Betweenness Centrality for B = ?

AC = 1/1 = 1

AD = 1/1 = 1

AE = 0/1 = 0

AF = 0/1 = 0

CD = 0/1 = 0

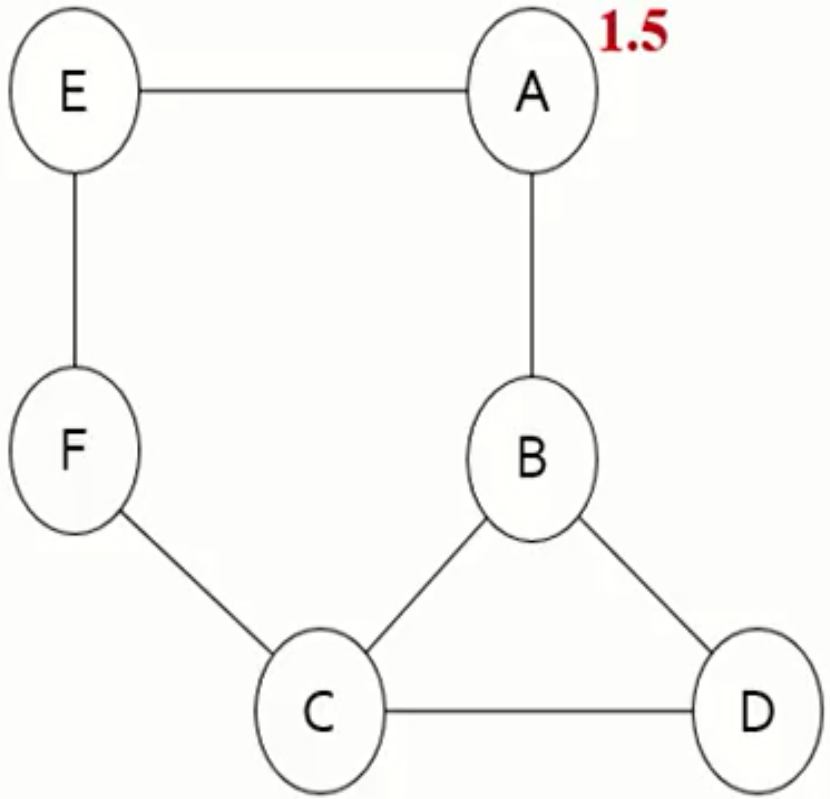
CE = 0/1 = 0

CF = 0/1 = 0

DE = 1/2 = 0.5

DF = 0/1 = 0

EF = 0/1 = 0



Betweenness Centrality for B = ?

AC = 1/1 = 1

AD = 1/1 = 1

AE = 0/1 = 0

AF = 0/1 = 0

CD = 0/1 = 0

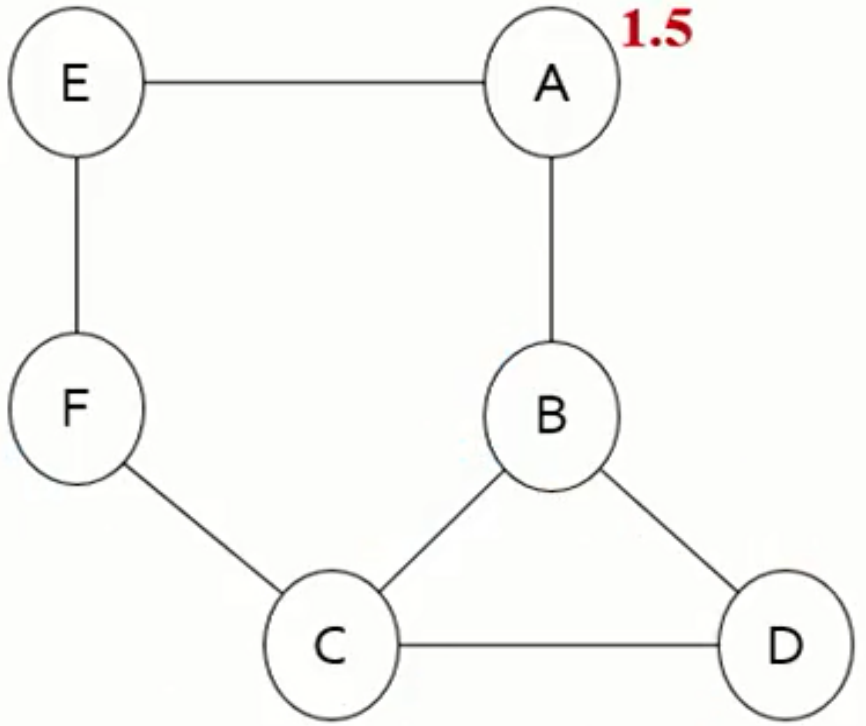
CE = 0/1 = 0

CF = 0/1 = 0

DE = 1/2 = 0.5

DF = 0/1 = 0

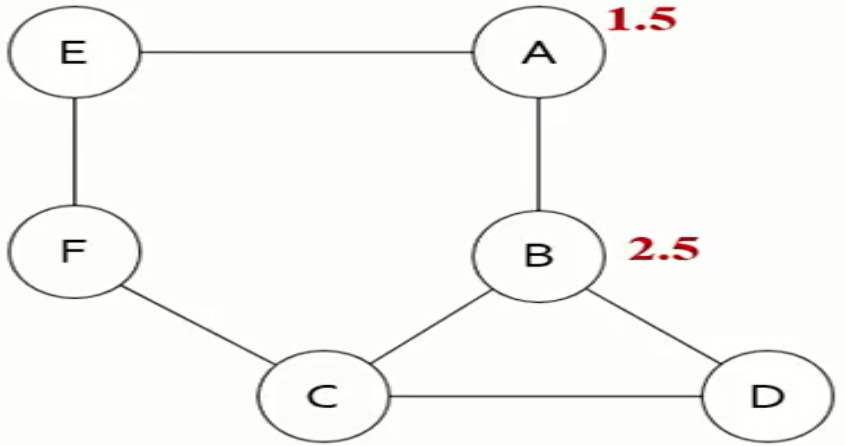
EF = 0/1 = 0



Betweenness Centrality for B = **2.5**

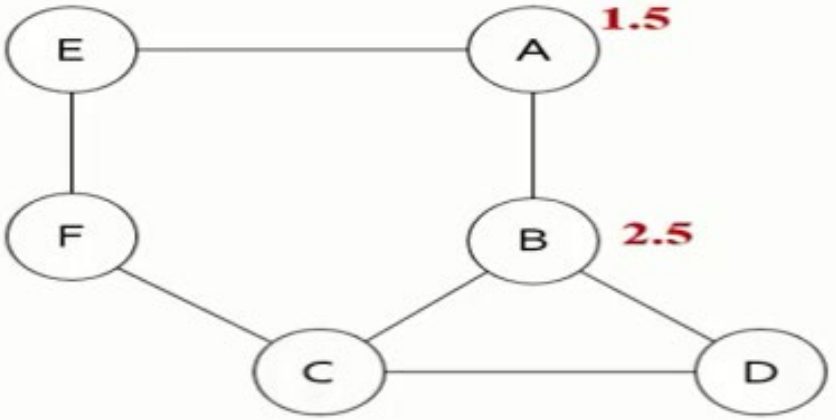
Betweenness Centrality for C = ?

AB = 0/1 = 0
AD = 0/1 = 0
AE = 0/1 = 0
AF = 0/1 = 0
BD = 0/1 = 0
BE = 0/1 = 0
BF = 1/1 = 1
DE = 1/2 = 0.5
DF = 1/1 = 1
EF = 0/1 = 0



Betweenness Centrality for C = ?

AB = 0/1 = 0
AD = 0/1 = 0
AE = 0/1 = 0
AF = 0/1 = 0
BD = 0/1 = 0
BE = 0/1 = 0
BF = 1/1 = 1
DE = 1/2 = 0.5
DF = 1/1 = 1
EF = 0/1 = 0



Betweenness Centrality for C = **2.5**

Betweenness Centrality for D = ?

AB = 0/1 = 0

AC = 0/1 = 0

AE = 0/1 = 0

AF = 0/1 = 0

BC = 0/1 = 0

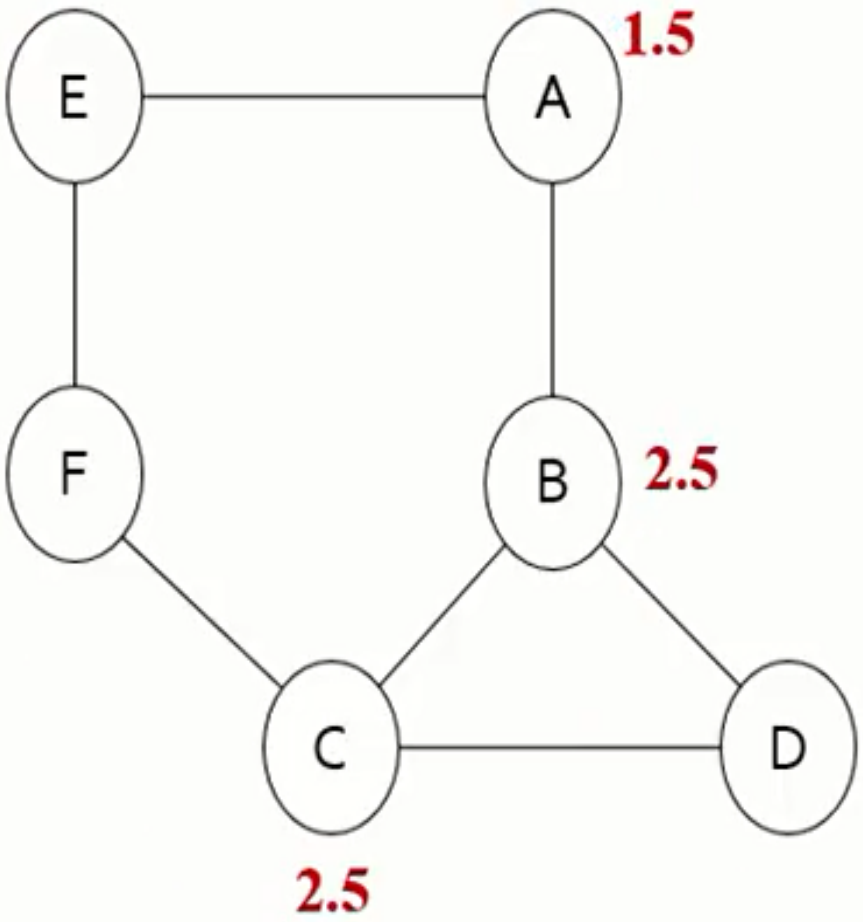
BE = 0/1 = 0

BF = 0/1 = 0

CE = 0/1 = 0

CF = 0/1 = 0

EF = 0/1 = 0



Betweenness Centrality for D = **0**

Betweenness Centrality for E = ?

AB = 0/1 = 0

AC = 0/1 = 0

AD = 0/1 = 0

AF = 1/1 = 1

BC = 0/1 = 0

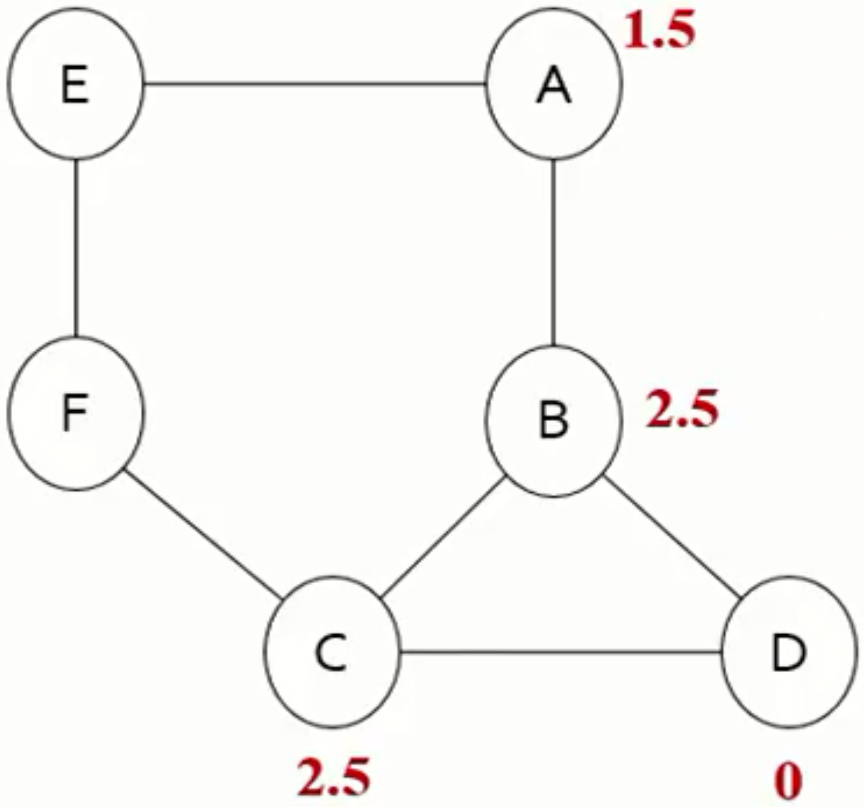
BD = 0/1 = 0

BF = 0/1 = 0

CD = 0/1 = 0

CF = 0/1 = 0

DF = 0/1 = 0



Betweenness Centrality for E = **1**

Betweenness Centrality for F = ?

AB = 0/1 = 0

AC = 0/1 = 0

AD = 0/1 = 0

AE = 0/1 = 0

BC = 0/1 = 0

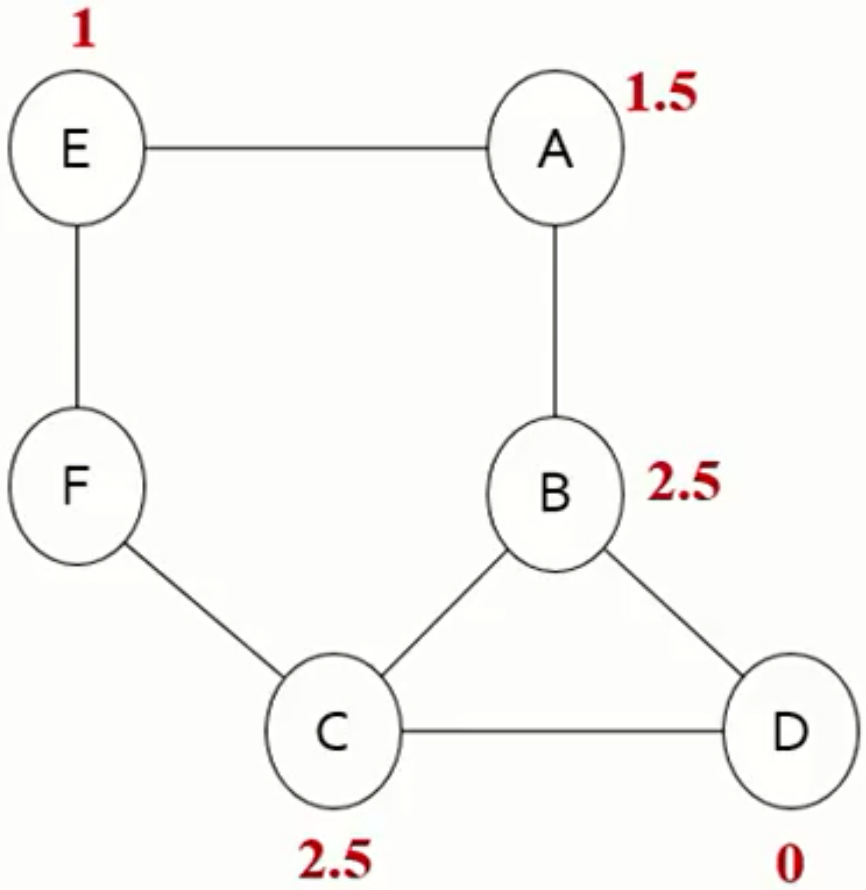
BD = 0/1 = 0

BE = 0/1 = 0

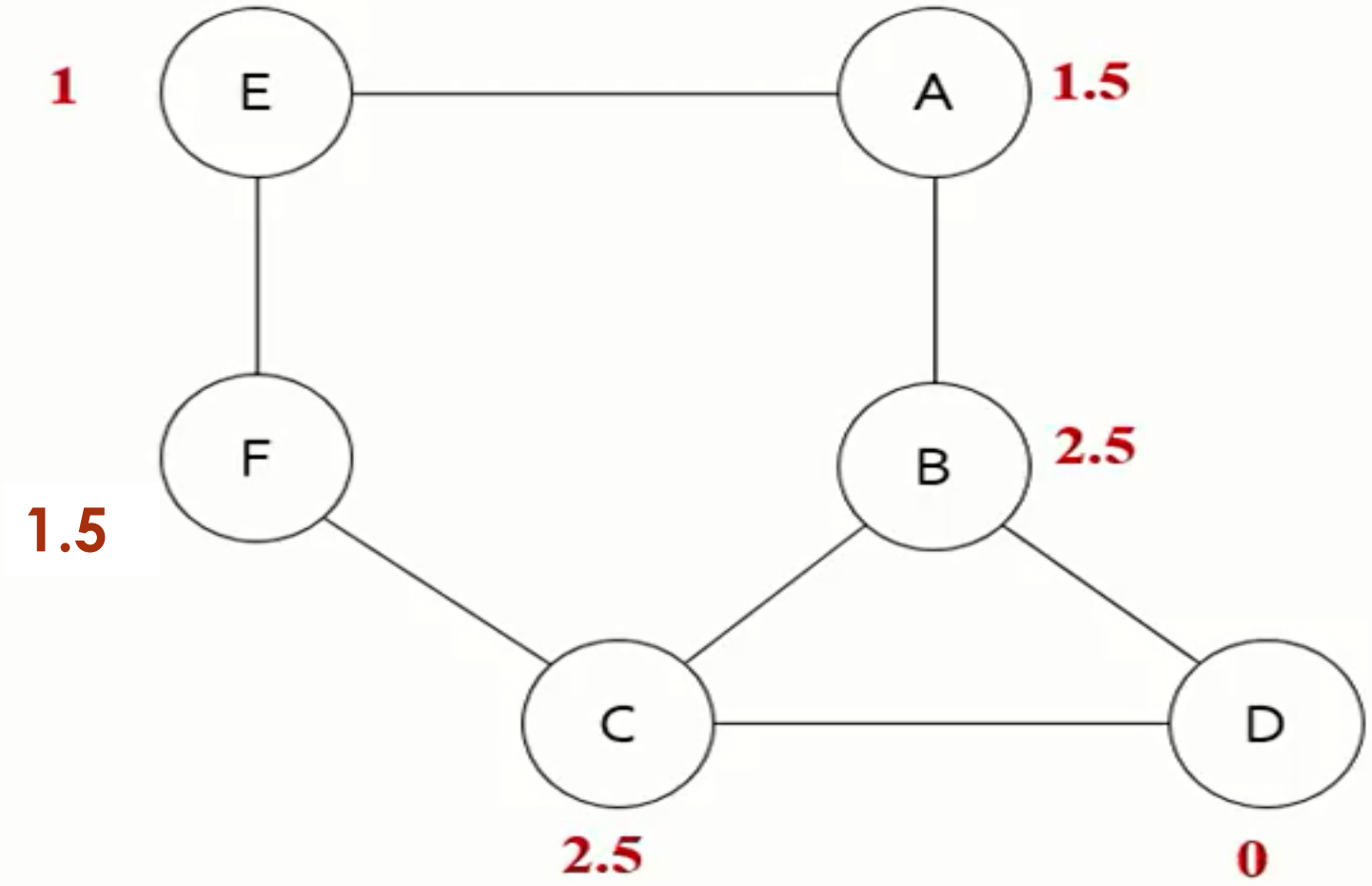
CD = 0/1 = 0

CE = 1/1 = 1

DE = 1/2 = 0.5

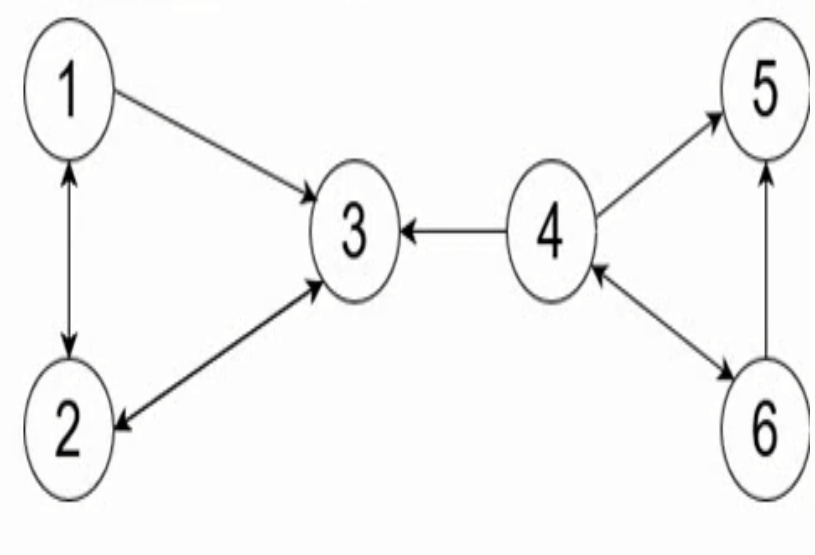


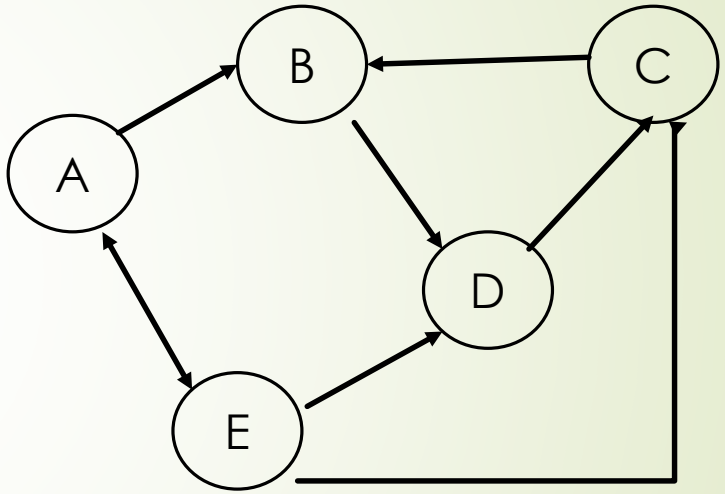
Betweenness Centrality for F = **1.5**


1
E
A
1.5
F
1.5
B
2.5
C
2.5
D
0

H.W /

Calculate The Between Centrality For All Node ?





شكرًا للاستماع

النجاح ليس عدم فعل الأخطاء

...النجاح هو عدم تكرار الخطأ...