

# Tutorial 09: Graph, DFS and BFS

---

CSCI2520 - DATA STRUCTURES AND APPLICATIONS

TUTOR: ZHANG KAI

# Outline

---

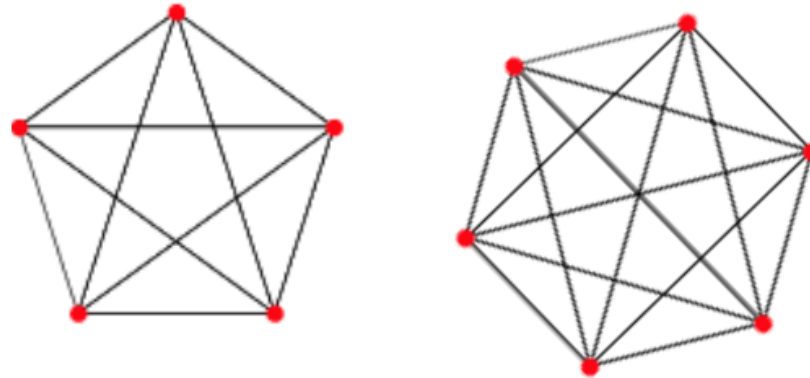
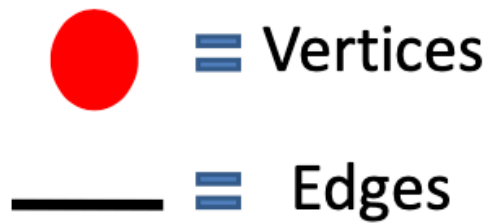
1. Graph
2. Depth First Search
3. Breadth First Search

# Graph

---

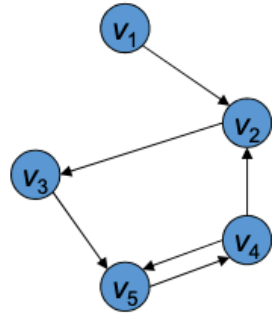
Graph is a mathematical structure used to model pair wise relations between objects from a certain collection.

A tree is also a graph.

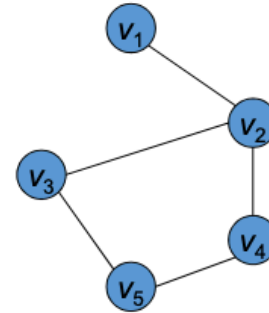


# Graph

## (Un)directed (Un)weighted Graph

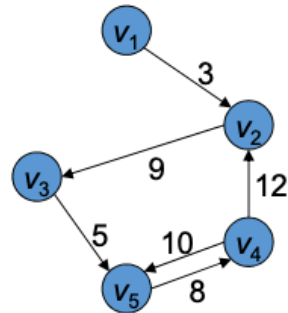


Directed unweighted graph

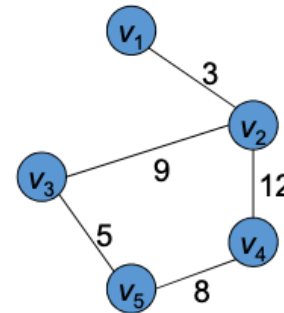


Undirected unweighted graph

All four  
graphs  
are  
different.



Directed weighted graph



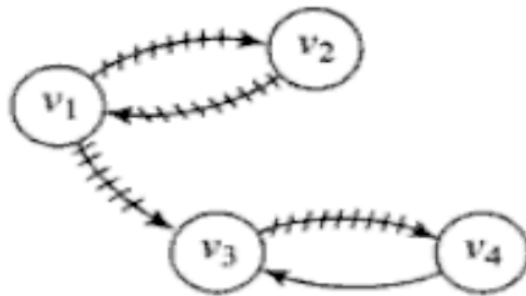
Undirected weighted graph

# Graph

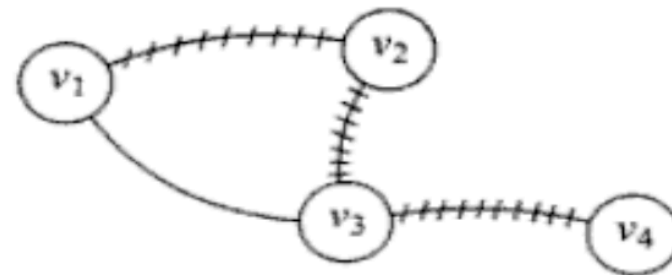
## Paths

A path in a graph is a sequence of vertices such that from each of its vertices there is an edge to the next vertex in the sequence.

The length of a path is the number of edges on it. The length can be zero for the case of a single vertex.



(a) A path from  $v_1$  to  $v_4$  in  
directed graph  $G_1$   
 $P_1 = \{v_1, v_2, v_1, v_3, v_4\}$



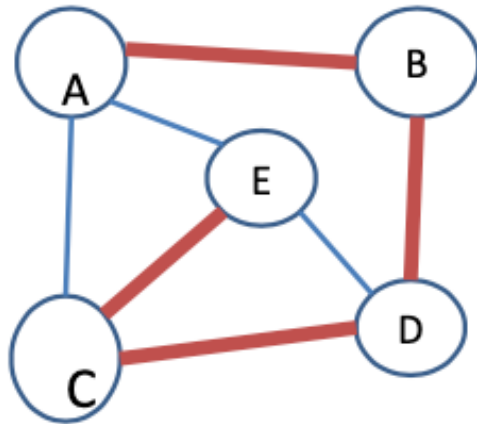
(b) A path from  $v_1$  to  $v_4$  in  
undirected graph  $G_2$   
 $P_2 = \{v_1, v_2, v_3, v_4\}$

# Graph

---

## Paths

- A path may be infinite.
- A finite path always has a first vertex, called its **start vertex**, and a last vertex, called its **end vertex**.
- Both of them are called **terminal vertices** of the path.
- The other vertices in the path are internal vertices.



Path= A B D C E

A- Start vertex

E- End vertex

# Graph

---

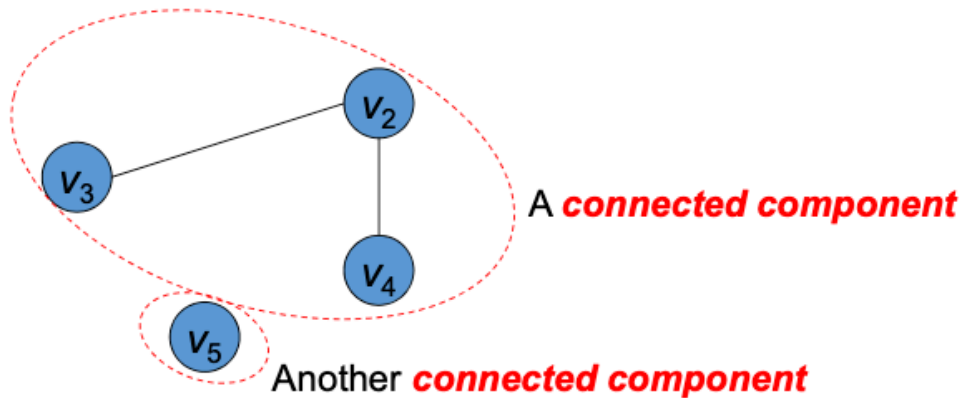
## Degree

- In an undirected graph, the degree of a vertex  $v$  is the number of adjacent vertices to  $v$ .
- In a directed graph,
  - the out-degree of a vertex  $v$  is the number of edges whose head is  $v$ .
  - the in-degree of a vertex  $v$  is the number of edges whose tail is  $v$ .

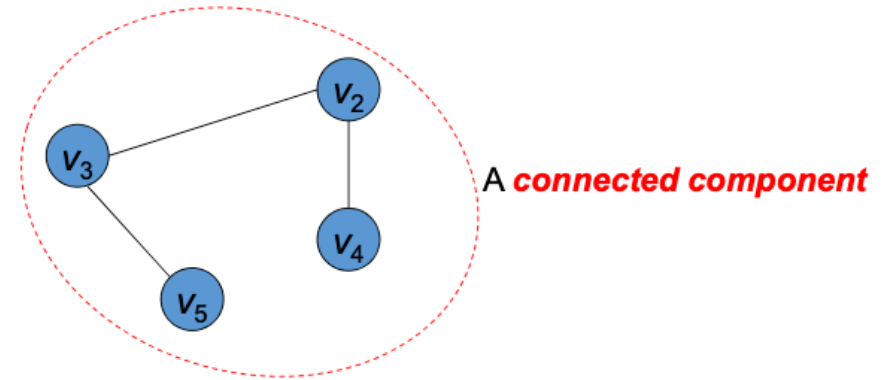
# Graph

## Graph Connectivity

This graph has 2 **connected components**.



This graph has 1 **connected component**.



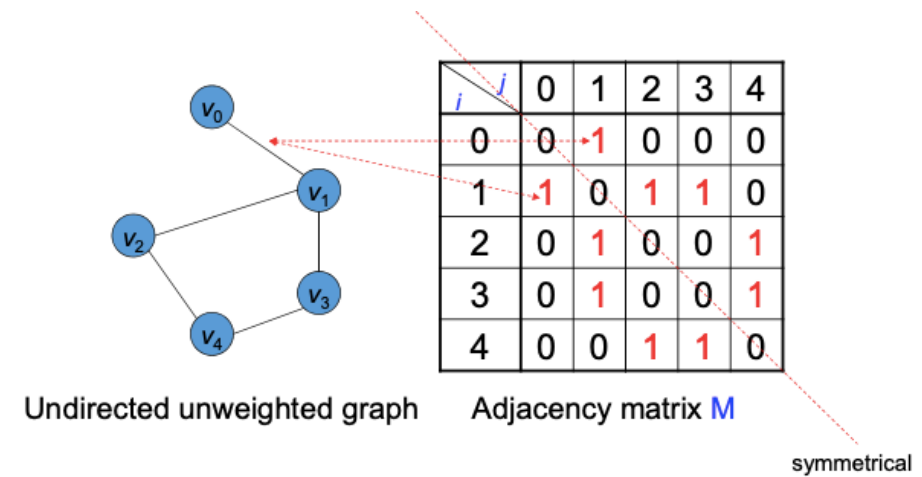
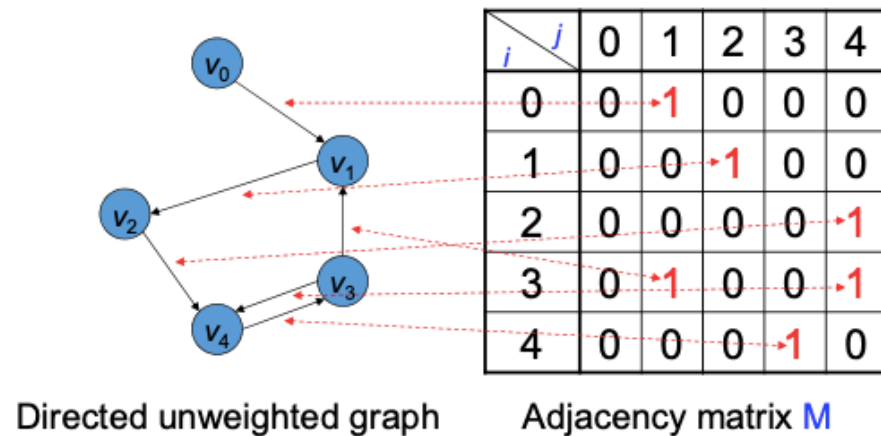
A **connected** graph is a graph that has only **one connected component**.



# Graph

## Representation

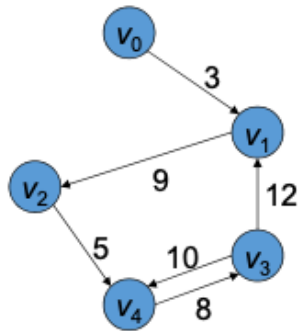
Adjacency Matrix : Unweighted Graph



# Graph

## Representation

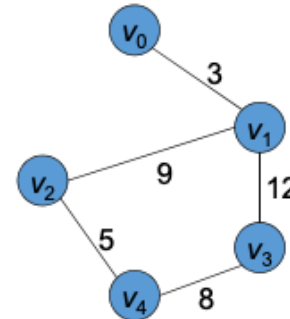
Adjacency Matrix : Weighted Graph



Directed weighted graph

$i \backslash j$	0	1	2	3	4
0	$\infty$	3	$\infty$	$\infty$	$\infty$
1	$\infty$	$\infty$	9	$\infty$	$\infty$
2	$\infty$	$\infty$	$\infty$	$\infty$	5
3	$\infty$	12	$\infty$	$\infty$	10
4	$\infty$	$\infty$	$\infty$	8	$\infty$

Adjacency matrix  $M$



Undirected weighted graph

$i \backslash j$	0	1	2	3	4
0	$\infty$	3	$\infty$	$\infty$	$\infty$
1	3	$\infty$	9	12	$\infty$
2	$\infty$	9	$\infty$	$\infty$	5
3	$\infty$	12	$\infty$	$\infty$	8
4	$\infty$	$\infty$	5	8	$\infty$

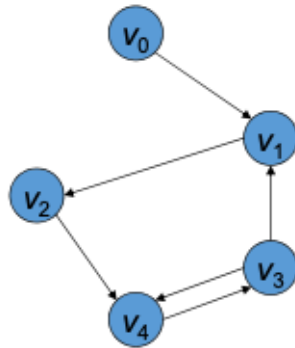
Adjacency matrix  $M$

symmetrical

# Graph

## Representation

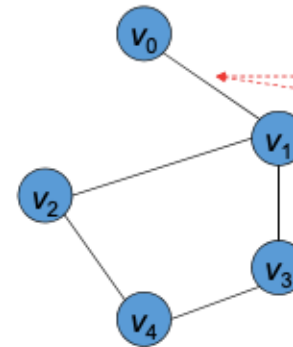
### Adjacency List : Unweighted Graph



Directed unweighted graph

$i$	$L[i]$	
0	[1]	Denotes $(v_0, v_1)$
1	[2]	Denotes $(v_1, v_2)$
2	[4]	Denotes $(v_2, v_4)$
3	[1, 4]	Denotes $(v_3, v_1)$ and $(v_3, v_4)$
4	[3]	Denotes $(v_4, v_3)$

Adjacency list  $L$



Undirected unweighted graph

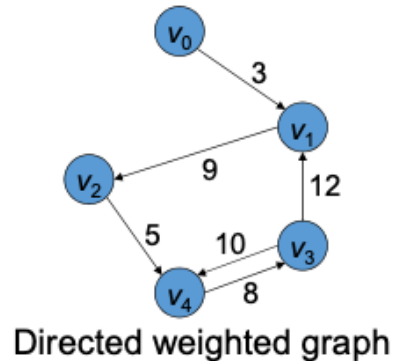
$i$	$L[i]$
0	[1]
1	[0, 2, 3]
2	[1, 4]
3	[1, 4]
4	[2, 3]

Adjacency list  $L$

# Graph

## Representation

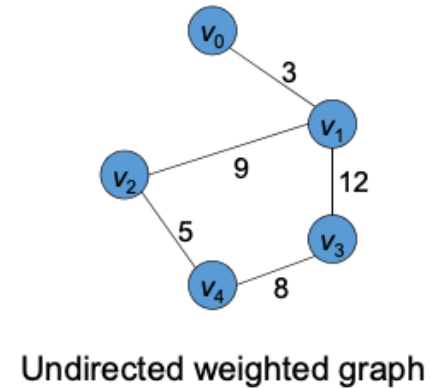
### Adjacency List : Weighted Graph



$i$	$L[i]$
0	$[\langle 1, 3 \rangle]$
1	$[\langle 2, 9 \rangle]$
2	$[\langle 4, 5 \rangle]$
3	$[\langle 1, 12 \rangle, \langle 4, 10 \rangle]$
4	$[\langle 3, 8 \rangle]$

Adjacency list  $L$

vertex weight



$i$	$L[i]$
0	$[\langle 1, 3 \rangle]$
1	$[\langle 0, 3 \rangle, \langle 2, 9 \rangle, \langle 3, 12 \rangle]$
2	$[\langle 1, 9 \rangle, \langle 4, 5 \rangle]$
3	$[\langle 1, 12 \rangle, \langle 4, 8 \rangle]$
4	$[\langle 2, 5 \rangle, \langle 3, 8 \rangle]$

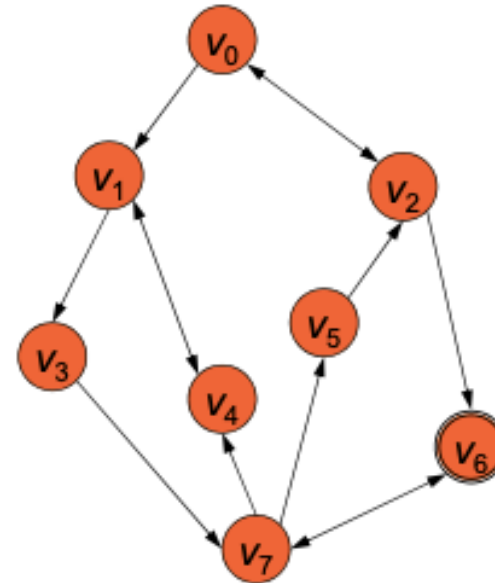
Adjacency list  $L$

# Depth First Search

Once a possible path is found, continue the search until the end of the path.

Visit all neighbors of a neighbor before visiting your other neighbors.

- Start from a vertex  $v$ .
- “Visit” vertex  $v$ . (i.e., mark  $v$  as visited.)
- For each unvisited vertex  $w$  adjacent to  $v$ 
  - Do a DFS starting from vertex  $w$ .



Order of visit:  $v_0, v_1, v_3, v_7, v_4, v_5, v_2, v_6$

# Depth First Search

---

## Analysis

For a Graph  $G=(V, E)$  and  $n = |V|$  and  $m=|E|$ ,

When Adjacency List is used, Complexity is  $O(m + n)$ .

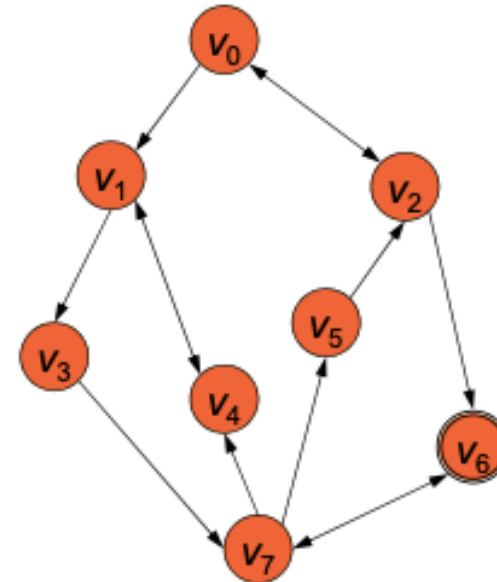
When Adjacency Matrix is used, Scanning each row for checking the connectivity of a Vertex is in order  $O(n)$ , Complexity is  $O(n^2)$ .

# Breadth First Search

Start several paths at a time, and advance in each one step at a time

Pick each child of S in turn and discover their vertices adjacent to that child.

- Start from a vertex  $v$ .
- Enqueue  $v$  to a queue  $Q$  and **mark**  $v$
- While  $Q$  is not empty
  - Dequeue a vertex  $u$  from  $Q$ .
  - “Visit”  $u$ .
  - For each un-marked vertex  $w$  adjacent to  $u$ 
    - Enqueue  $w$  to  $Q$  and **mark**  $w$ .




Order of visit:  $v_0, v_1, v_2, v_3, v_4, v_6, v_7, v_5$

# Exercise 1

---

Which of the following is an advantage of adjacency list representation over adjacency matrix representation of a graph?

- (A) In adjacency list representation, space is saved for sparse graphs
- (B) DFS and BSF can be done in  $O(V + E)$  time for adjacency list representation. These operations take  $O(V^2)$  time in adjacency matrix representation. Here  $V$  and  $E$  are number of vertices and edges respectively.
- (C) Adding a vertex in adjacency list representation is easier than adjacency matrix representation
-  (D) All of the above




# Exercise 2

---

Which of the following statements is/are TRUE for an undirected graph?

P: Number of odd degree vertices is even

Q: Sum of degrees of all vertices is even

- (A) P Only
- (B) Q Only
-  (C) Both P and Q
- (D) Neither P nor Q


P is true for undirected graph as adding an edge always increases degree of two vertices by 1.

Q is true: If we consider sum of degrees and subtract all even degrees, we get an even number because every edge increases the sum of degrees by 2. So total number of odd degree vertices must be even.

# Exercise 3

---

Given an undirected graph  $G$  with  $V$  vertices and  $E$  edges, the sum of the degrees of all vertices is


- (A)  $E$
-  (B)  $2E$
- (C)  $V$
- (D)  $2V$

Since the given graph is undirected, every edge contributes as 2 to sum of degrees. So the sum of degrees is  $2E$ .

# Exercise 4

---

Which of the following data structure is useful in traversing a given graph by breadth first search?

- (A) Stack
- (B) List
-  (C) Queue
- (D) None of the above.

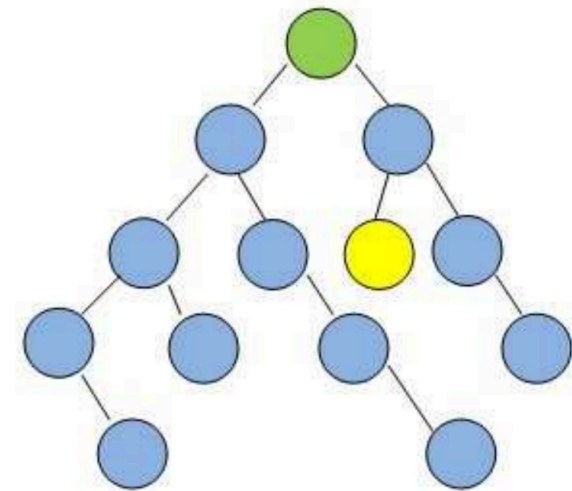
# Exercise 5

---

In the following graphs, assume that if there is ever a choice amongst multiple nodes, both the BFS and DFS algorithms will choose the left-most node first.

Starting from the green node at the top, which algorithm will visit **the least** number of nodes **before visiting the yellow goal node**?

- ☒ (A) BFS
- (B) DFS
- (C) Neither BFS nor DFS will ever encounter the goal node in this graph
- (D) BFS and DFS encounter same number of nodes before encounter the goal node



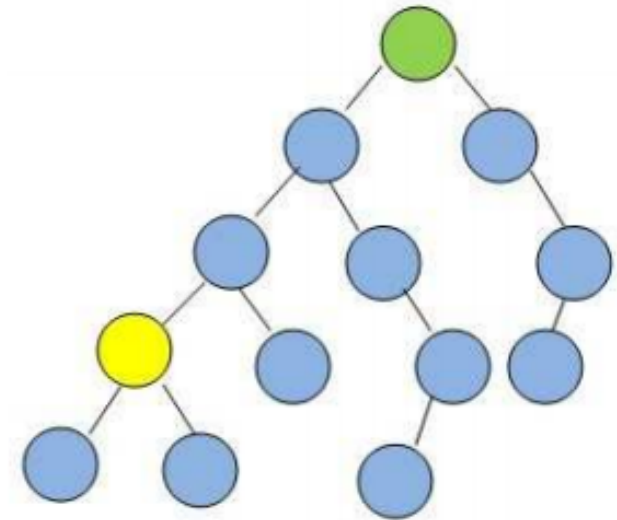
# Exercise 6

---

In the following graphs, assume that if there is ever a choice amongst multiple nodes, both the BFS and DFS algorithms will choose the left-most node first.

Starting from the green node at the top, which algorithm will visit **the least** number of nodes **before visiting the yellow goal node**?

- (A) BFS
- ☒ (B) DFS
- (C) Neither BFS nor DFS will ever encounter the goal node in this graph
- (D) BFS and DFS encounter same number of nodes before encounter the goal node



# Exercise 7

---

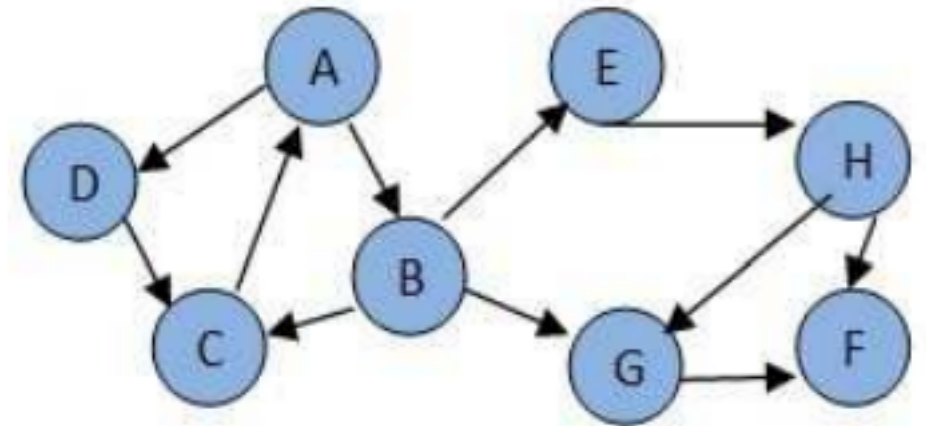
Consider the following graph. If there is ever a decision between multiple neighbor nodes in the BFS or DFS algorithms, assume we always choose the letter closest to the beginning of the alphabet first.

In what order will the nodes be visited using a Breadth First Search?

The answer is: **ABDCEGHF**

In what order will the nodes be visited using a Depth First Search?

The answer is: **ABCEHFGD**



# Exercise 8

---

Consider an undirected graph with  $N$  vertices,

- at most how many connected components?  **$N$**
- at least how many connected components?  **$1$**
- If we can visit all the vertices from any start vertex in one DFS, then the graph is **Connected Graph**

# References

---

<https://www.geeksforgeeks.org/data-structure-gq/graph-gq/>