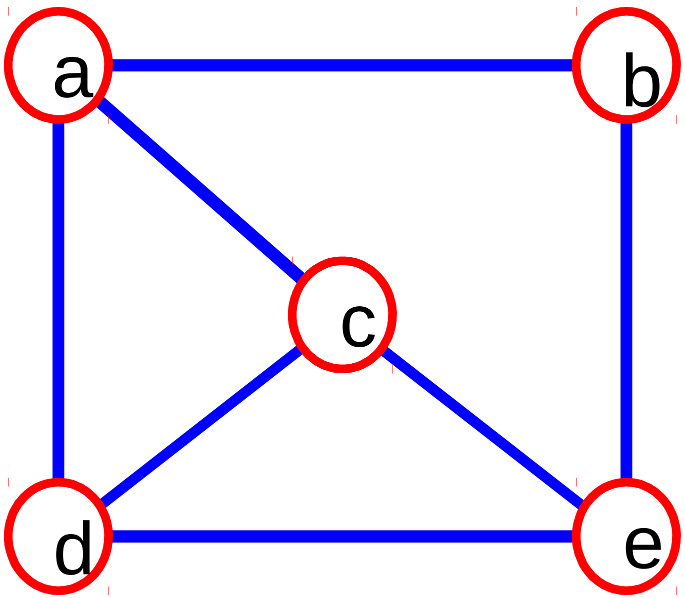
Graphs

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:

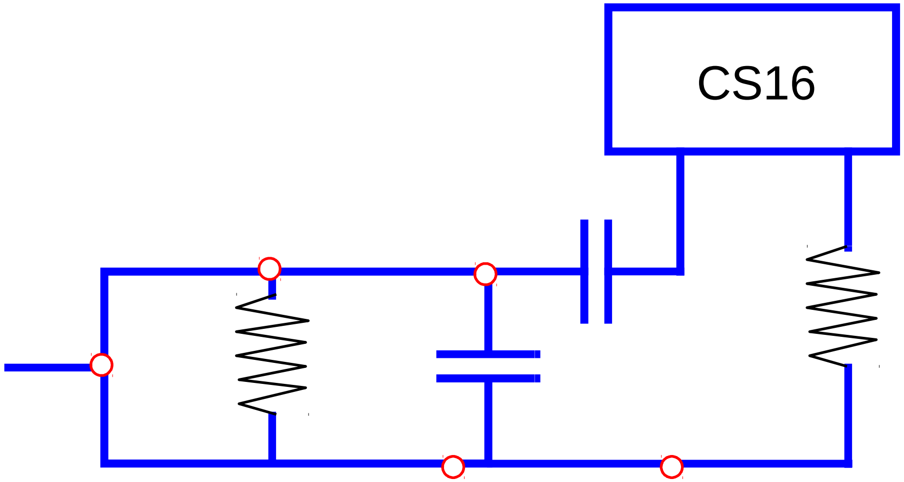


V= {a,b,c,d,e}

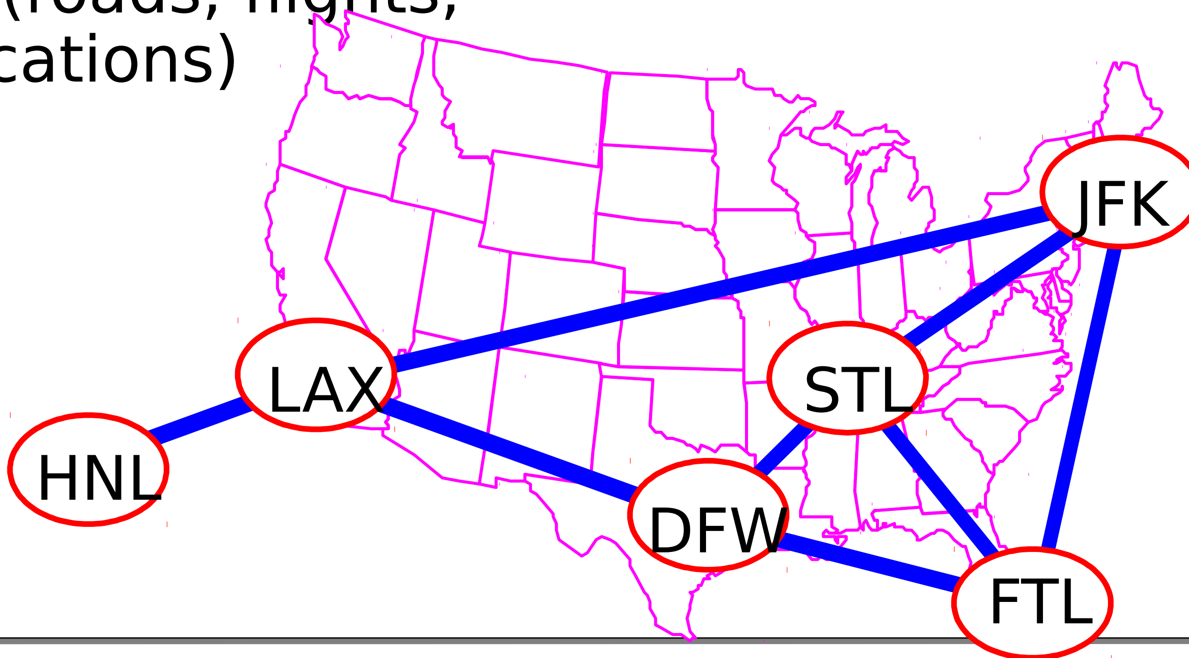
E= {(a,b),(a,c),(a,d),
(b,e),(c,d),(c,e),
(d,e)}

Applications

- electronic circuits



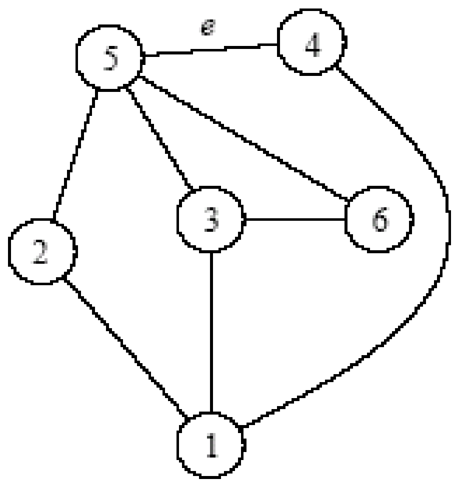
- networks (roads, flights, communications)



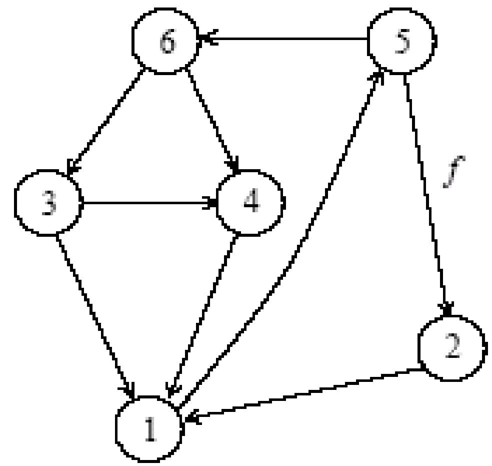
Type of Graphs

- Two type of Graphs:
  - Undirected graphs
  - Directed graphs
- Undirected graphs: An undirected graph is one where in each edge E is an unordered pair of vertices.
- Directed Graphs: A directed graph is one, where each edge is represented by a specific direction or by directed pair <v1,v2>, where v1 is the tail and v2 is the head of the edge

tail $\longrightarrow$ head



(a)



(b)

An undirected graph

A directed graph.

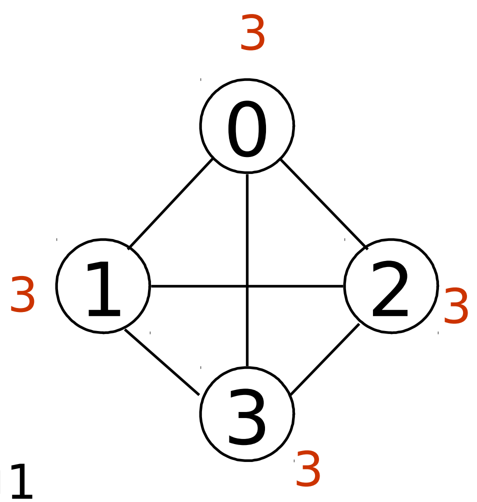
Terminology: Degree of a Vertex

- The degree of a vertex is the number of edges incident to that vertex
- For directed graph,
  - the in-degree of a vertex *v* is the number of edges that have *v* as the head
  - the out-degree of a vertex *v* is the number of edges that have *v* as the tail
  - if *di* is the degree of a vertex *i* in a graph *G* with *n* vertices and *e* edges, the number of edges is

$$e = \left(\sum_{0}^{n-1} d_i\right) / 2$$

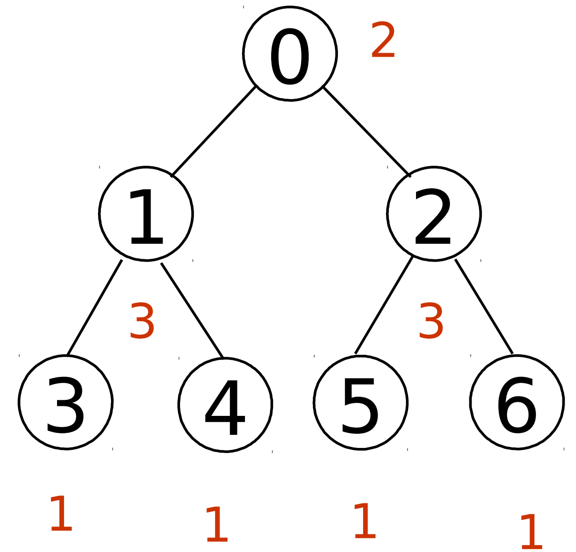
Why? Since adjacent vertices each count the adjoining edge, it will be counted twice

Examples



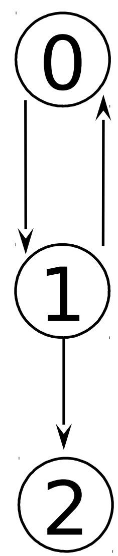
G_1

No of edges- 6



G_2

No of edges- 6



in:1, out: 1

in: 1, out: 2

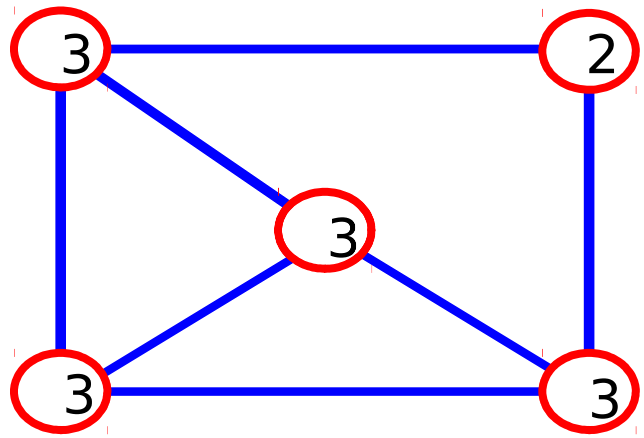
in: 1, out: 0

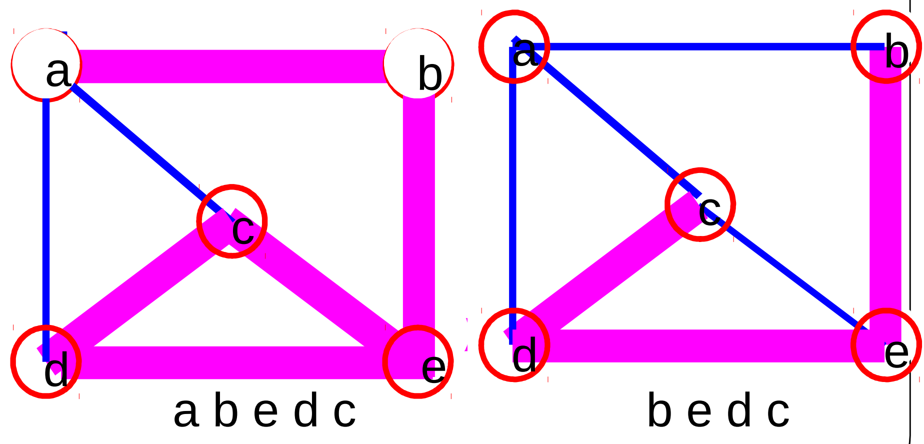
in-degree
out-degree

G_3

Terminology: Path

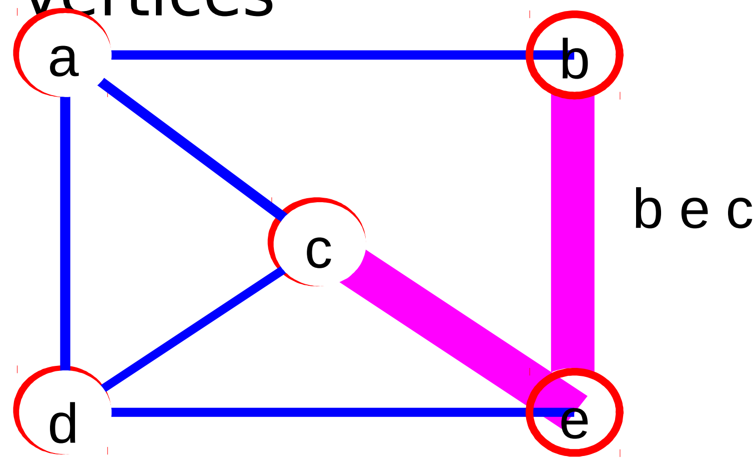
- path: sequence of vertices $v_1, v_2, \ldots v_k$ such that consecutive vertices v_i and v_{i+1} are adjacent.



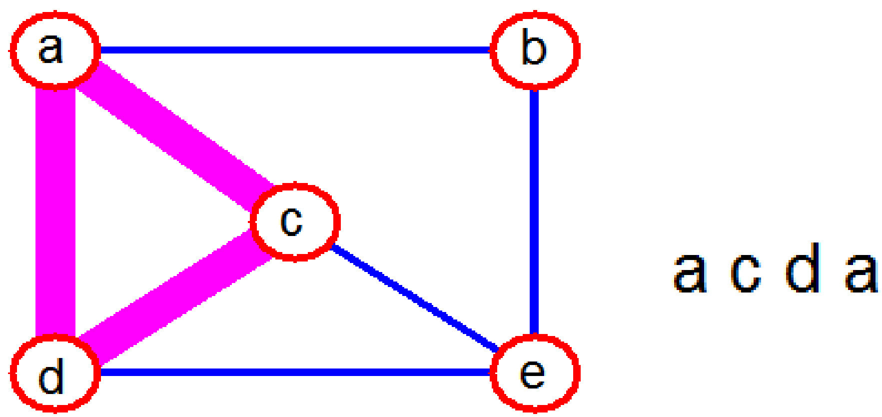


More Terminology

- simple path: no repeated vertices

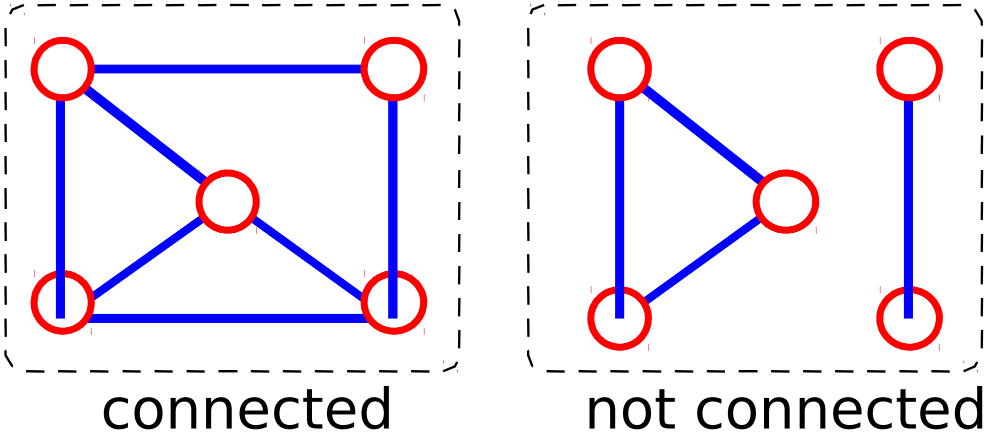


- cycle: simple path, except that the last vertex is the same as the first vertex



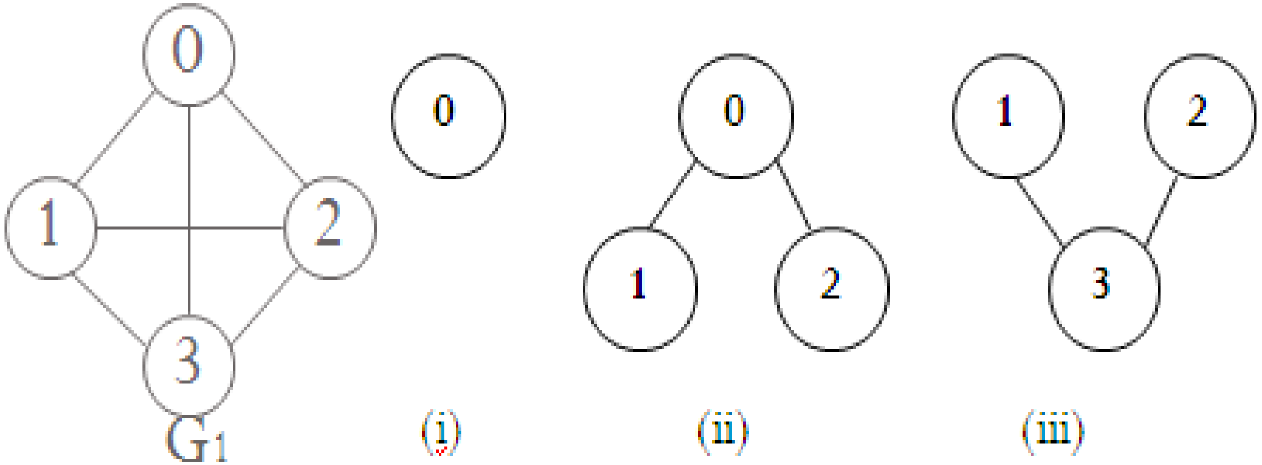
Even More Terminology

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



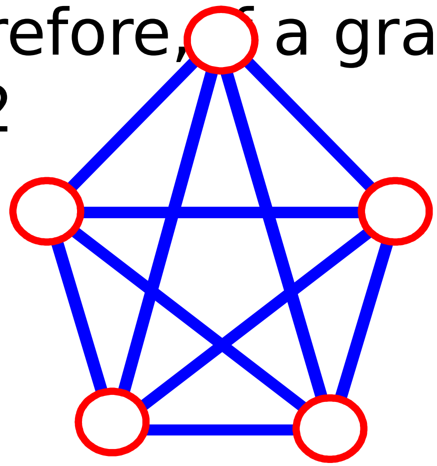
- subgraph: subset of vertices and edges forming a graph

Subgraph Examples



Connectivity

- **A complete graph**: one in which all pairs of vertices are adjacent
- Let **n** = #vertices, and **m** = #edges
- *How many total edges in a complete graph?*
  - 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.
- Therefore, if a graph is not complete, m < **n**(**n** -1)/2



n = 5

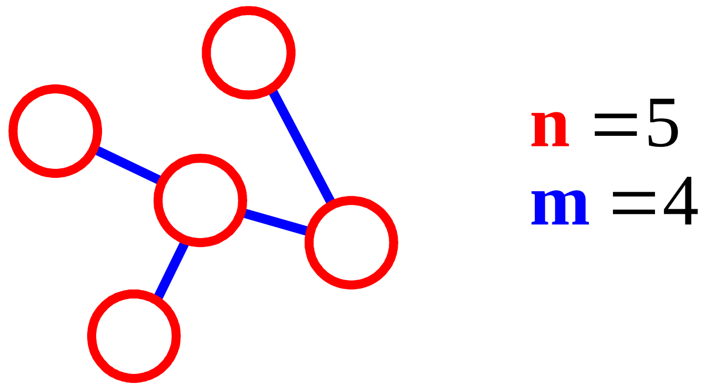
m = (5 ⋆ 4)/2 = 10

More Connectivity

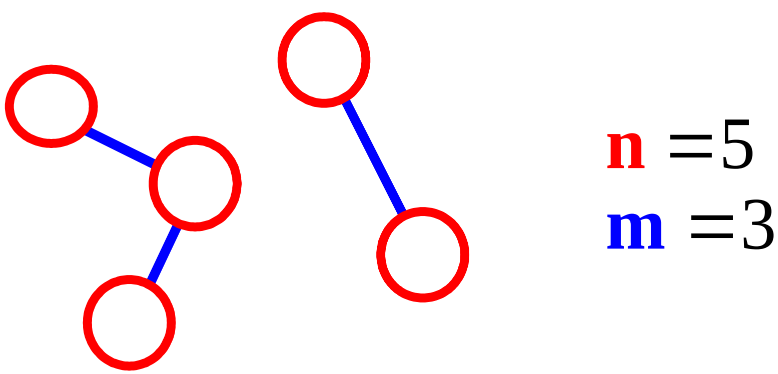
n = #vertices

m = #edges

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



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



ADT for Graph

objects: a nonempty set of vertices and a set of undirected edges, where each edge is a pair of vertices

functions: for all *graph* $\in$ *Graph*, v, v_1 and $v_2 \in$ *Vertices*

Graph Create()::=return an empty graph

Graph InsertVertex(*graph*, v)::= return a graph with v inserted. v has no incident edge.

Graph InsertEdge(*graph*, v_1,v_2)::= return a graph with new edge between v_1 and v_2

Graph DeleteVertex(*graph*, v)::= return a graph in which v and all edges incident to it are removed

Graph DeleteEdge(*graph*, v_1, v_2)::=return a graph in which the edge (v_1, v_2) is removed

Boolean IsEmpty(*graph*)::= if (*graph*==*empty graph*) return TRUE else return FALSE

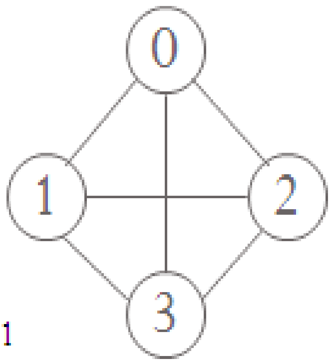
Graph Representations

- Adjacency Matrix
- Adjacency Lists

Adjacency Matrix

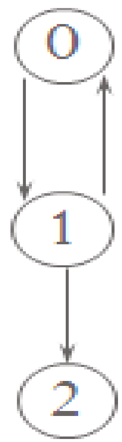
- Let G=(V,E) be a graph with n vertices.
- The adjacency matrix of G is a two-dimensional
  n by n array, say adj_mat
- If the edge (vi, vj) is in E(G), adj_mat[i][j]=1
- If there is no such edge in E(G), adj_mat[i][j]=0
- The adjacency matrix for an undirected graph is symmetric; the adjacency matrix for a digraph

Examples for Adjacency Matrix



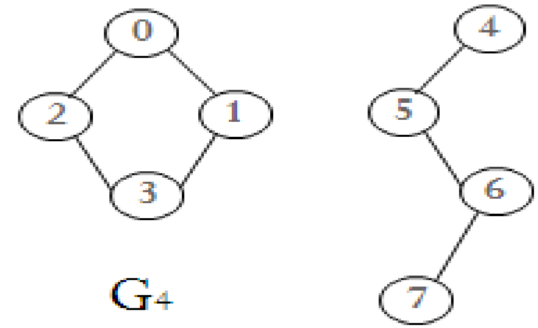
G1

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

G2



G4

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

symmetric

Merits of Adjacency Matrix

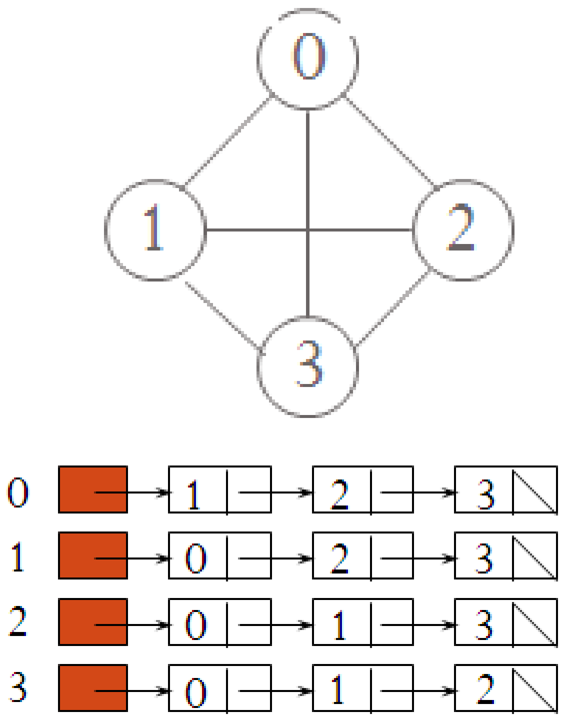
- From the adjacency matrix, to determine the connection of vertices is easy
- The degree of a vertex is $\sum_{j=0}^{n-1} adj\_mat[i][j]$
- For a digraph (= directed graph), the row sum is the out_degree, while the column sum is the in_degree

$$ind(vi) = \sum_{j=0}^{n-1} A[j,i] \quad outd(vi) = \sum_{j=0}^{n-1} A[i,j]$$

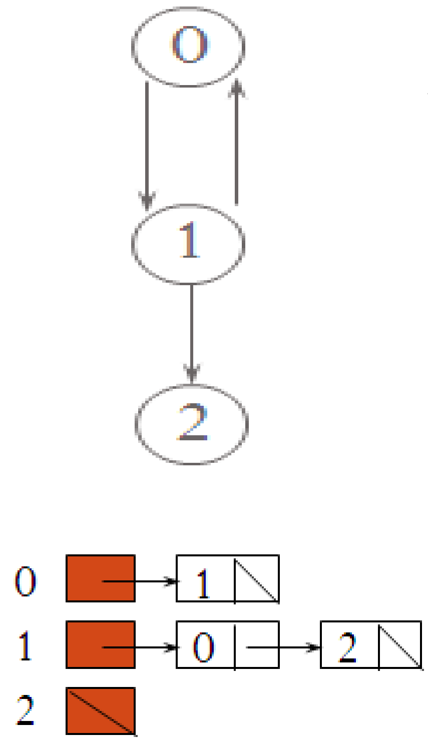
Adjacency Lists (data structures)

Each row in adjacency matrix is represented as an adjacency list.

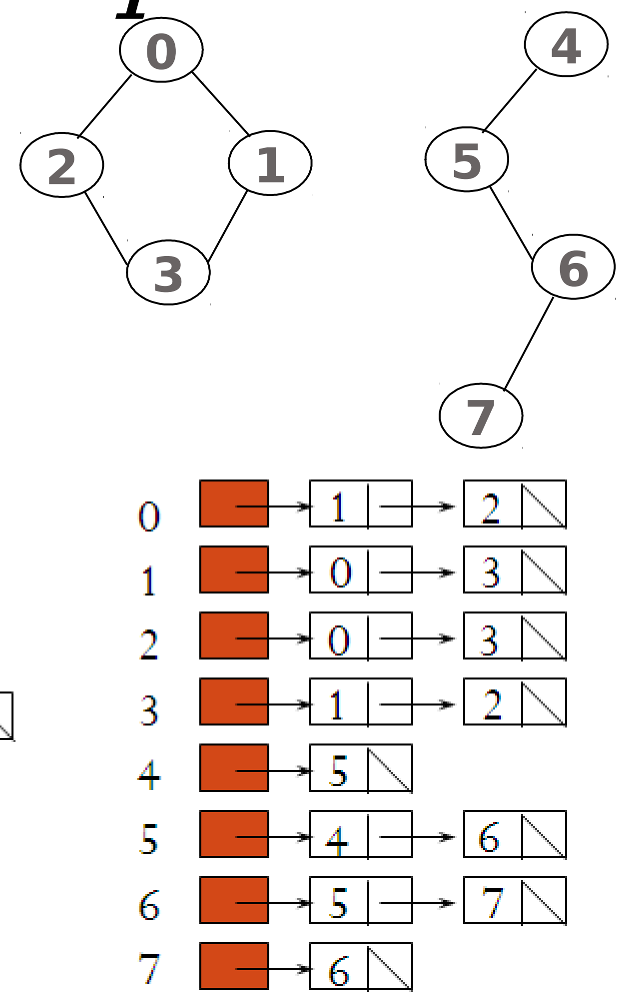
Adjacency Lists Examples



G_1



G_3



G_4

rected graph with n vertices and e edges ==> n head nodes and 2e list

Graph Traversal

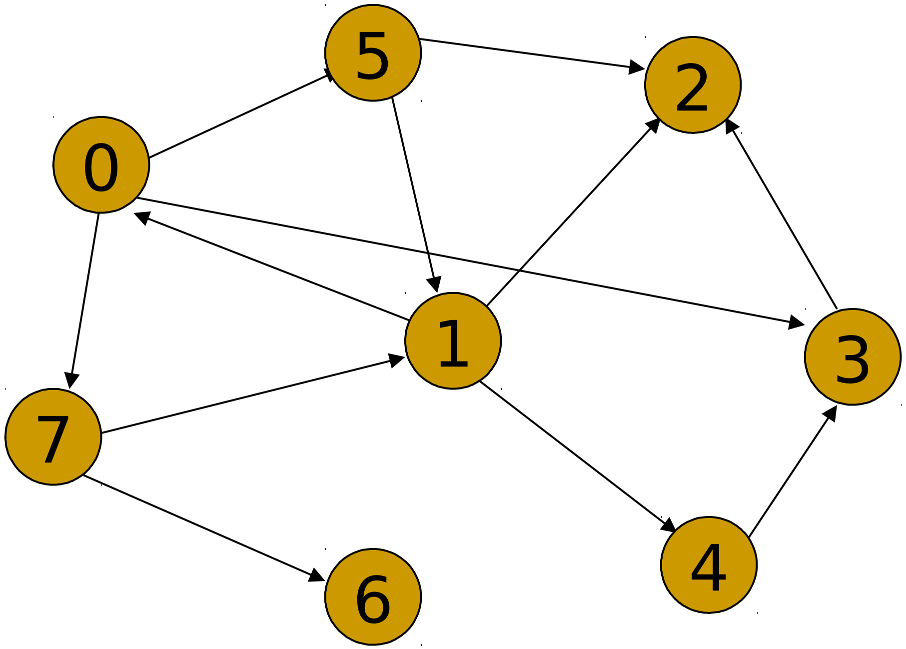
- **Problem:** Search for a certain node or traverse all nodes in the graph

- **Depth First Search**
  - Once a possible path is found, continue the search until the end of the path

- **Breadth First Search**
  - Start several paths at a time, and advance in each one step at a time

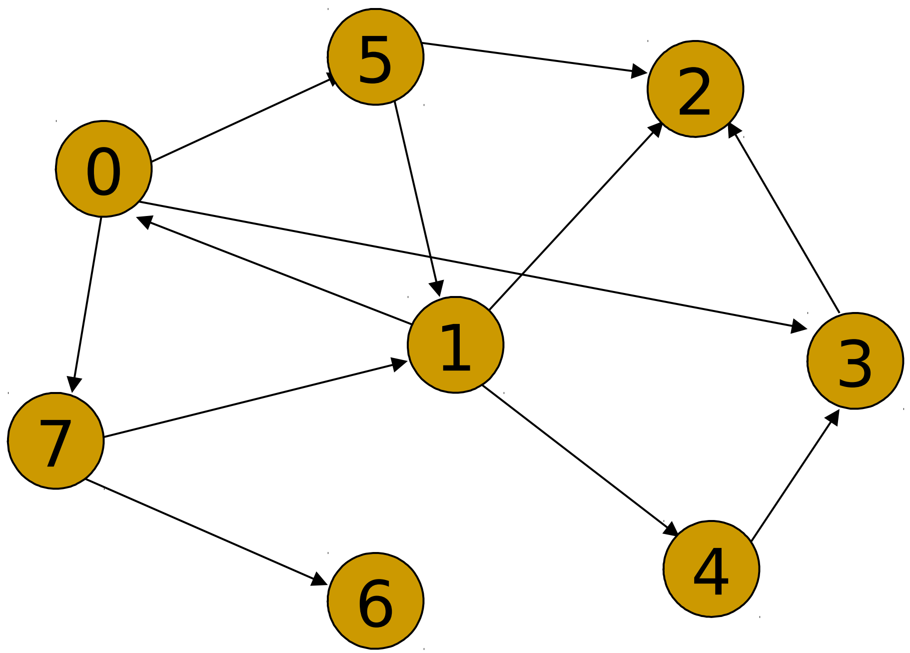
- DFS follows the following rules:
  - Select an unvisited node s, visit it, and treat as the current node
  - Find an unvisited neighbor of the current node, visit it, and make it the new current node;
  - If the current node has no unvisited neighbors, backtrack to the its parent, and make that the new current node; Repeat the above two steps until no more nodes can be visited.
  - If there are still unvisited nodes, repeat from step 1.

Example



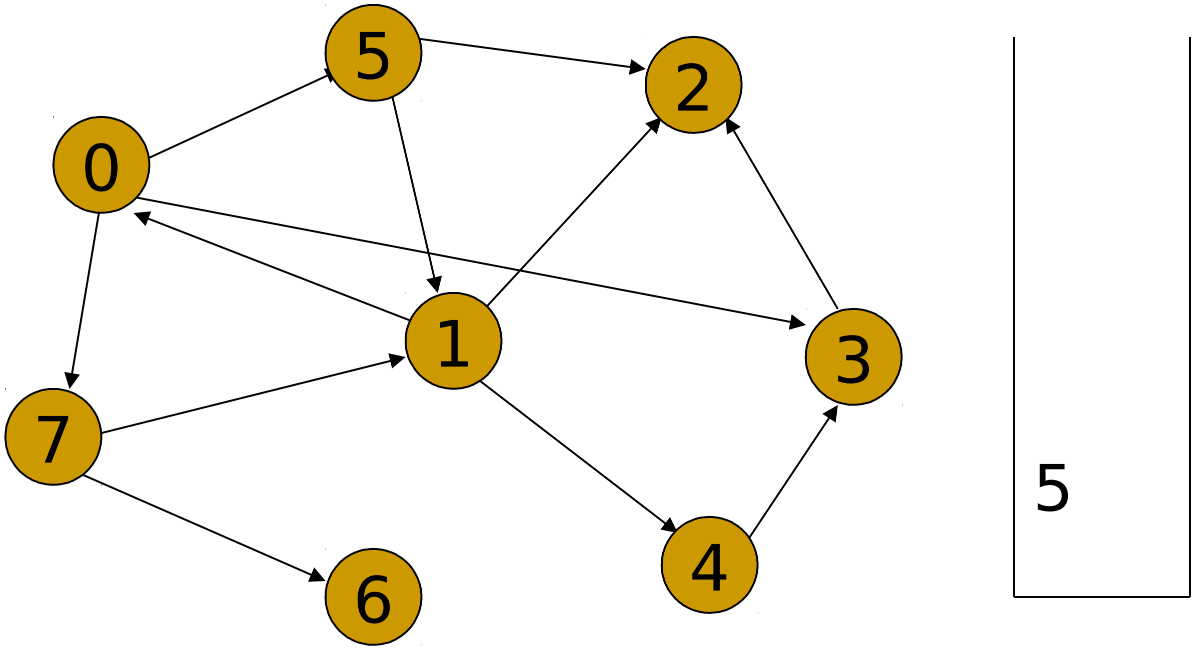
Policy: Visit adjacent nodes in increasing index order

Preorder DFS: Start with Node 5



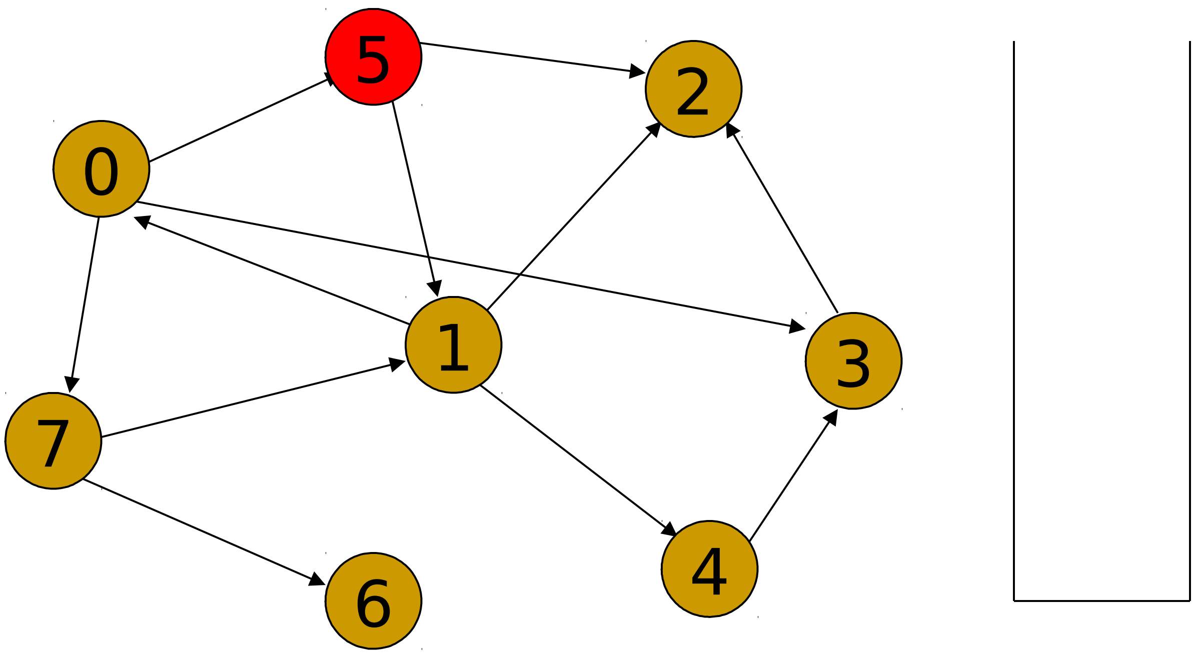
5 1 0 3 2 7 6
4

Preorder DFS: Start with Node 5



Push 5

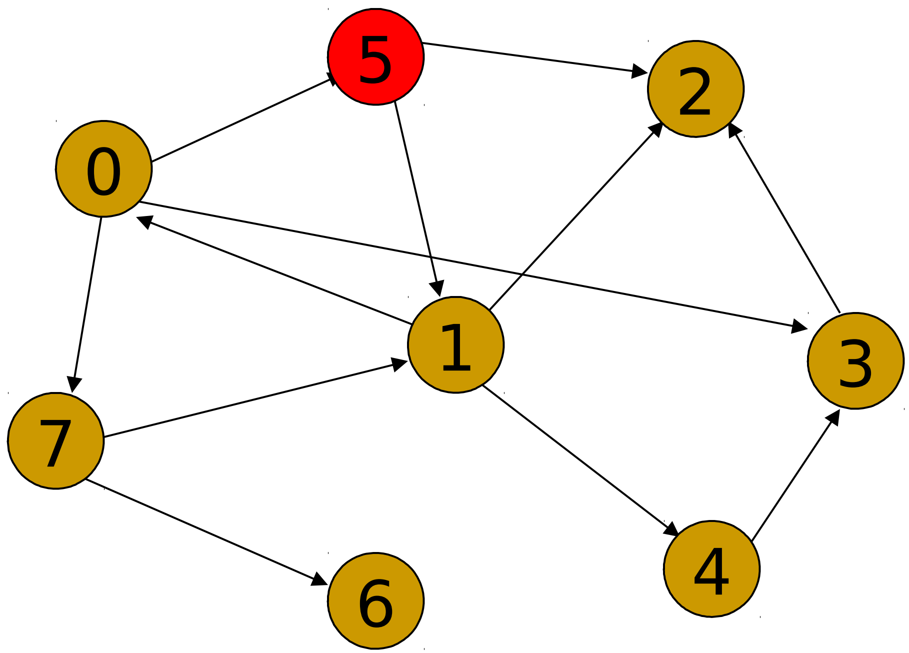
Preorder DFS: Start with Node 5

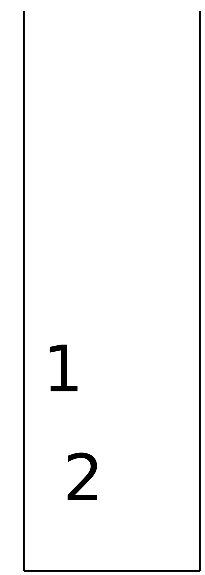


Pop/Visit/Mark 5

5

Preorder DFS: Start with Node 5

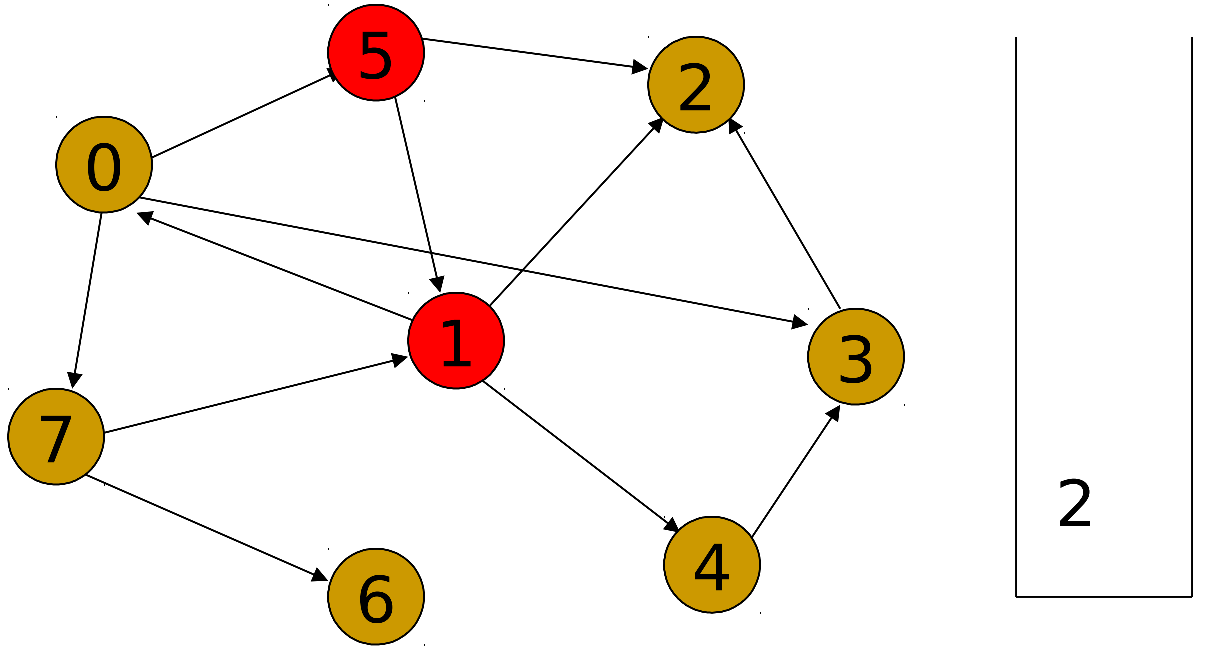




Push 2, Push 1

5

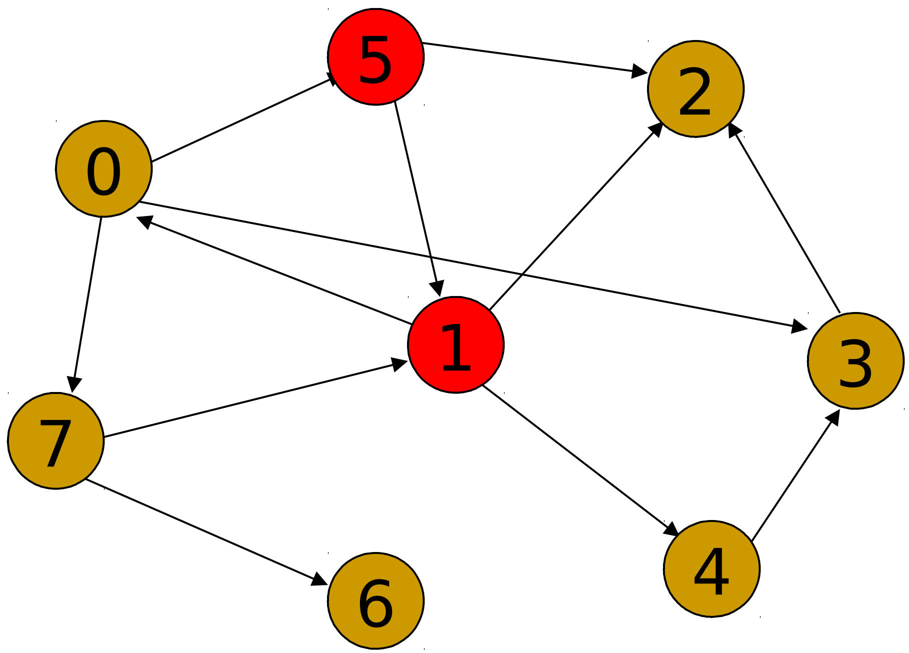
Preorder DFS: Start with Node 5

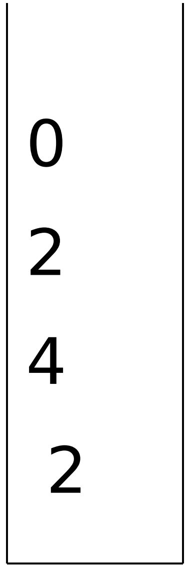


Pop/Visit/Mark 1

5 1

Preorder DFS: Start with Node 5

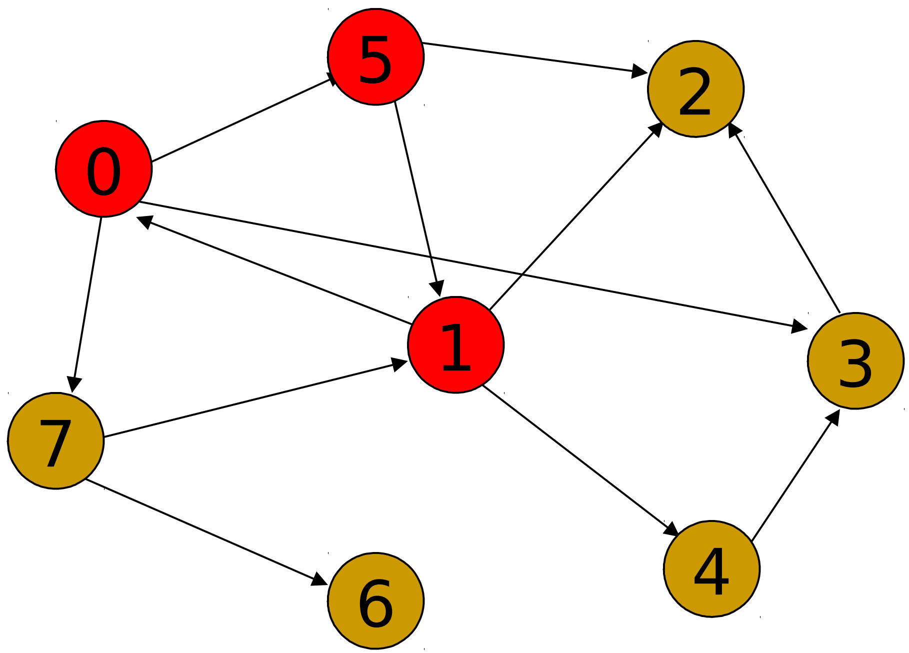




Push 4, Push 2, Push 0

5 1

Preorder DFS: Start with Node 5



2

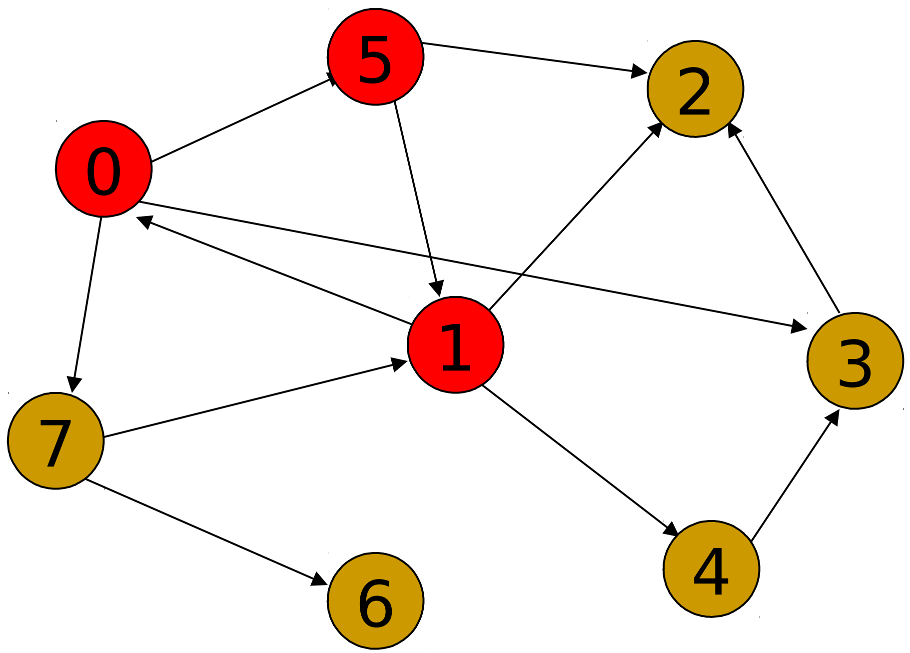
4

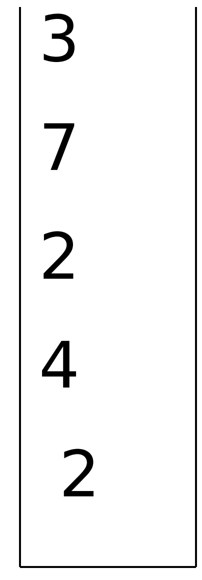
2

Pop/Visit/Mark 0

5 1 0

Preorder DFS: Start with Node 5

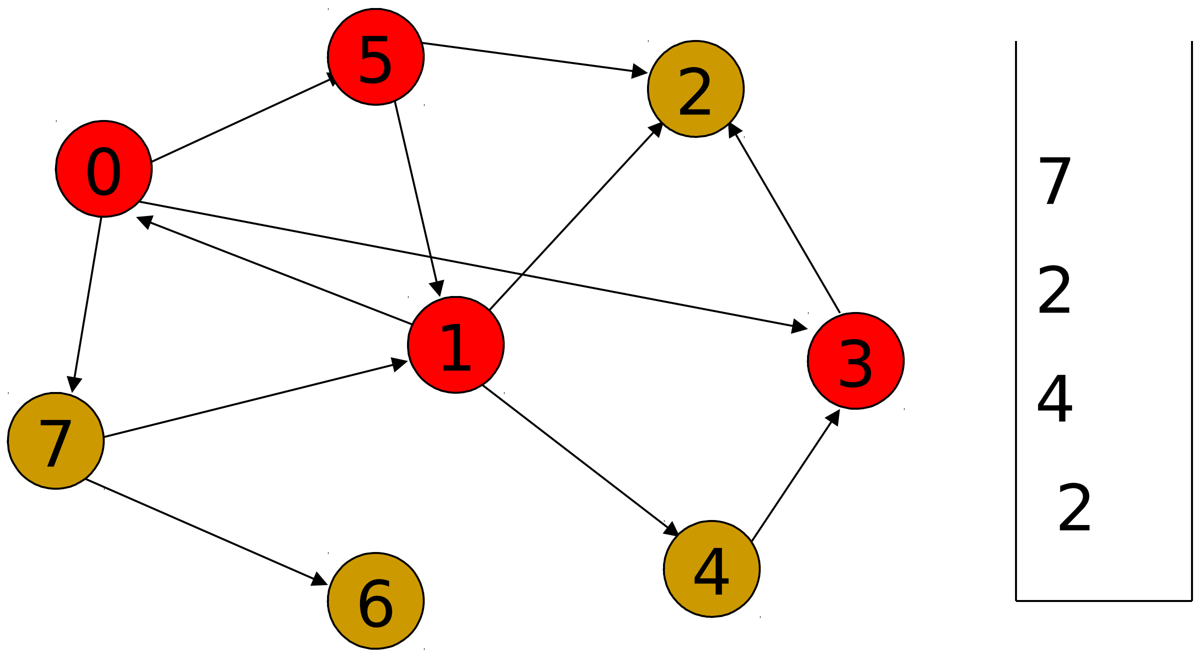




Push 7, Push 3

5 1 0

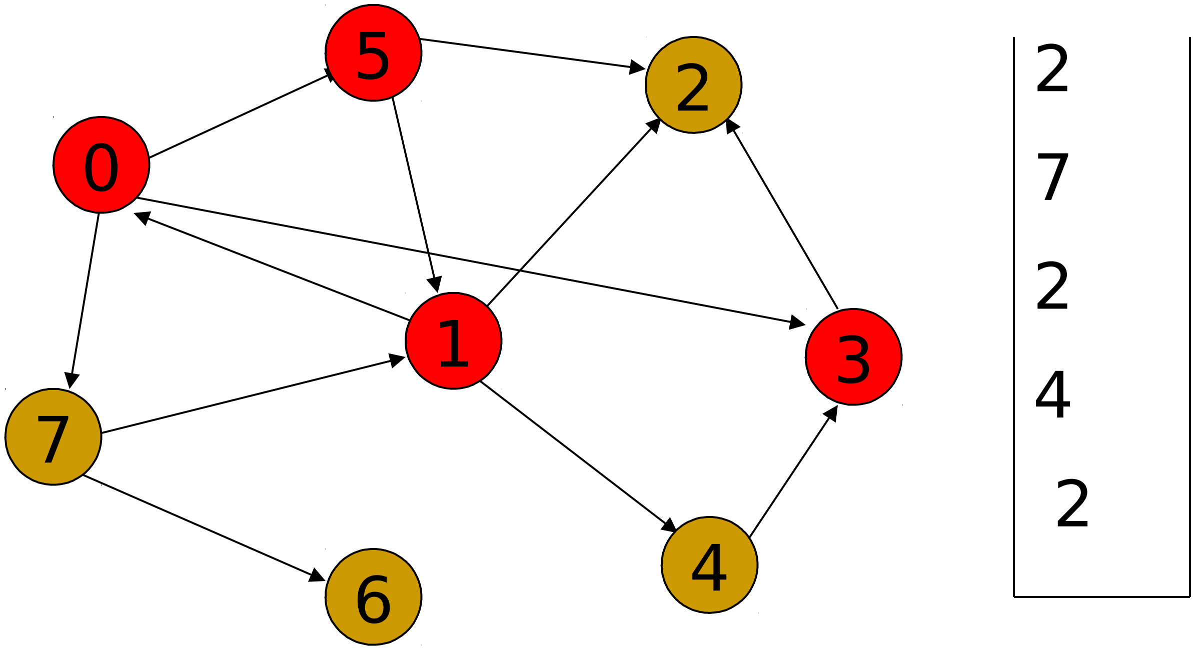
Preorder DFS: Start with Node 5



Pop/Visit/Mark 3

5 1 0 3

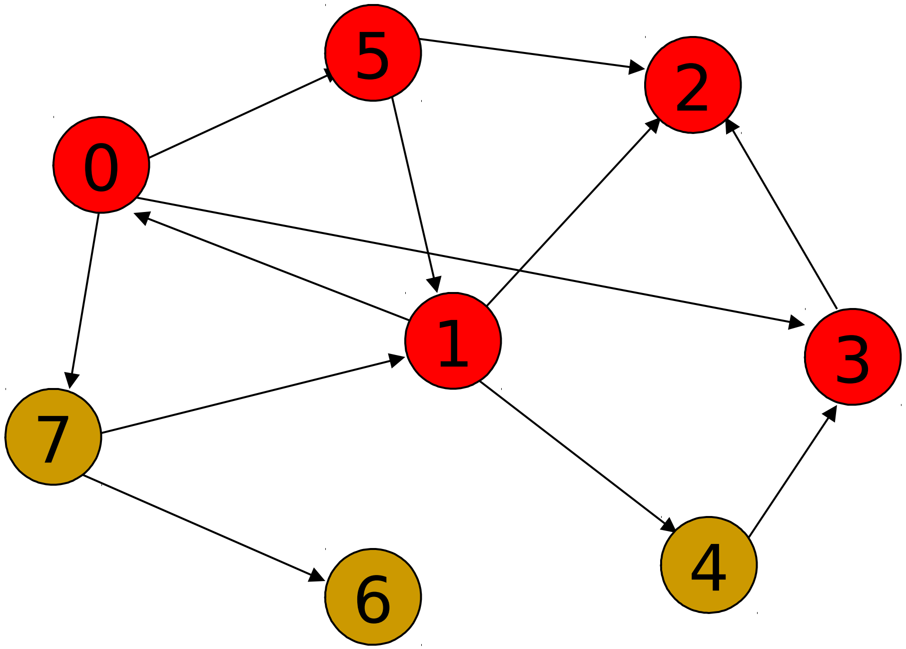
Preorder DFS: Start with Node 5

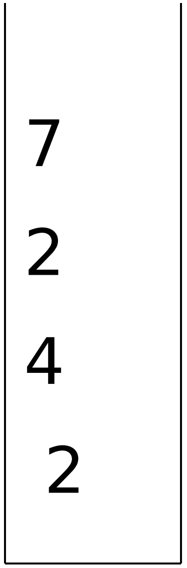


Push 2

5 1 0 3

Preorder DFS: Start with Node 5

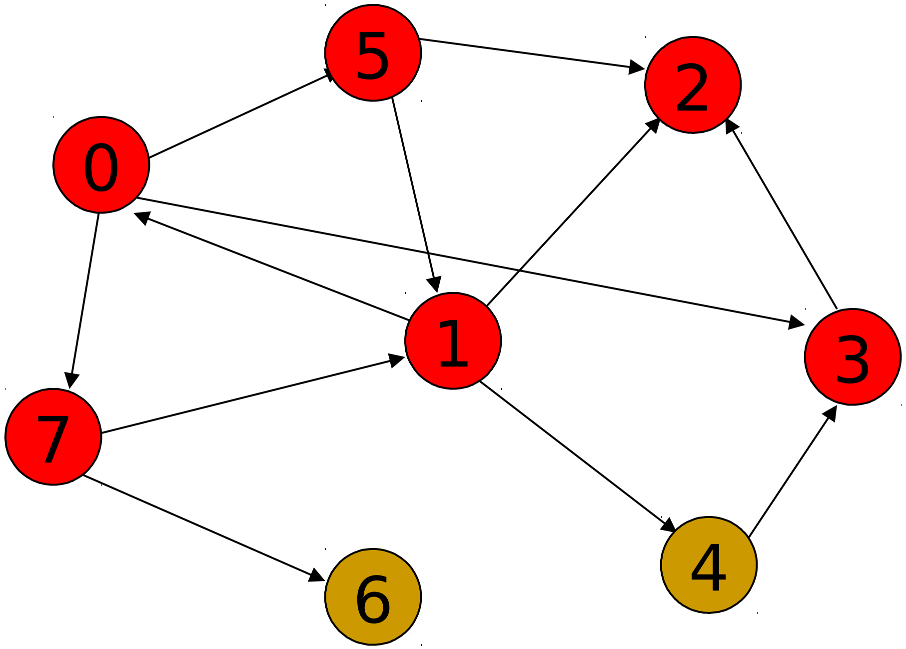


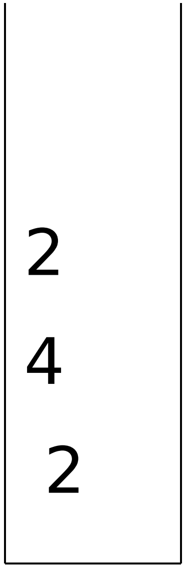


Pop/Mark/Visit 2

5 1 0 3 2

Preorder DFS: Start with Node 5

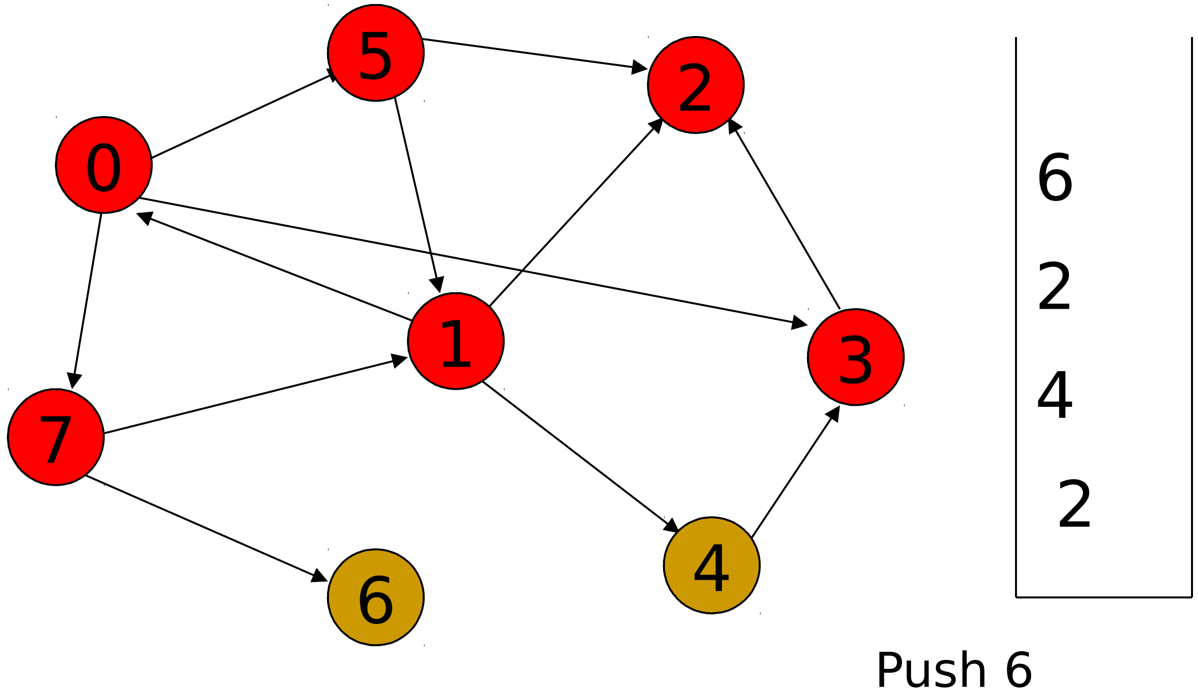




Pop/Mark/Visit 7

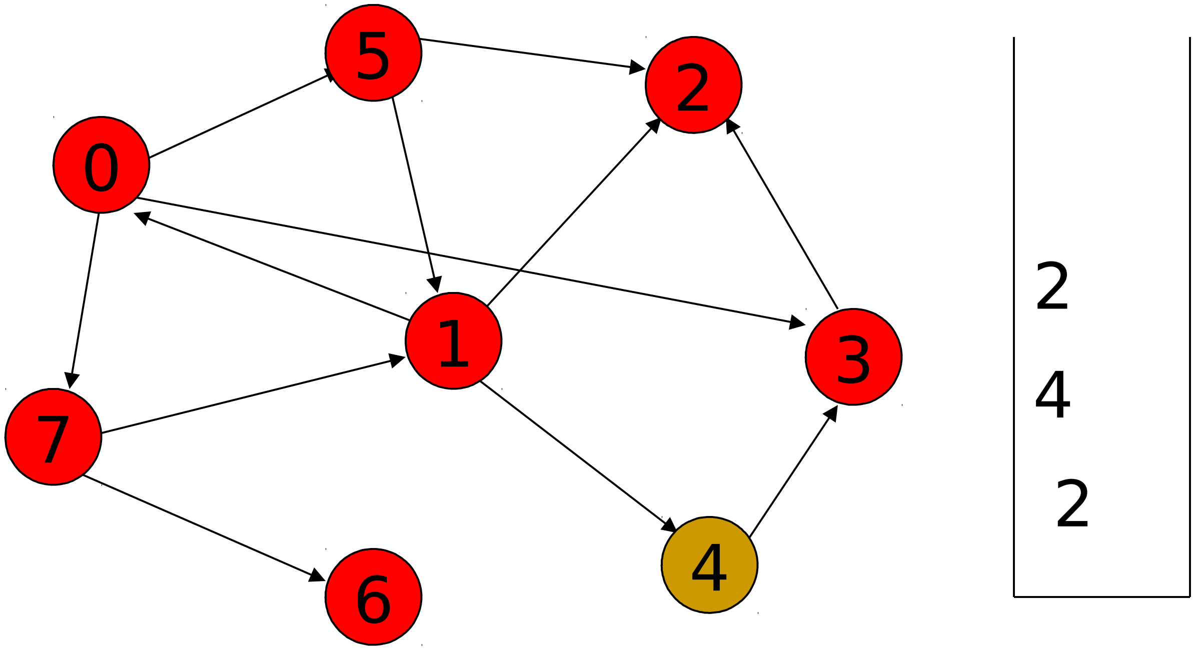
5 1 0 3 2 7

Preorder DFS: Start with Node 5



5 1 0 3 2 7

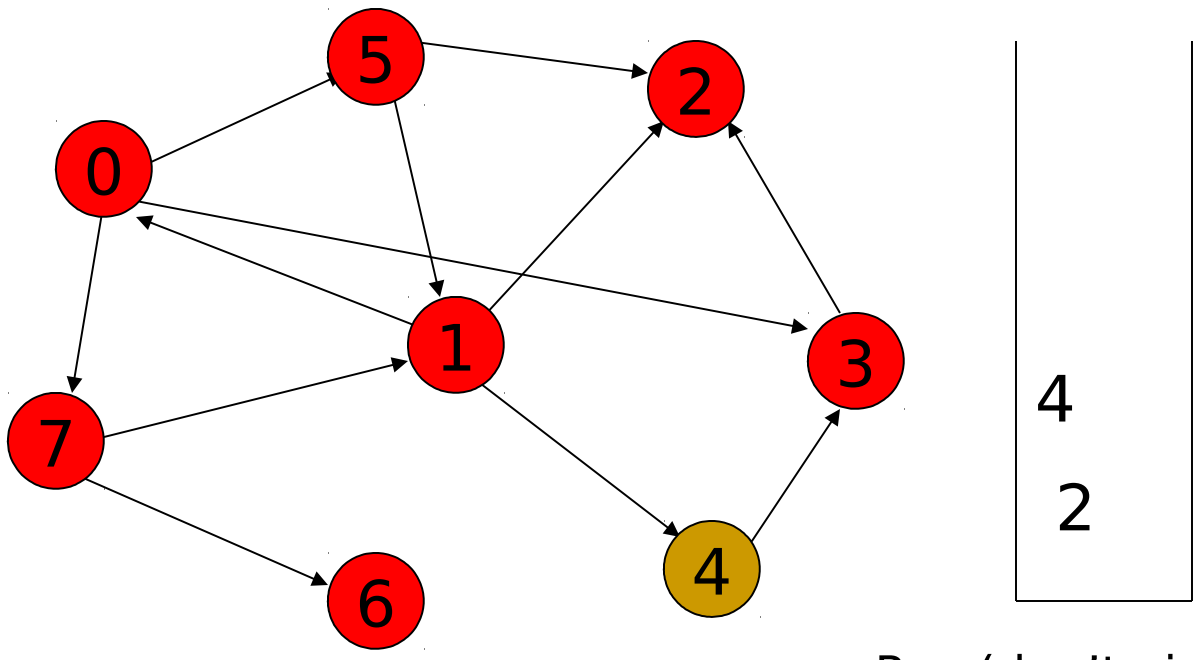
Preorder DFS: Start with Node 5



Pop/Mark/Visit 6

5 1 0 3 2 7 6

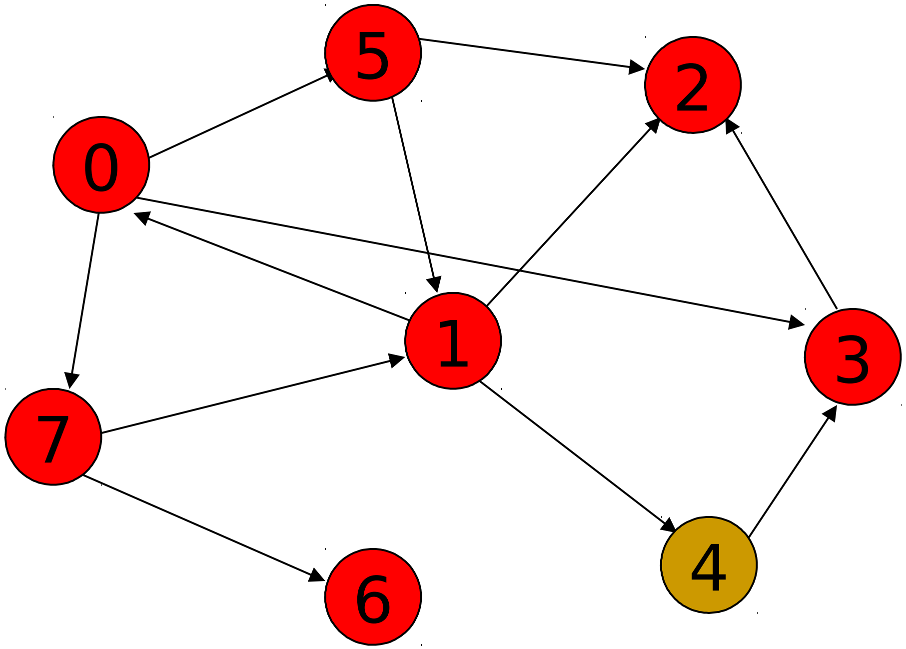
Preorder DFS: Start with Node 5

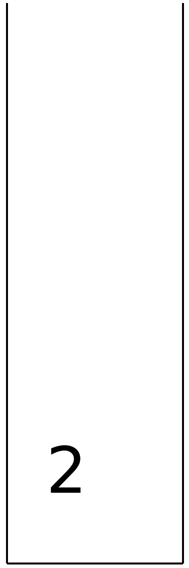


Pop (don't visit) 2

5 1 0 3 2 7 6

Preorder DFS: Start with Node 5

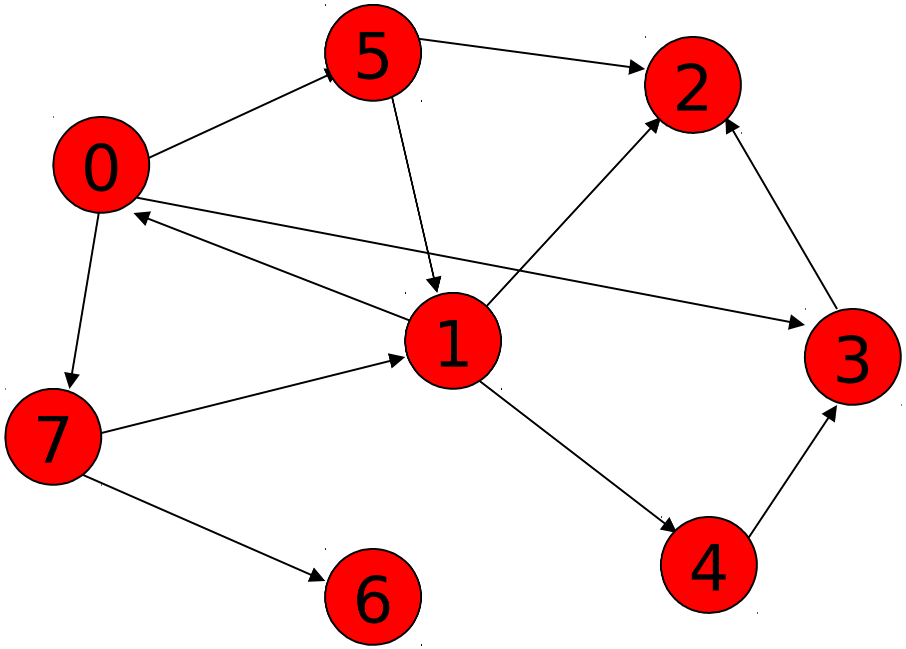




Pop/Mark/Visit 4

5 1 0 3 2 7 6
4

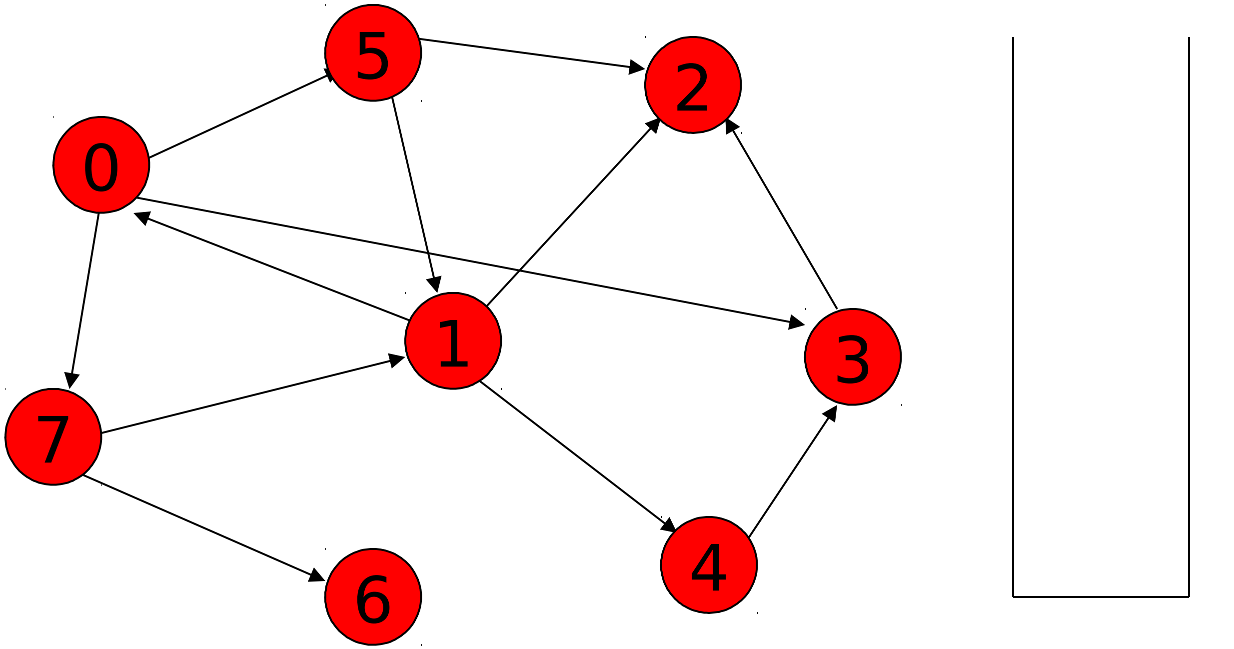
Preorder DFS: Start with Node 5



Pop (don't visit) 2

5 1 0 3 2 7 6
4

Preorder DFS: Start with Node 5



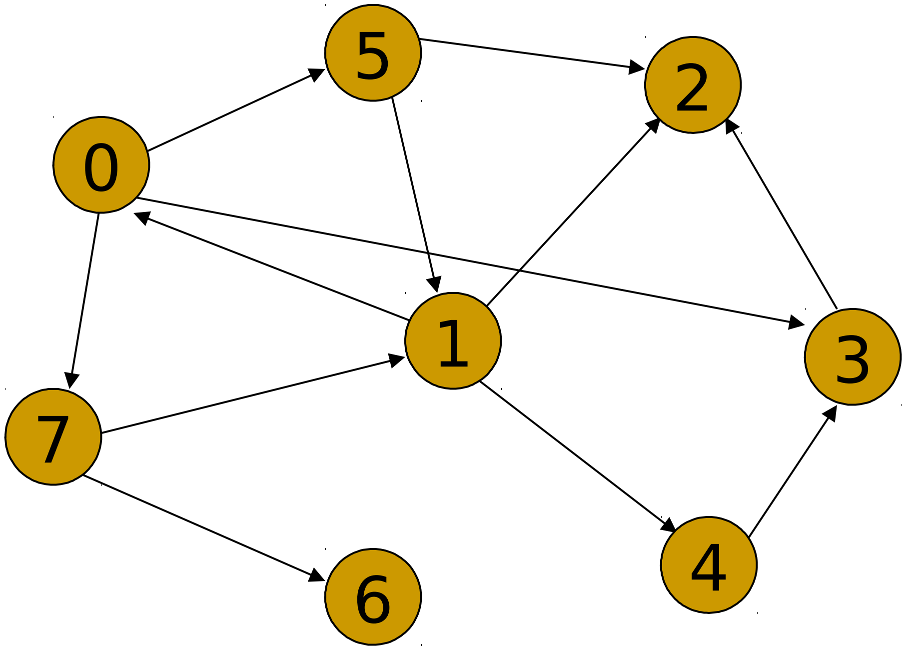
Done

5 1 0 3 2 7 6
4

Breadth First Search

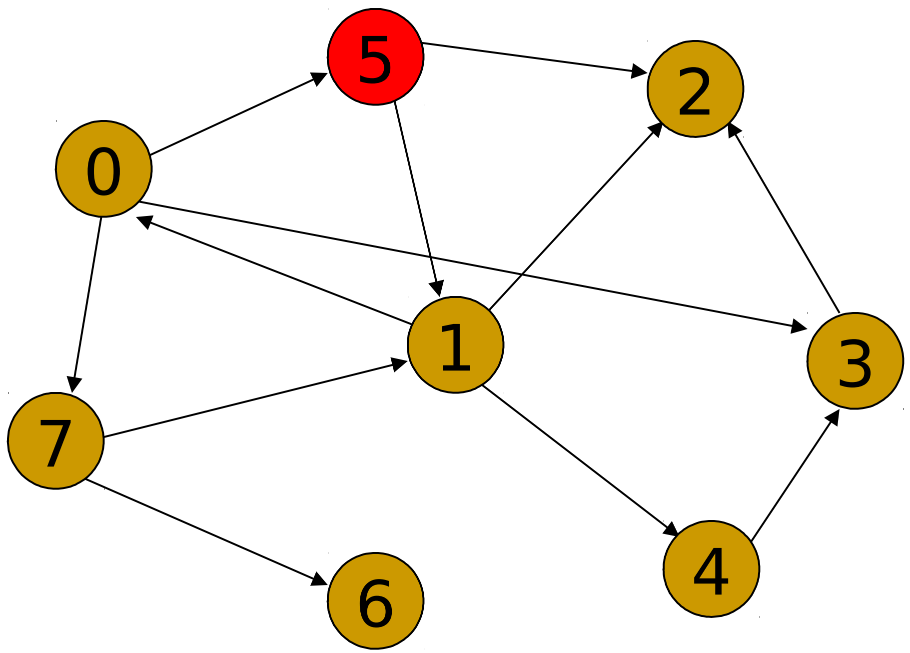
- BFS follows the following rules:
  - Select an unvisited node s, visit it, have it be the root in a BFS tree being formed. Its level is called the current level.
  - From each node x in the current level, in the order in which the level nodes were visited, visit all the unvisited neighbors of x. The newly visited nodes from this level form a new level that becomes the next current level.
  - Repeat the previous step until no more nodes can be visited.
  - If there are still unvisited nodes, repeat from Step 1.

BFS: Start with Node 5



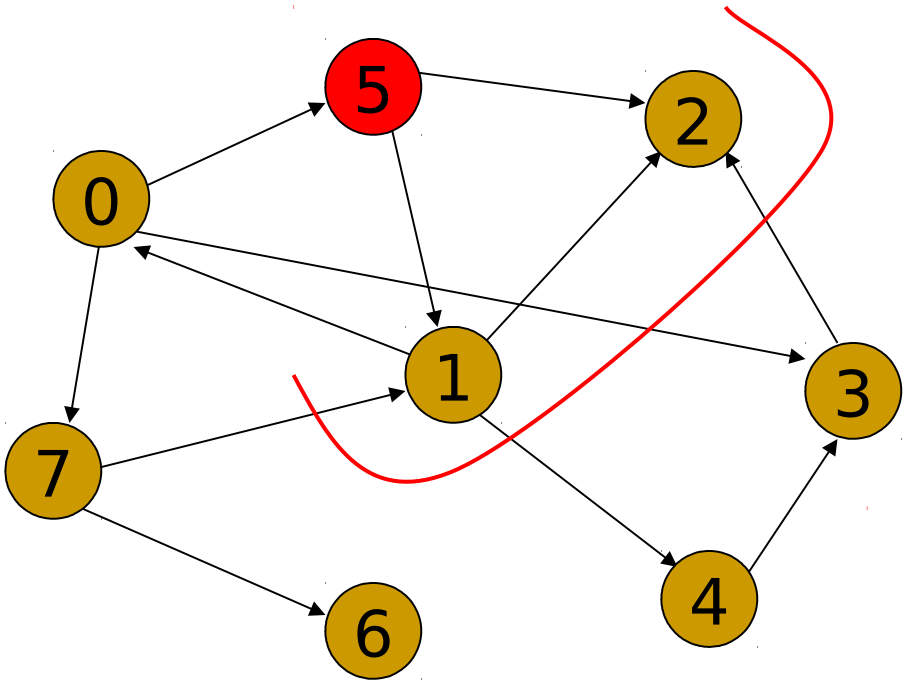
5 1 2 0 4 3 7
6

BFS: Start with Node 5



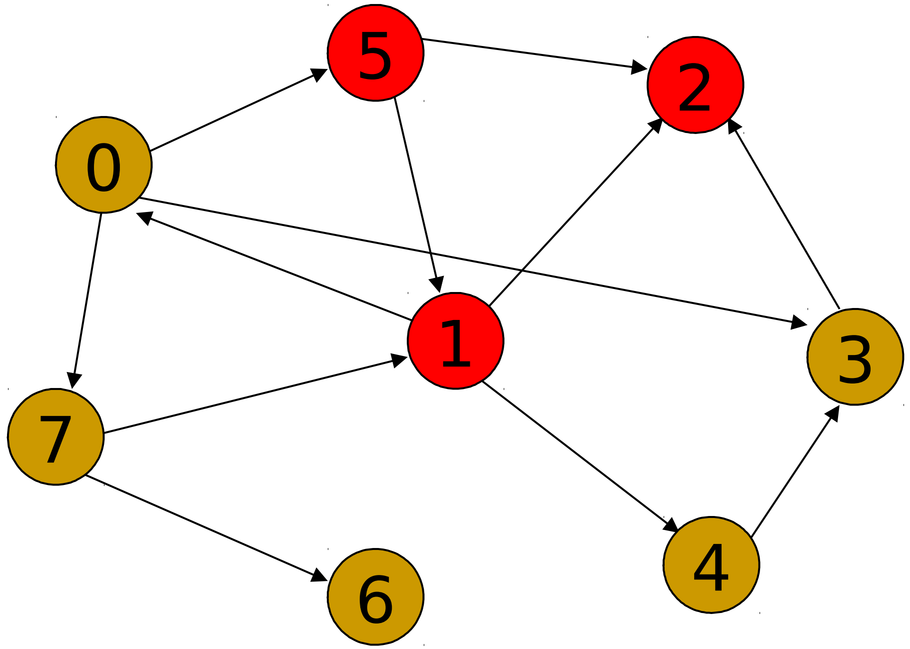
5

BFS: Node one-away



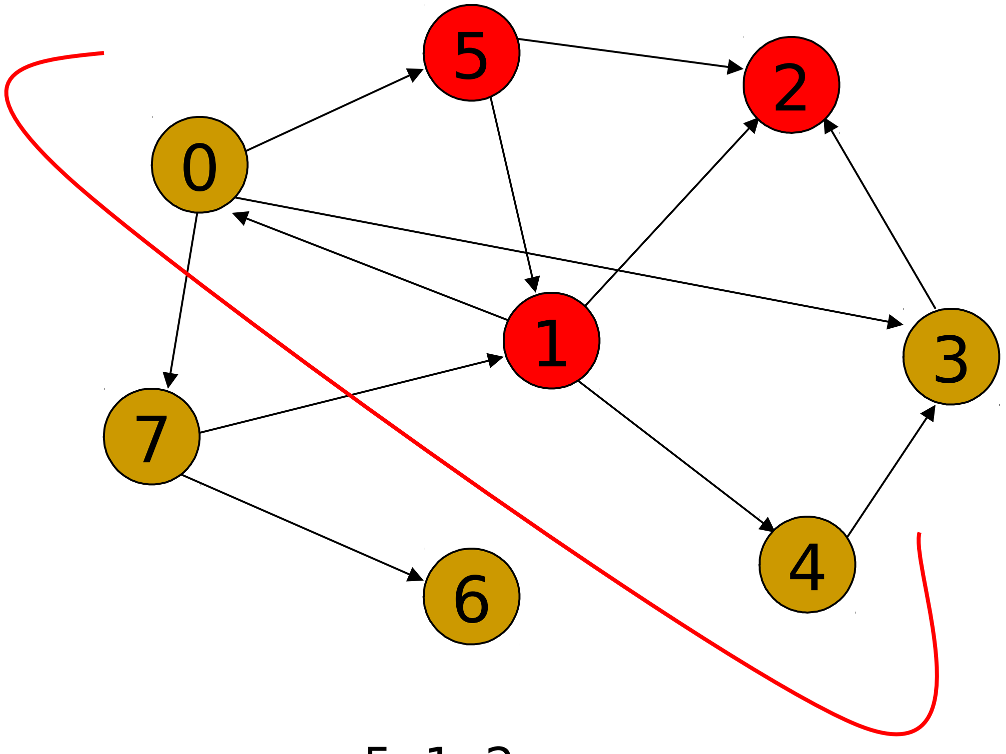
5

BFS: Visit 1 and 2



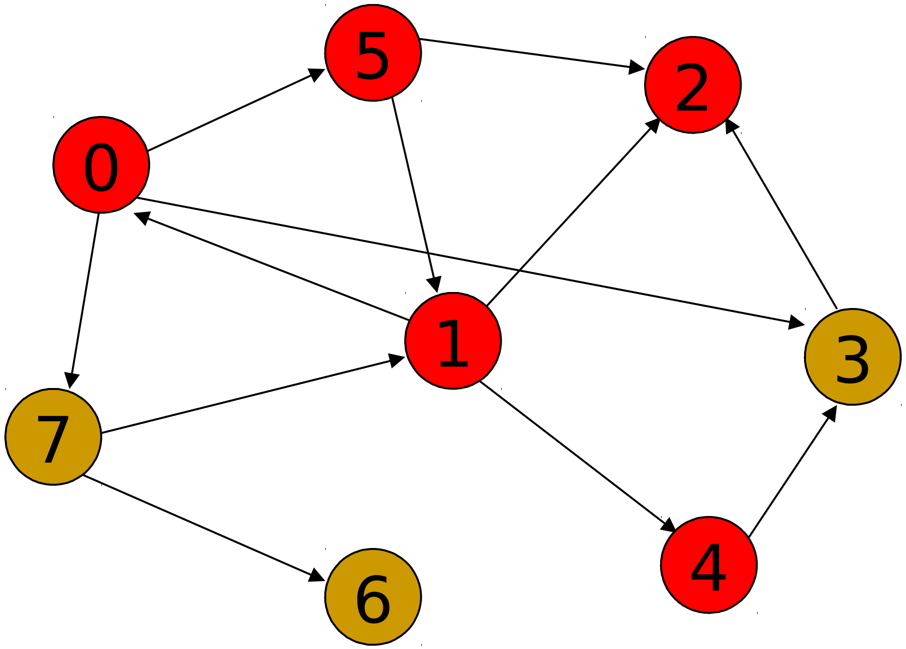
5 1 2

BFS: Nodes two-away



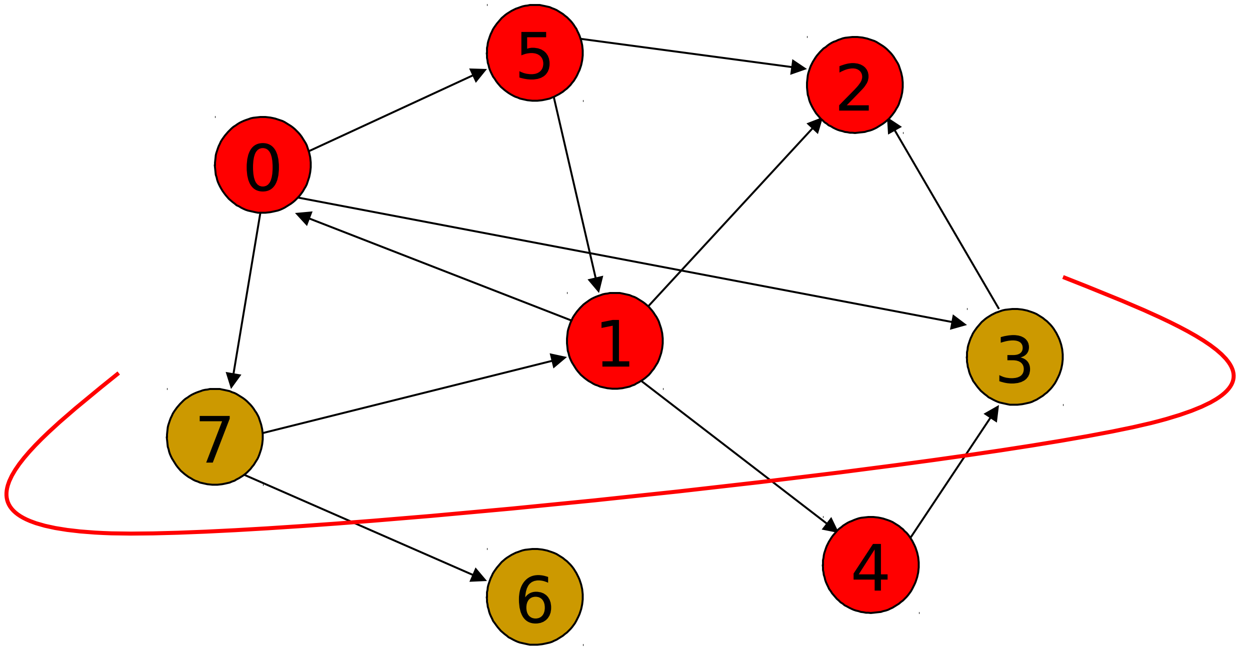
5 1 2

BFS: Visit 0 and 4



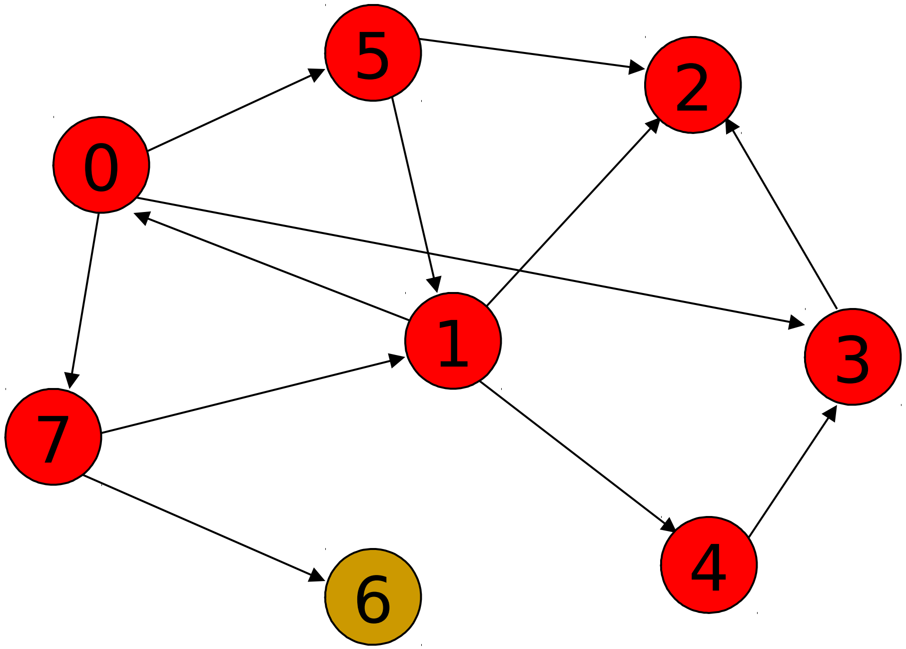
5 1 2 0 4

BFS: Nodes three-away



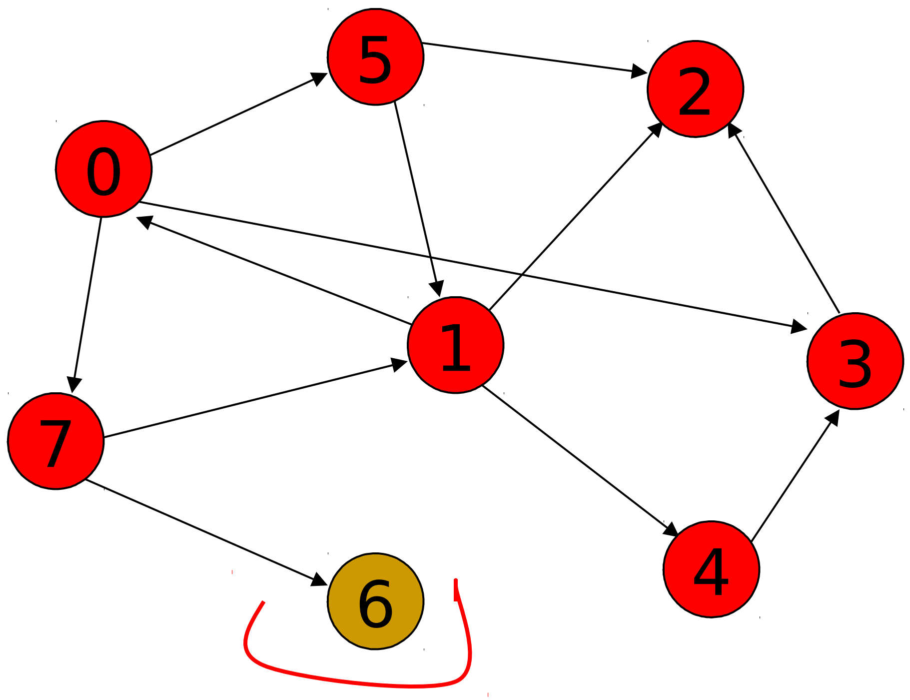
5 1 2 0 4

BFS: Visit nodes 3 and 7



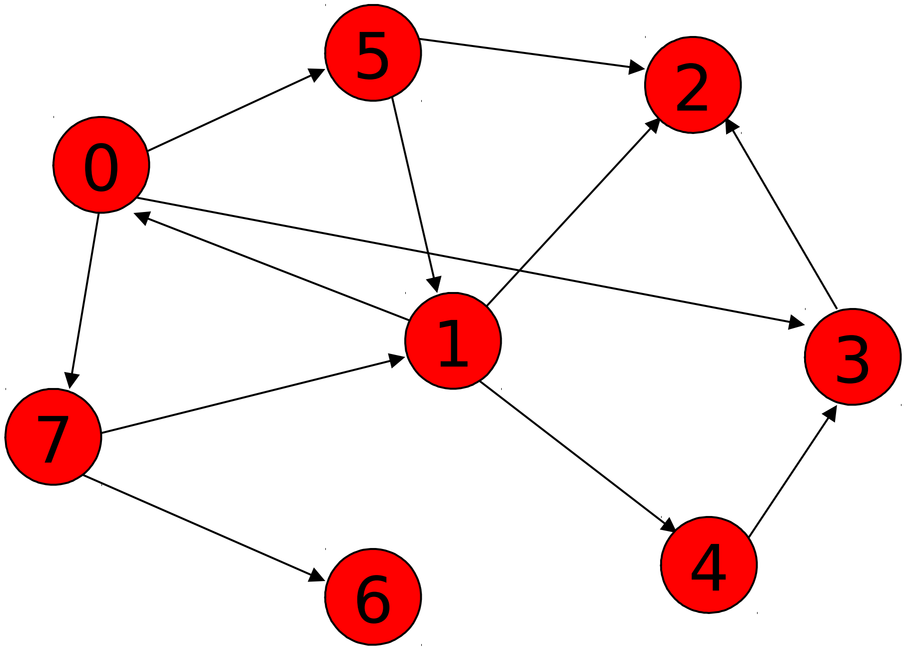
5 1 2 0 4 3 7

BFS: Node four-away



5 1 2 0 4 3 7

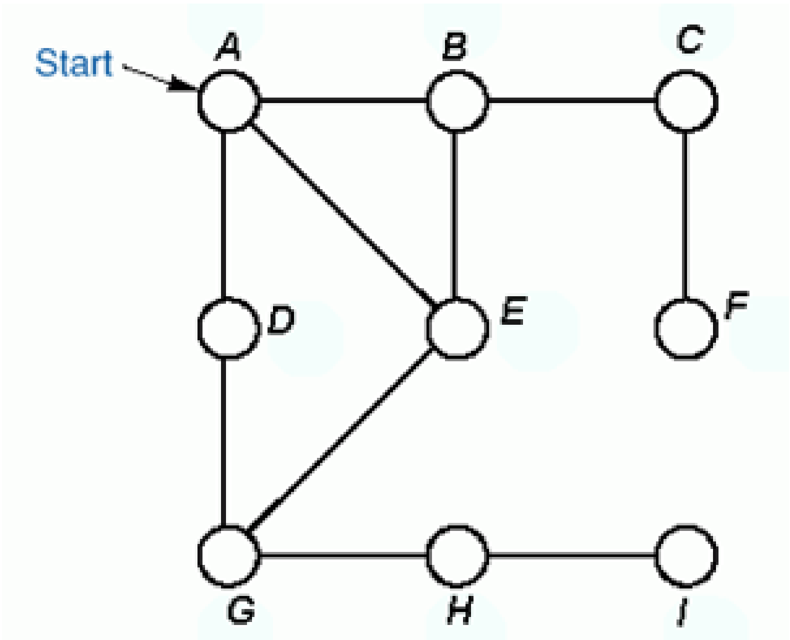
BFS: Visit 6



5 1 2 0 4 3 7
6

Breadth-First Traversal Example

Consider the following graph:

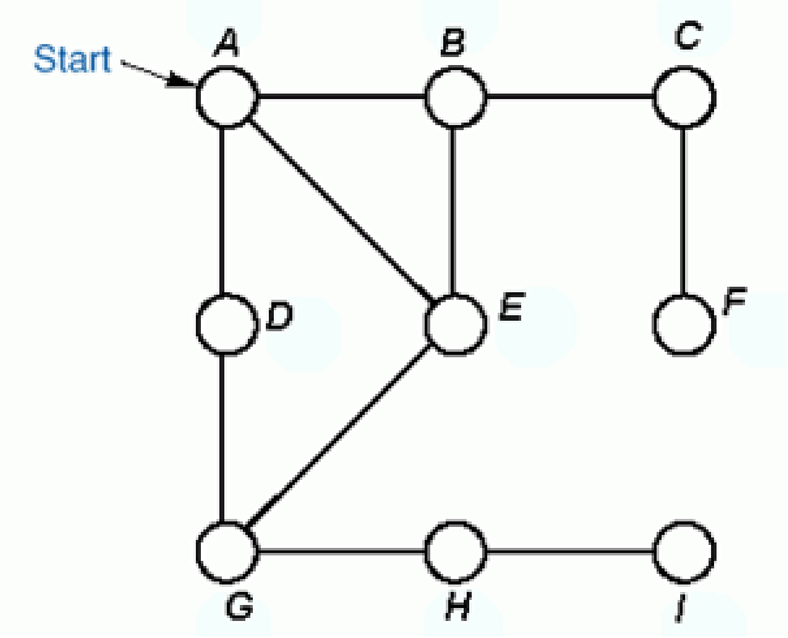


Order of Traversal

1	2	3	4	5	6	7	8	9
A	B	D	E	C	G	F	H	I

Depth-First Traversal Example

Consider the following graph:

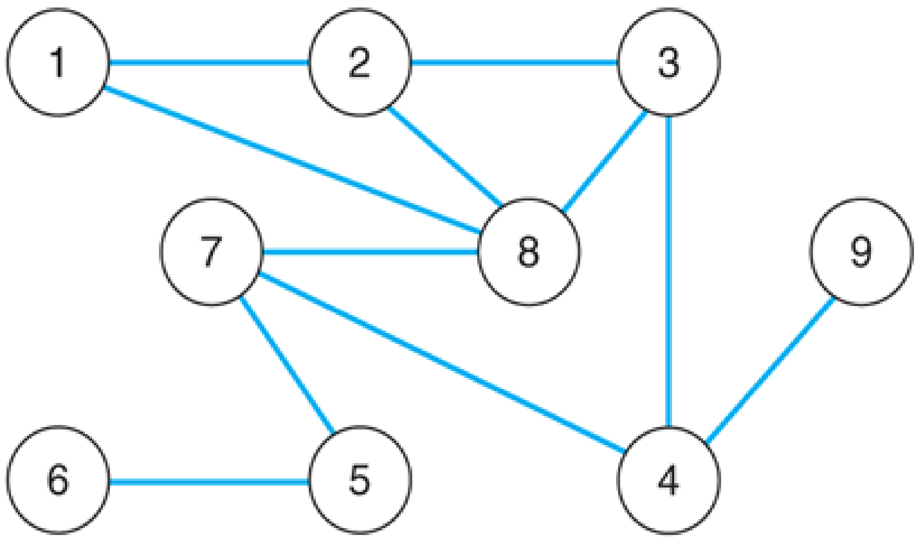


Order of Traversal

1	2	3	4	5	6	7	8	9
A	B	C	F	E	G	D	H	I

Depth-First Traversal Example

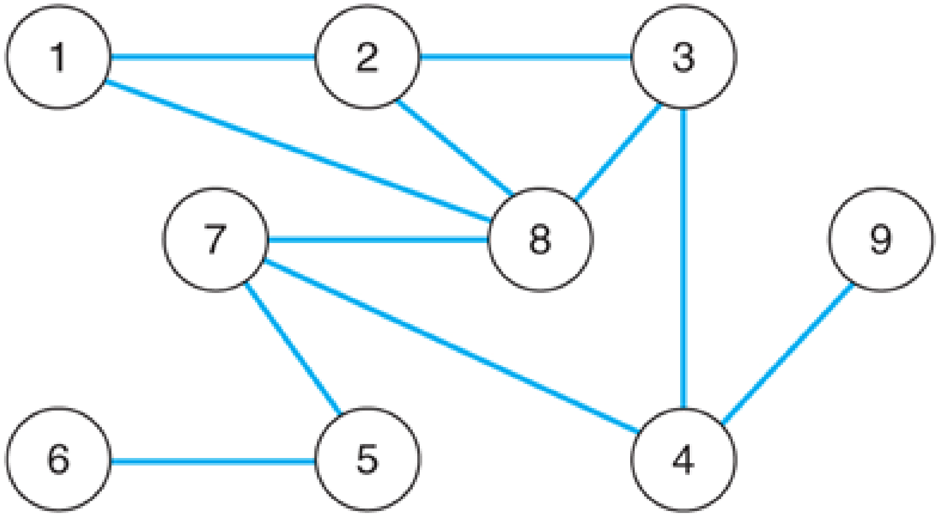
- Consider the following graph: **Start nod**



- The order of the depth-first traversal of this graph starting at node 1 would be: 1, 2, 3, 4, 7, 5, 6, 8, 9

Breadth-First Traversal Example

- Consider the following graph: **Start node is 1**



- The order of the breadth-first traversal of this graph starting at node 1 would be: 1, 2, 8, 3, 7, 4, 5, 9, 6

END OF THE CHAPTER