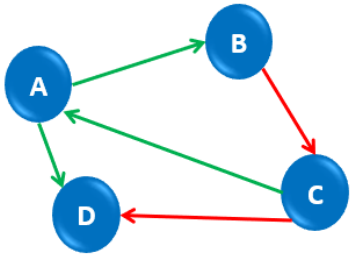
Graphs

Definitions

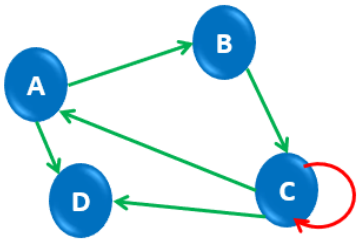
- A graph is a non-linear data structure that consists of a set of vertices, V, and a set of edges, E. Each edge is a pair (v, w), where v , w Є V
- If the pair (v, w) is ordered, then the graph is directed, called directed graph/ digraph
- Vertex w is adjacent to vertex v if v and w are connected by an edge.

- A path in a graph is a sequence of vertices w_1, w_2, w_3, . . . , w_n such that (w_i, w_{i+i}) Є E for 1 <= i < n.
- The length of a path is the number of edges on the path



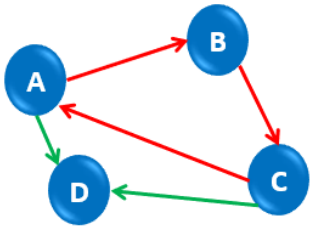
BCD is a path of length 2

- Loop is an edge (v, v) that connects a vertex v to itself.

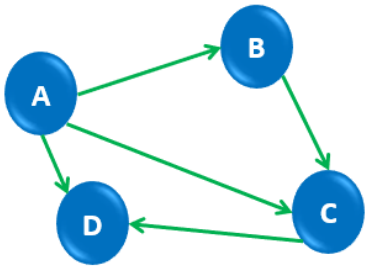


- A simple path is a path such that all vertices are distinct, except that the first and last could be the same.

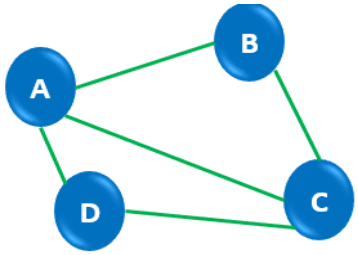
- A cycle in a directed graph is a path (w_1, w_2, w_3,…, w_n) of length at least 1 such that $w_1 = w_n$



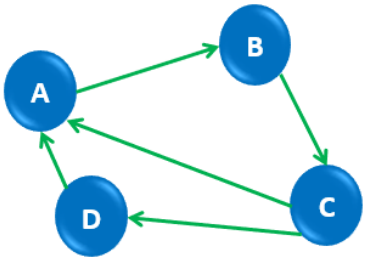
- A directed graph is acyclic if it has no cycles. A directed acyclic graph is sometimes referred to by its abbreviation, DAG



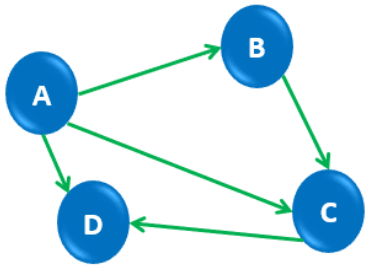
- An undirected graph is connected if there is a path from every vertex to every other vertex.



- A directed graph which is connected is called strongly connected.



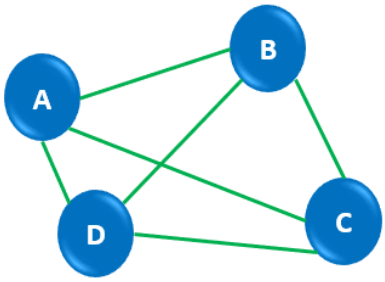
- If a directed graph is not strongly connected, but the underlying graph (without direction to the arcs) is connected, then the graph is said to be weakly connected. For example



In this graph there is no path from vertex D to A, B, and C.

But without direction to the arcs, there is a path from D to remaining vertices. So this is a weakly connected graph.

- A complete graph is a graph in which there is an edge between every pair of vertices

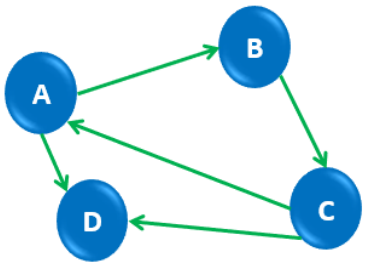


- Outgoing Edge - A directed edge is said to be outgoing edge on its source vertex.
- Incoming Edge - A directed edge is said to be incoming edge on its destination vertex.
- **Degree** - Total number of edges connected to a vertex is said to be degree of that vertex.
    - **Indegree** - Total number of incoming edges connected to a vertex is said to be indegree of that vertex.
    - **Outdegree** - Total number of outgoing edges connected to a vertex is said to be outdegree of that vertex.

Representation of Graphs

1. Adjacency Matrix

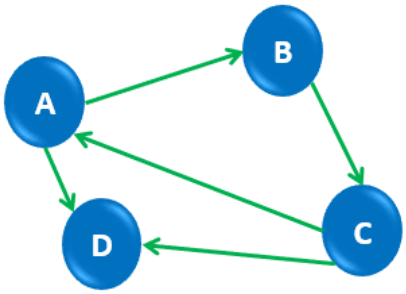
- One simple way to represent a graph is to use a two-dimensional array. This is known as an adjacency matrix representation.
- For each edge (u, v), we set a[u][v] = 1; otherwise the entry in the array is 0.
- If the edge has a weight associated with it, then we can set a[u][v] equal to the weight and use either a very large or a very small weight as a sentinel to indicate nonexistent edges.

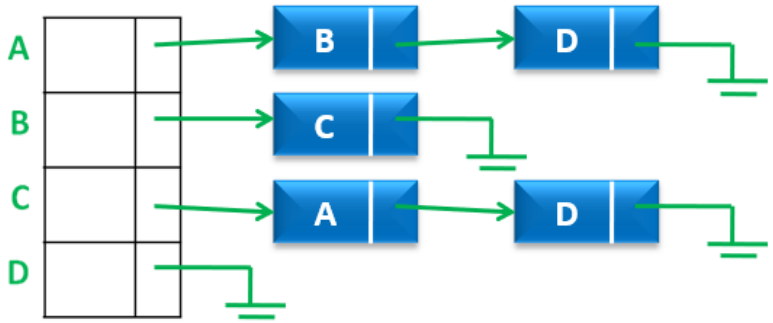


	A	B	C	D
A	0	1	0	1
B	0	0	1	0
C	1	0	0	1
D	0	0	0	0

2. Adjacency List

- For each vertex, we keep a list of all adjacent vertices.
- If the edges have weights, then this additional information is also stored in the cells.



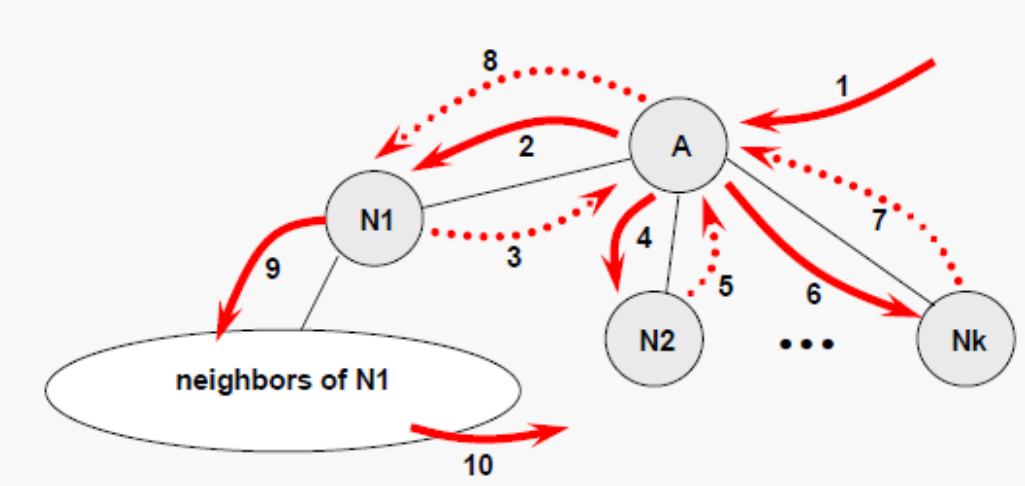


Graph Traversal

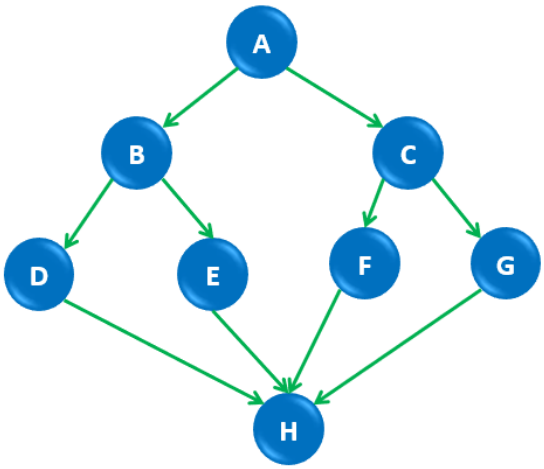
- Graph traversal is the process of visiting all the nodes in a graph in a particular manner
- The order in which the vertices are visited may be important, and may depend upon the particular algorithm.
- The two common traversals are:
  - Depth-first Search
  - Breadth-first Search

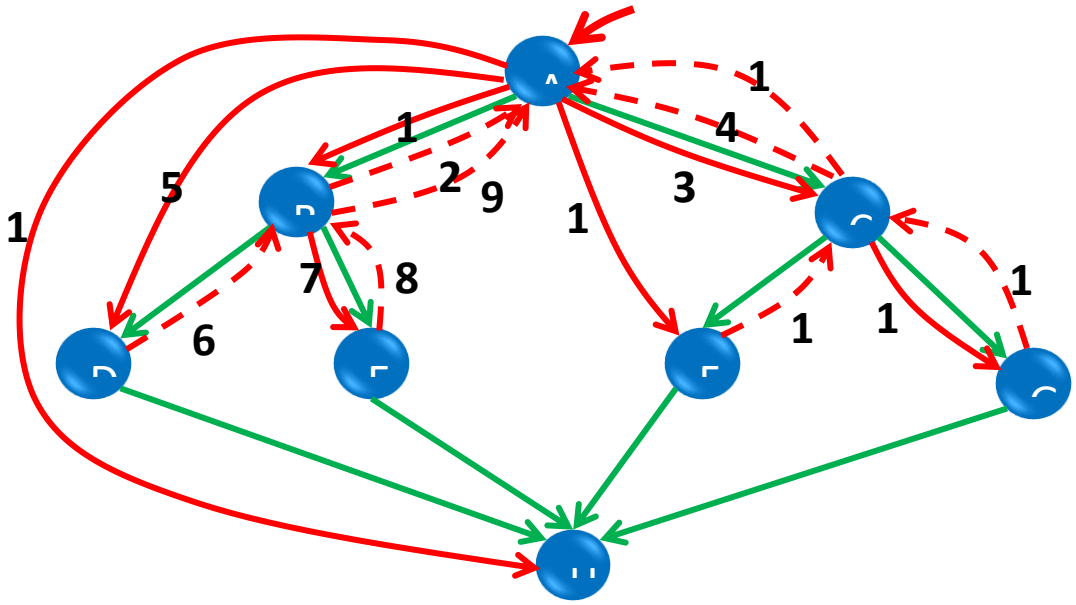
Breadth-first Search: The BFS begins at a node and inspects all the neighboring nodes. Then for each of those neighbor nodes in turn, it inspects their neighbor nodes which were unvisited, and so on. BFS uses a queue to hold the nodes that are unexplored. The algorithm follows the strategy given below:

- Assume a particular node has been designated as the starting point.
- Let A be the last node visited and suppose A has neighbors N_1, N_2, …, N_k.
- A breadth-first traversal will:
  - Visit N_1, then N_2, and so forth through N_k, then
  - Proceed to traverse all the unvisited immediate neighbors of N_1, then
  - Traverse the immediate neighbors of N_2, … N_k in similar fashion.



Following figure shows the order in which nodes are visited, while using BFS. Consider the following graph.





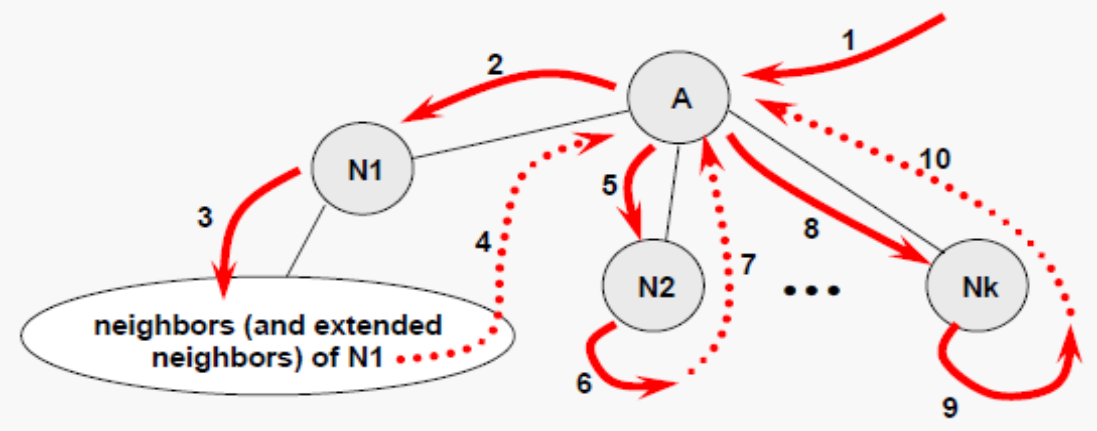
Algorithm for BFS

```
Algorithm BFS(v)
{
u:=v;
Visited[v]:=1;
while(true){
      for all vertices w adjacent from u do
      {
            if ( visited[w]=0) then
            { Add w to q;
               Visited[w]:=1;
            }
      }
      if q is empty then return;
      Delete the next element, u, from q;
      }
}
```

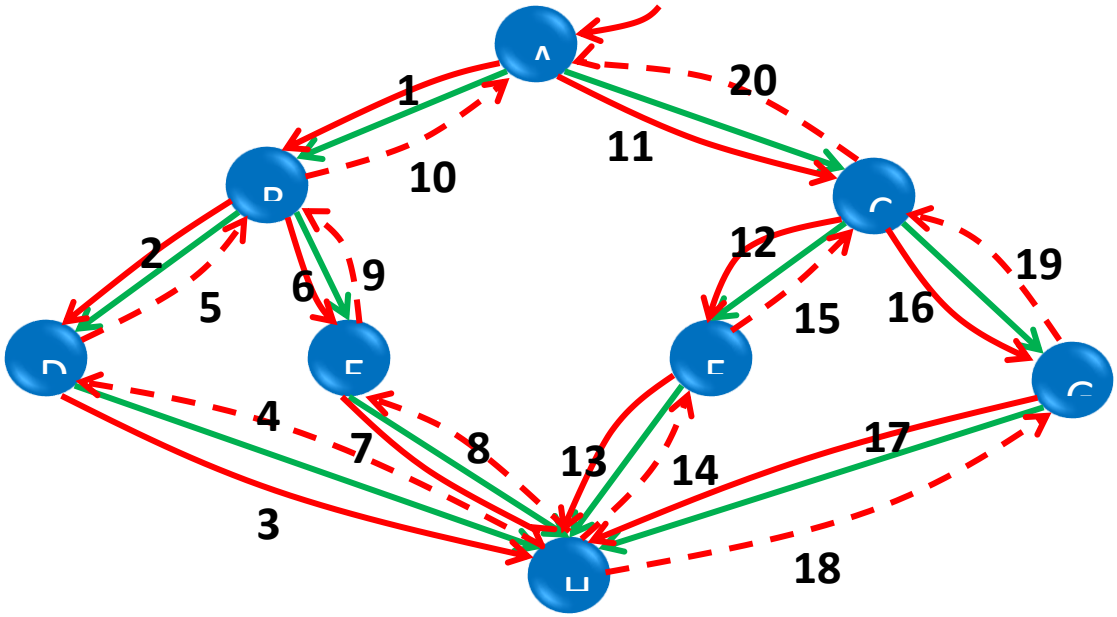
The time complexity for BFS can be expressed as O(|V| + |E|), where |V| is the number of vertices and |E| is the number of edges in the graph.

Depth-First Search: The algorithm starts at the starting node (selecting some node as the starting node in the graph case) and explores as far as possible along each branch before backtracking. The idea of DFS is as follows:

- In DFS, edges are explored out of the most recently discovered vertex, v. Only edges to unexplored vertices are explored.
- When all of v's edges have been explored, the search "backtracks" to explore edges leaving the vertex from which v was discovered.
- The process continues until all the vertices that are reachable from the original source vertex have discovered.



Following figure shows the order in which nodes are visited, while using DFS. Consider the graph used for BFS.



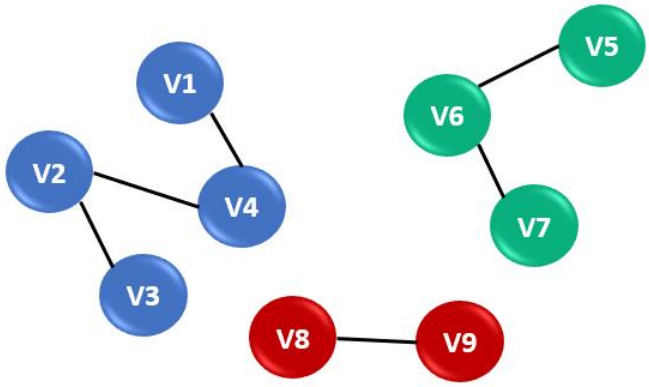
Algorithm for DFS

```
Algorithm DFS(v)
{
visited[v]:=1;
For each vertex w adjacent from v do
      if(visited[w]=0) then DFS(w);
}
```

The time complexity for DFS can be expressed as O(|E|), where |E| is the number of edges in the graph.

Connected Components

A connected component of an undirected graph is a subgraph in which each pair of nodes are connected with each other via a path. For example, the following figure shows three connected components.



- If G is an undirected graph, then one can determine whether or not it is connected by simply making a call to either DFS or BFS and then determining if there is any unvisited vertex.
- The connected components of a graph may be obtained by making repeated calls to either DFS(v) or BFS(v); where v is a vertex that has not yet been visited as shown in the following algorithm. The algorithm uses DFS (BFS may be used instead if desired).

```
void connected(void)
{
     for (i=0; i<n; i++)
     {
          if(!visited[i])
          {
          dfs(i);
          printf("\n");
          }
     }
}
```

Graph Applications

Graphs can be used to model many types of relations and processes in physical, biological, social and information systems. Many practical problems can be represented by graphs.

In computer science, graphs are used to represent networks of communication, data organization, computational devices, the flow of computation, etc. For instance, the link structure of a website can be represented by a directed graph, in which the vertices represent web pages and directed edges represent links from one page to another. A similar approach can be taken to problems in travel, biology, computer chip design, and many other fields.

Graph-theoretic methods, in various forms, have proven particularly useful in linguistics. Syntax of natural language is modeled using typed feature structures, which are directed acyclic graphs.

Graph theory is also used to study molecules in chemistry and physics. In chemistry a graph makes a natural model for a molecule, where vertices represent atoms and edges bonds. In statistical physics, graphs can represent local connections between interacting parts of a system, as well as the dynamics of a physical process on such systems.

Graph theory is also widely used in sociology as a way, for example, to measure actors' prestige or to explore rumor spreading, notably through the use of social network analysis software. Under the umbrella of social networks are many different types of graphs: Acquaintanceship and friendship graphs describe whether people know each other. Influence graphs model whether certain people can influence the behavior of others. Finally, collaboration graphs model whether two people work together in a particular way, such as acting in a movie together.

Likewise, graph theory is useful in biology and conservation efforts where a vertex can represent regions where certain species exist (or habitats) and the edges represent migration paths, or movement between the regions. This information is important when looking at breeding patterns or tracking the spread of disease, parasites or how changes to the movement can affect other species.

In mathematics, graphs are useful in geometry.

Weighted graphs, are used to represent structures in which pairwise connections have some numerical values. For example if a graph represents a road network, the weights could represent the length of each road. weighted graphs are useful for representing some practical problems as given below.

Finding the shortest/cheapest/fastest path for a car from one city to another, by using given roads can be solved by using graph algorithms. E.g. satellite navigation system that shows to drivers which way they should better go uses shortest path algorithms of graph theory.

Consider some communications stations (for telephony, cable television, Internet etc.) and a list of possible connections between them, having different costs. Find the cheapest way to connect these stations in a network, so that a station is connected to any other (directly, or through intermediate stations). This may be used for example to connect villages to cable television, or to Internet.

Also consider the problem of connecting villages by roads which is very similar to the above problem. The above two problems can be solved by using Minimal Spanning Trees.
A postman has to visit a set of streets in order to deliver mails and packages. It is needed to find a path that starts and ends at the post-office, and that passes through each street (edge) exactly once. This way the postman will deliver mails and packages to all streets he has to, and in the same time will spend minimum efforts/time for the road.

A warehouse should be placed in a city (a region) so that the sum of shortest distances to all other points (regions) is minimal. This is useful for lowering the cost of transporting goods from a warehouse to clients. Same thing can be considered for selecting the place of a shop, market, office, other buildings like hospital, fire/police station etc.