# Graph Data-Structure

Fundamentals and Basic Terminologies..

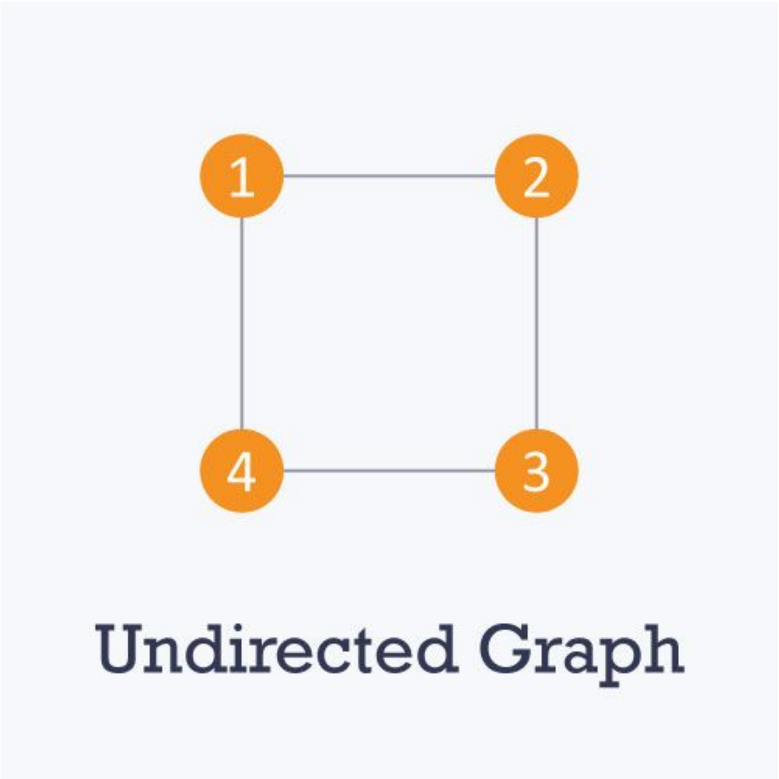
# Basic Graph Terminologies

Graphs are mathematical structures that represent pairwise relationships between objects. A graph is a flow structure that represents the relationship between various objects. It can be visualized by using the following two basic components:

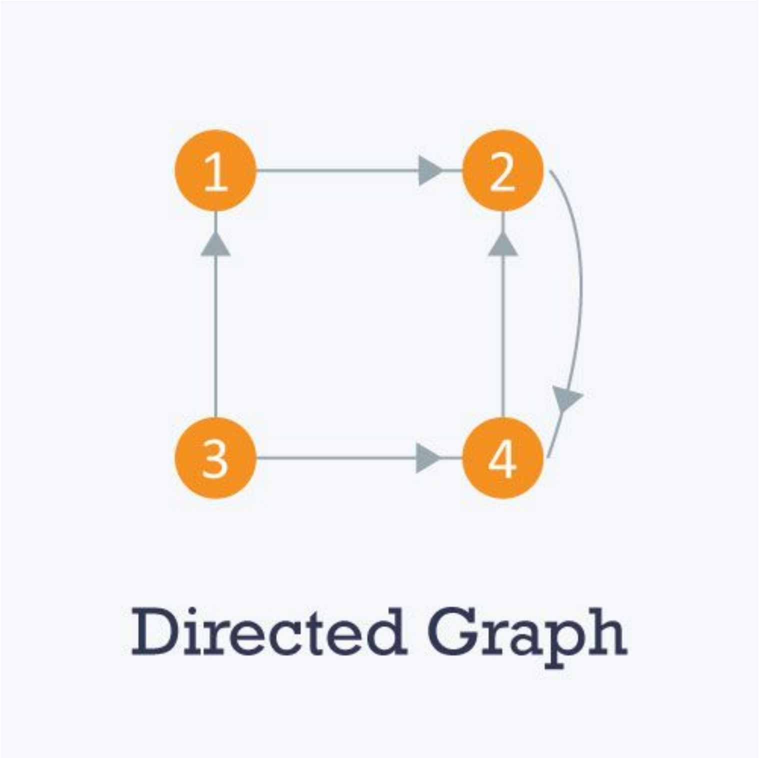
- **Nodes:** These are the most important components in any graph. Nodes are entities whose relationships are expressed using edges. If a graph comprises 2 nodes A and B and an undirected edge between them, then it expresses a bi-directional relationship between the nodes and edge.
- **Edges:** Edges are the components that are used to represent the relationships between various nodes in a graph. An edge between two nodes expresses a one-way or two-way relationship between the nodes.

# TYPES OF GRAPH

Undirected: An undirected graph is a graph in which all the edges are bi-directional i.e. the edges do not point in any specific direction.



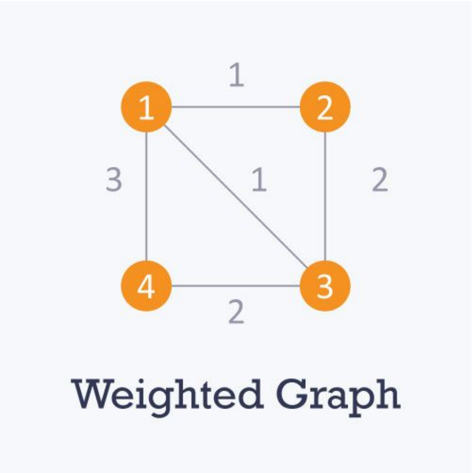
- Directed: A directed graph is a graph in which all the edges are uni-directional i.e. the edges point in a single direction.



Weighted: In a weighted graph, each edge is assigned a weight or cost. Consider a graph of 4 nodes as in the diagram below. As you can see each edge has a weight/cost assigned to it. If you want to go from vertex 1 to vertex 3, you can take one of the following 3 paths:

- 1 -> 2 -> 3
- 1 -> 3
- 1 -> 4 -> 3

Therefore the total cost of each path will be as follows: - The total cost of 1 -> 2 -> 3 will be (1 + 2) i.e. 3 units - The total cost of 1 -> 3 will be 1 unit - The total cost of 1 -> 4 -> 3 will be (3 + 2) i.e. 5 units



# #Important Points

Cyclic: A graph is cyclic if the graph comprises a path that starts from a vertex and ends at the same vertex. That path is called a cycle. An acyclic graph is a graph that has no cycle.

A tree is an undirected graph in which any two vertices are connected by only one path. A tree is an acyclic graph and has N - 1 edges where N is the number of vertices. Each node in a graph may have one or multiple parent nodes. However, in a tree, each node (except the root node) comprises exactly one parent node.

*Note*: A root node has no parent.
A tree cannot contain any cycles or self loops, however, the same does not apply to graphs.

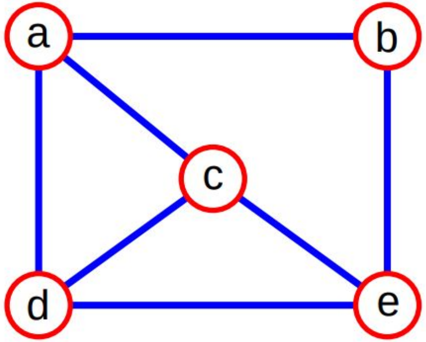
# What is a Graph?

- A **graph G** = (**V**,**E**) is composed of:

  **V**: set of *vertices*

  **E**: set of *edges* connecting the *vertices* in **V**

- An **edge e** = (u,v) is a pair of vertices
- Example:

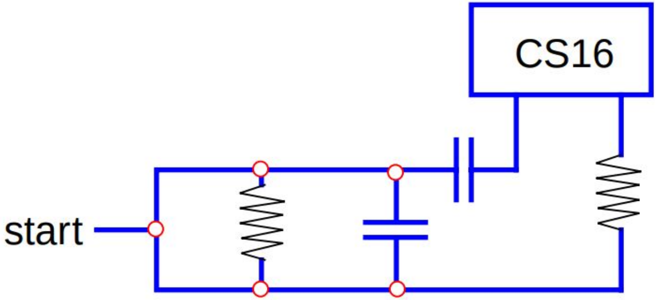



**V**= {a,b,c,d,e}

**E**= {(a,b),(a,c),(a,d), (b,e),(c,d),(c,e), (d,e)}

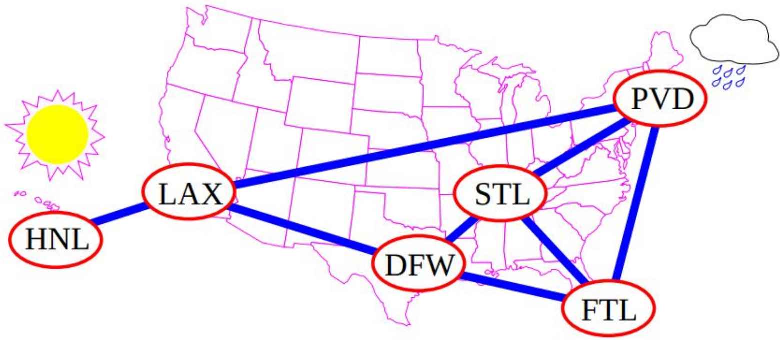
# Applications

- electronic circuits

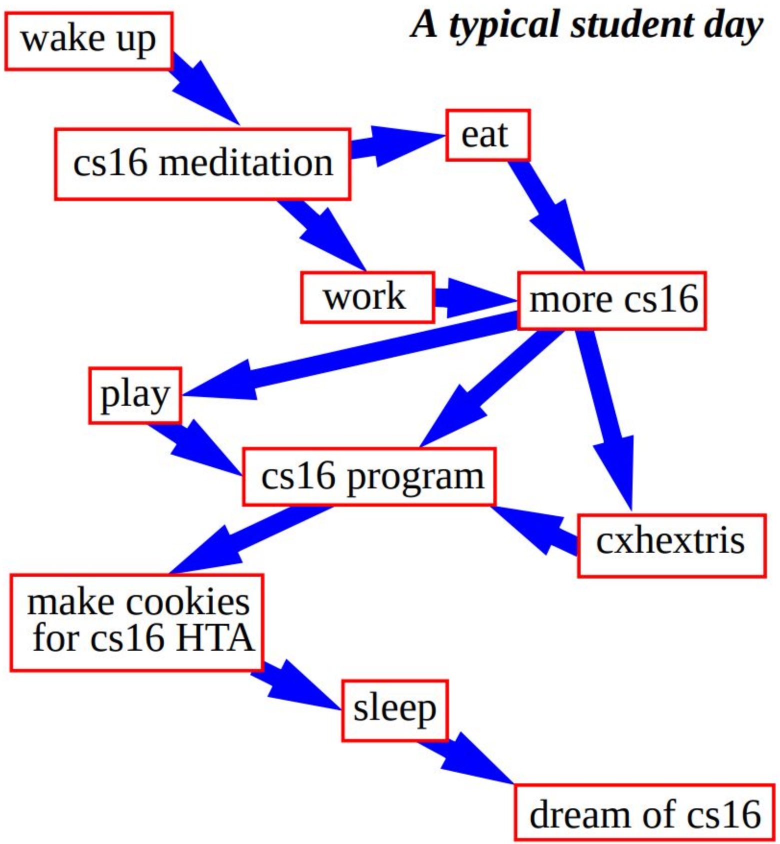



***find the path of least resistance to CS16***

- networks (roads, flights, communications)

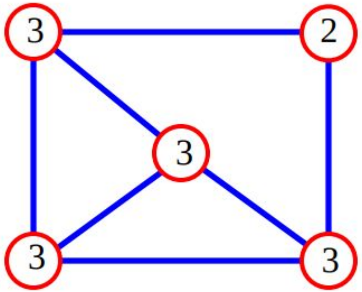


- scheduling (project planning)



# Graph Terminology

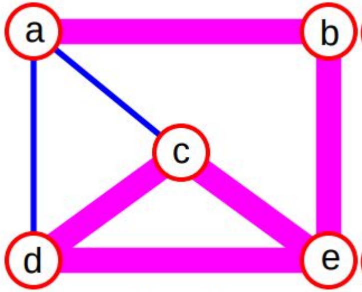
- adjacent vertices: connected by an edge
- degree (of a vertex): # of adjacent vertices



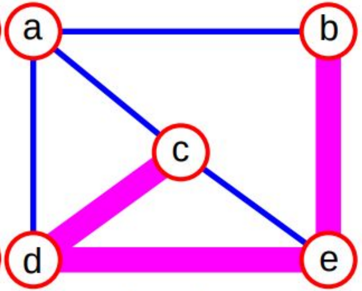

$$\sum_{v \in V} \deg(v) = 2(\text{\# edges})$$

- Since adjacent vertices each count the adjoining edge, it will be counted twice

path: sequence of vertices $v_1, v_2, \ldots v_k$ such that consecutive vertices $v_i$ and $v_{i+1}$ are adjacent.

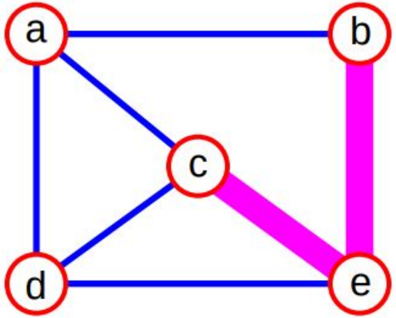



a b e d c



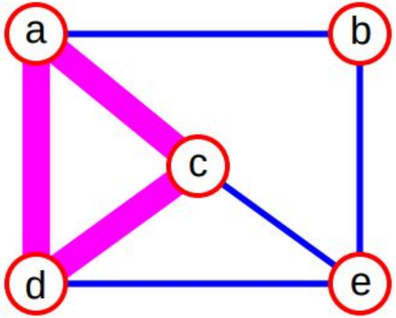

b e d c

- simple path:  no repeated vertices



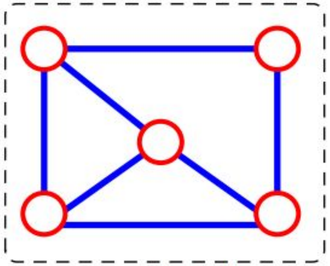

b e c

- cycle:  simple path, except that the last vertex is the same as the first vertex

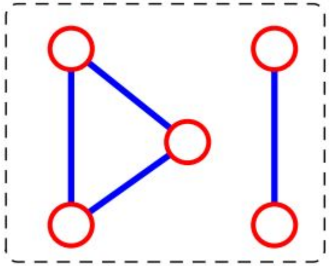



a c d a

- connected graph: any two vertices are connected by some path



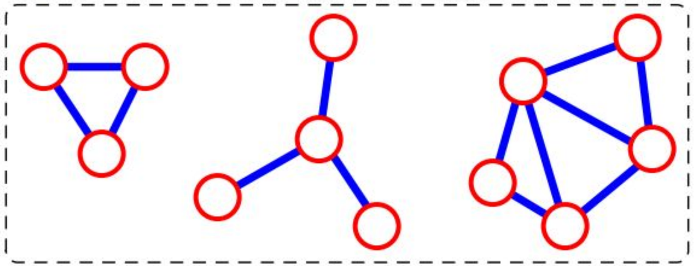
connected



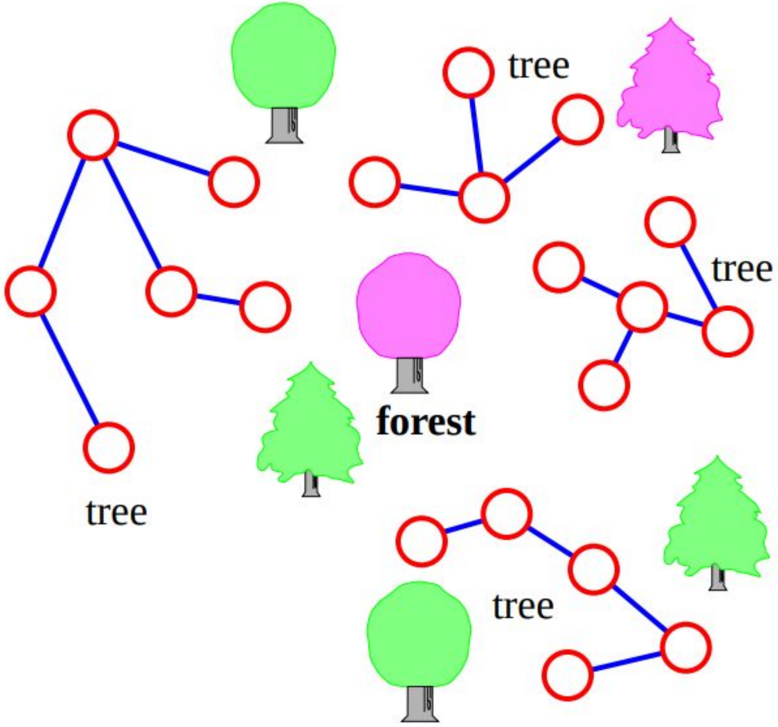
not connected

- subgraph: subset of vertices and edges forming a graph

- connected component: maximal connected subgraph. E.g., the graph below has 3 connected components.



- (free) tree - connected graph without cycles
- forest - collection of trees



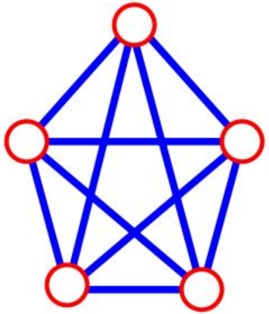
# Connectivity

Let **n** = #vertices
**m** = #edges

- complete graph - all pairs of vertices are adjacent

$$\mathbf{m} = (1/2)\sum_{v \in \mathbf{V}} \deg(\mathbf{v}) = (1/2)\sum_{v \in V} (\mathbf{n} - 1) = \mathbf{n}(\mathbf{n}-1)/2$$

- Each of the **n** vertices is incident to **n** - 1 edges, however, we would have counted each edge twice!!! Therefore, intuitively, **m** = **n**(**n**-1)/2.

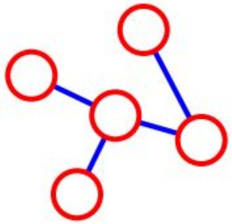


$\mathbf{n} = 5$
$\mathbf{m} = (5 * 4)/2 = 10$

- Therefore, if a graph is ***not*** complete,
  $\mathbf{m} < \mathbf{n}(\mathbf{n}-1)/2$

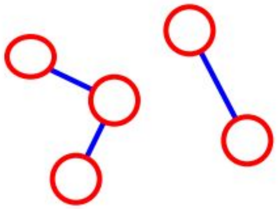
**n** = #vertices
**m** = #edges

- For a tree **m** = **n** - 1



**n** = 5
**m** = 4

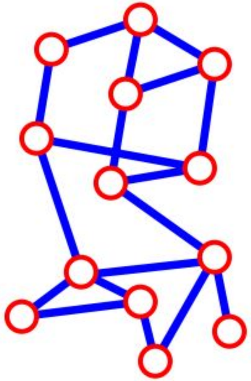
- If **m** < **n** - 1, G is not connected



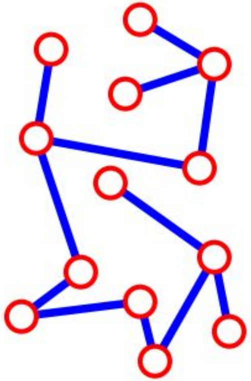
**n** = 5
**m** = 3

# Spanning Tree

- A **spanning tree** of **G** is a subgraph which
  - is a tree
  - contains all vertices of **G**



**G**



spanning tree of **G**

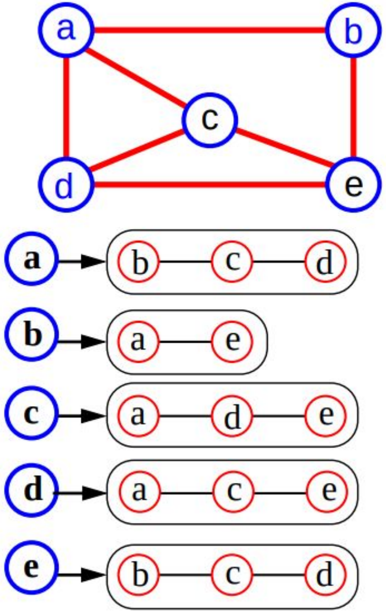
- Failure on any edge disconnects system (least fault tolerant)

# Adjacency List
# (traditional)

- adjacency list of a vertex v:

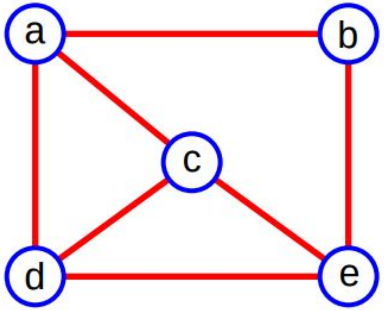
  sequence of vertices adjacent to v

- represent the graph by the adjacency lists of all the vertices




- Space = $\Theta(\mathbf{N} + \sum \deg(\mathbf{v})) = \Theta(\mathbf{N} + \mathbf{M})$

# Adjacency Matrix (traditional)

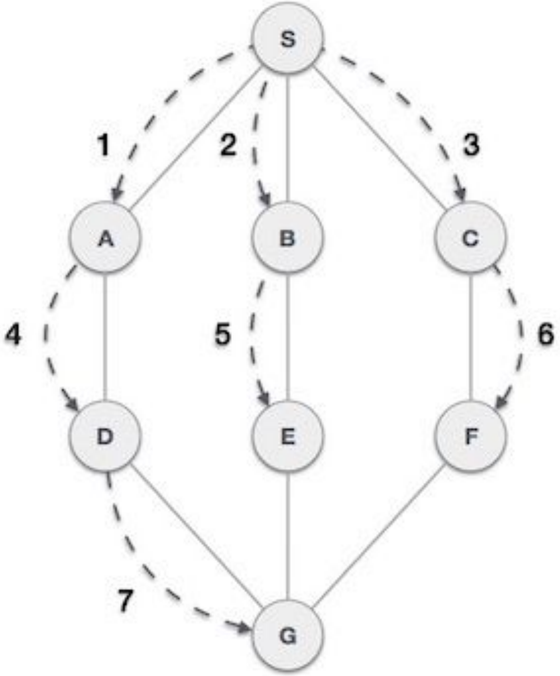


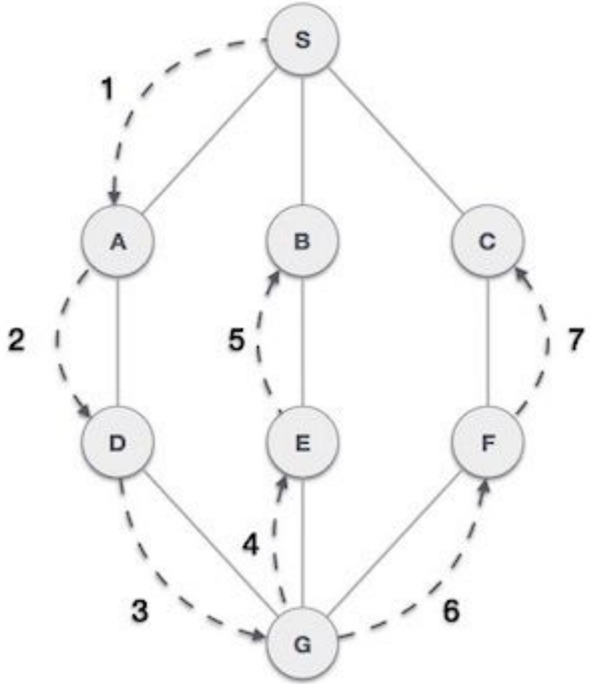

| | a | b | c | d | e |
|---|---|---|---|---|---|
| a | F | T | T | T | F |
| b | T | F | F | F | T |
| c | T | F | F | T | T |
| d | T | F | T | F | T |
| e | F | T | T | T | F |

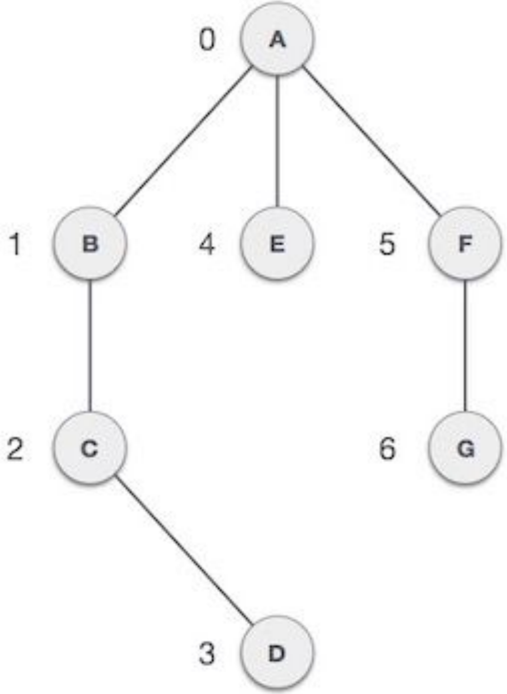
- matrix M with entries for all pairs of vertices
- M[i,j] = true means that there is an edge (i,j) in the graph.
- M[i,j] = false means that there is no edge (i,j) in the graph.
- There is an entry for every possible edge, therefore:

  Space = $\Theta(\mathbf{N^2})$

# Graphs..{BFS and DFS}



Continue with Graph traversal in detail in Next Lecture..