

Graphs and File Structure

Unit: 5

Data Structure and Algorithms

Course Details
(B Tech (AI-ML) 3rd Sem)



Sanchi Kaushik
Assistant Professor
(AI-ML)



Unit V Syllabus

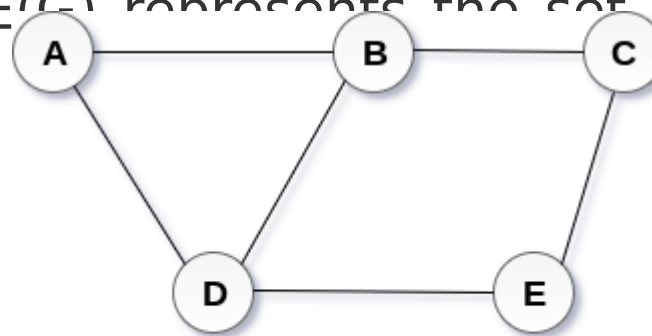
- **Graphs:** Terminology used with Graph, Data Structure for Graph Representations: Adjacency matrices, Adjacency List.
- **Graph Traversal:** Depth First Search and Breadth First Search. Connected Component, Spanning Trees, Minimum Cost Spanning Trees: Prim's and Kruskal's algorithm. Transitive Closure and Shortest Path algorithms: Dijkstra Algorithm.
- **File Structure:** Concepts of files, records and files, Sequential, Indexed and Random File Organization, Indexing structure for index files, hashing for direct files, Multi-Key file organization and Access Methods.

Graph

A graph can be defined as group of vertices and edges that are used to connect these vertices. A graph can be seen as a cyclic tree, where the vertices (Nodes) maintain any complex relationship among them instead of having parent child relationship.

Definition

A graph G can be defined as an ordered set $G(V, E)$ where $V(G)$ represents the set of vertices and $E(G)$ represents the set of edges which are used to connect these vertices.



Terminology of Graph in Data Structure

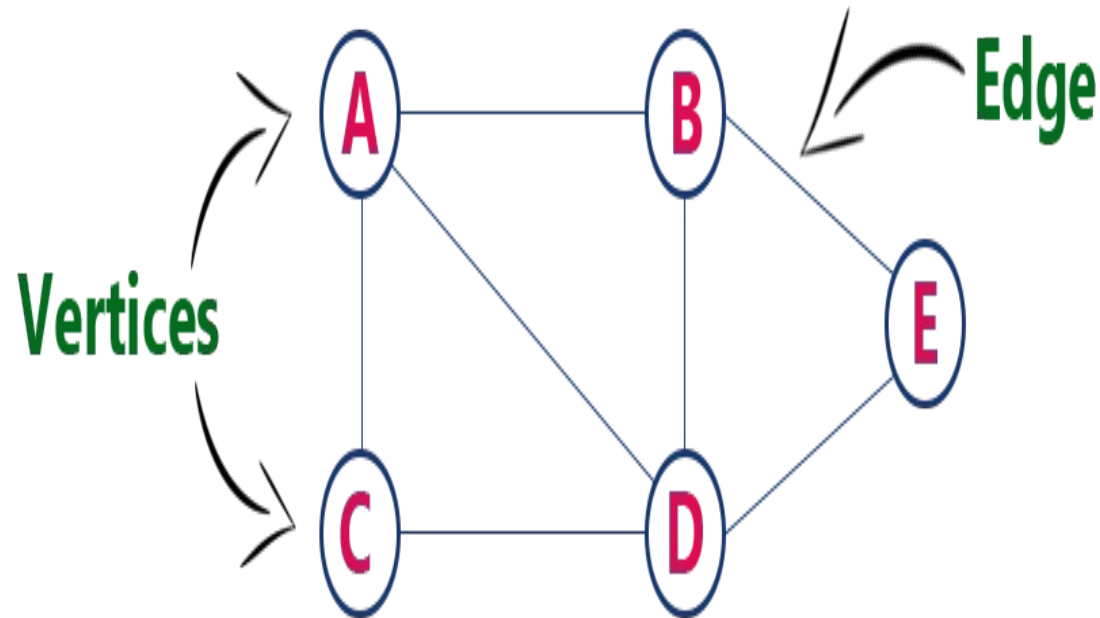
Term	Description
Vertex	Every individual data element is called a vertex or a node. In the above image, A, B, C, D & E are the vertices.
Edge (Arc)	It is a connecting link between two nodes or vertices. Each edge has two ends and is represented as (startingVertex, endingVertex).
Undirected Edge	It is a bidirectional edge.
Directed Edge	It is a unidirectional edge.
Weighted Edge.	An edge with value (cost) on it.
Degree	The total number of edges connected to a vertex in a graph.
Indegree	The total number of incoming edges connected to a vertex.
Outdegree	The total number of outgoing edges connected to a vertex.
Self-loop	An edge is called a self-loop if its two endpoints coincide with each other.
Adjacency	Vertices are said to be adjacent to one another if there is an edge connecting them

Example

The following is a graph with 5 vertices and 6 edges.

This graph G can be defined as $G = (V, E)$

Where $V = \{A, B, C, D, E\}$ and $E = \{(A, B), (A, C), (A, D), (B, D), (C, D), (B, E), (E, D)\}$.



Graph Terminology

Vertex

Individual data element of a graph is called as Vertex. **Vertex** is also known as **node**. In above example graph, A, B, C, D & E are known as vertices.

Edge

An edge is a connecting link between two vertices. **Edge** is also known as **Arc**. An edge is represented as (startingVertex, endingVertex). For example, in above graph the link between vertices A and B is represented as (A,B). In above example graph, there are 7 edges (i.e., (A,B), (A,C), (A,D), (B,D), (B,E), (C,D), (D,E)).

Edges are three types.

- **Undirected Edge** - An undirected edge is a bidirectional edge. If there is undirected edge between vertices A and B then edge (A , B) is equal to edge (B , A).
- **Directed Edge** - A directed edge is a unidirectional edge. If there is directed edge between vertices A and B then edge (A , B) is not equal to edge (B , A).
- **Weighted Edge** - A weighted edge is a edge with value (cost) on it.

Undirected Graph

A graph with only undirected edges is said to be undirected graph.

Directed Graph

A graph with only directed edges is said to be directed graph.

Mixed Graph

A graph with both undirected and directed edges is said to be mixed graph.

End vertices or Endpoints

The two vertices joined by edge are called end vertices (or endpoints) of that edge.

Origin

If a edge is directed, its first endpoint is said to be the origin of it.

Destination

If a edge is directed, its first endpoint is said to be the origin of it and the other endpoint is said to be the destination of that edge.

Adjacent

If there is an edge between vertices A and B then both A and B are said to be adjacent.

In other words, vertices A and B are said to be adjacent if there is an edge between them.

Incident

Edge is said to be incident on a vertex if the vertex is one of the endpoints of that edge.

Outgoing Edge

A directed edge is said to be outgoing edge on its origin 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.

Parallel edges or Multiple edges

If there are two undirected edges with same end vertices and two directed edges with same origin and destination, such edges are called parallel edges or multiple edges.

Self-loop

Edge (undirected or directed) is a self-loop if its two endpoints coincide with each other.

Simple Graph

A graph is said to be simple if there are no parallel and self-loop edges.

Path

A path is a sequence of alternate vertices and edges that starts at a vertex and ends at other vertex such that each edge is incident to its predecessor and successor vertex.

Graph Representations

Graph data structure is represented using following representations...

1.Adjacency Matrix

2.Incidence Matrix

3.Adjacency List

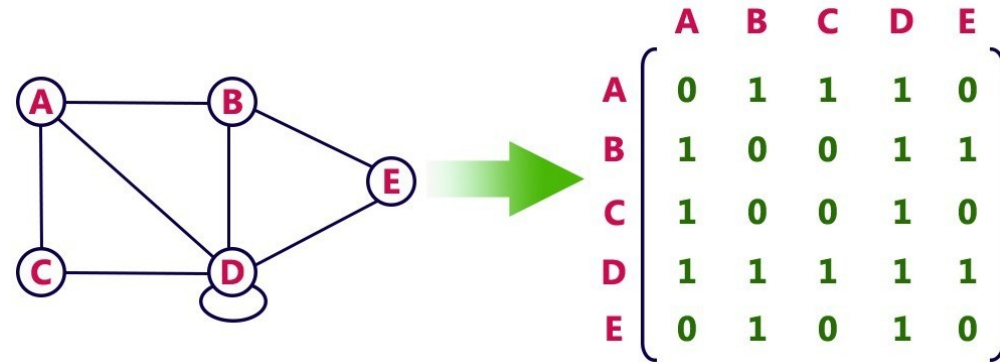
Adjacency Matrix

In this representation, the graph is represented using a matrix of size total number of vertices by a total number of vertices.

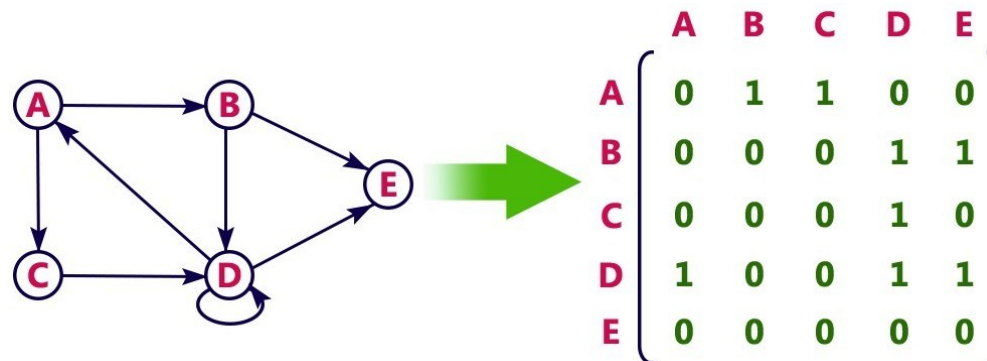
- That means a graph with 4 vertices is represented using a matrix of size 4X4.
- In this matrix, both rows and columns represent vertices.
- This matrix is filled with either 1 or 0.
- Here, 1 represents that there is an edge from row vertex to column vertex and 0 represents that there is no edge from row vertex to column vertex.

Graphs (CO5)

Undirected graph representation...



Directed graph representation...



Incidence Matrix

In this representation, the graph is represented using a matrix of size total number of vertices by a total number of edges.

That means graph with 4 vertices and 6 edges is represented using a matrix of size 4X6.

In this matrix, rows represent vertices and columns represents edges. This matrix is filled with 0 or 1 or -1.

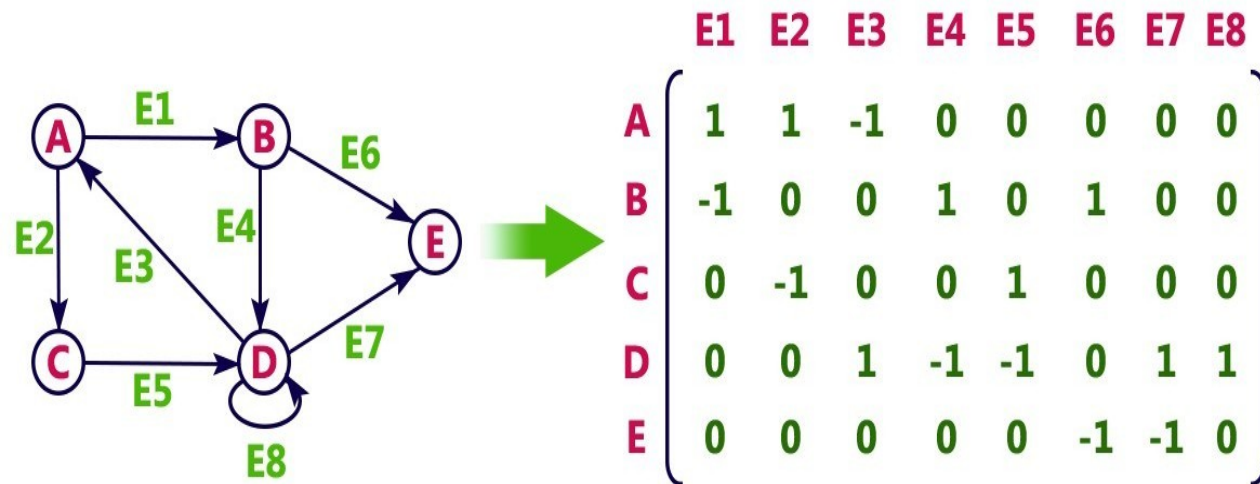
Incidence Matrix

Here, 0 represents that the row edge is not connected to column vertex

1 represents that the row edge is connected as the outgoing edge to column vertex and -

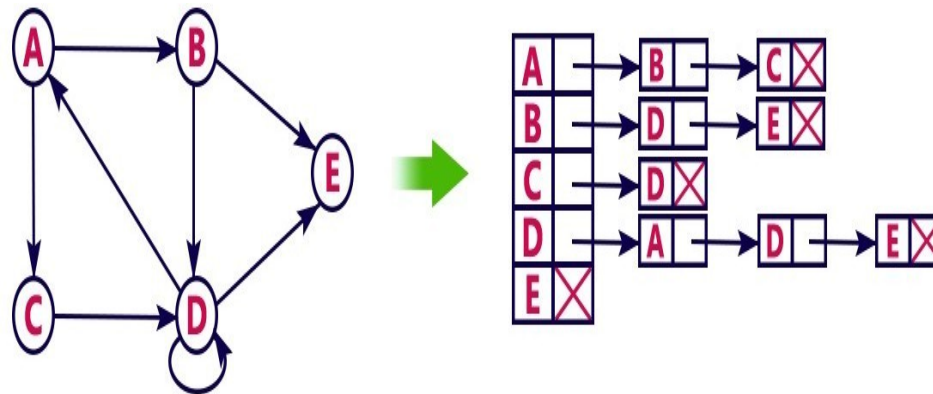
1 represents that the row edge is connected as the incoming edge to column vertex.

For example, consider the following directed graph representation...



Adjacency List

In this representation, every vertex of a graph contains list of its adjacent vertices.



Graph Traversal – DFS

Graph traversal is a technique used for a searching vertex in a graph. The graph traversal is also used to decide the order of vertices is visited in the search process. A graph traversal finds the edges to be used in the search process without creating loops. That means using graph traversal we visit all the vertices of the graph without getting into looping path.

There are two graph traversal techniques and they are as follows...

1.DFS (Depth First Search)

2.BFS (Breadth First Search)

DFS (Depth First Search)

DFS traversal of a graph produces a **spanning tree** as final result. **Spanning Tree** is a graph without loops.

We use **Stack data structure** with maximum size of total number of vertices in the graph to implement DFS traversal.

DFS (Depth First Search)

Step 1: SET STATUS = 1 (ready state) for each node in G

Step 2: Push the starting node A on the stack and set its STATUS = 2 (waiting state)

Step 3: Repeat Steps 4 and 5 until STACK is empty

Step 4: Pop the top node N. Process it and set its STATUS = 3 (processed state)

Step 5: Push on the stack all the neighbours of N that are in the ready state (whose STATUS = 1) and set their
STATUS = 2 (waiting state)

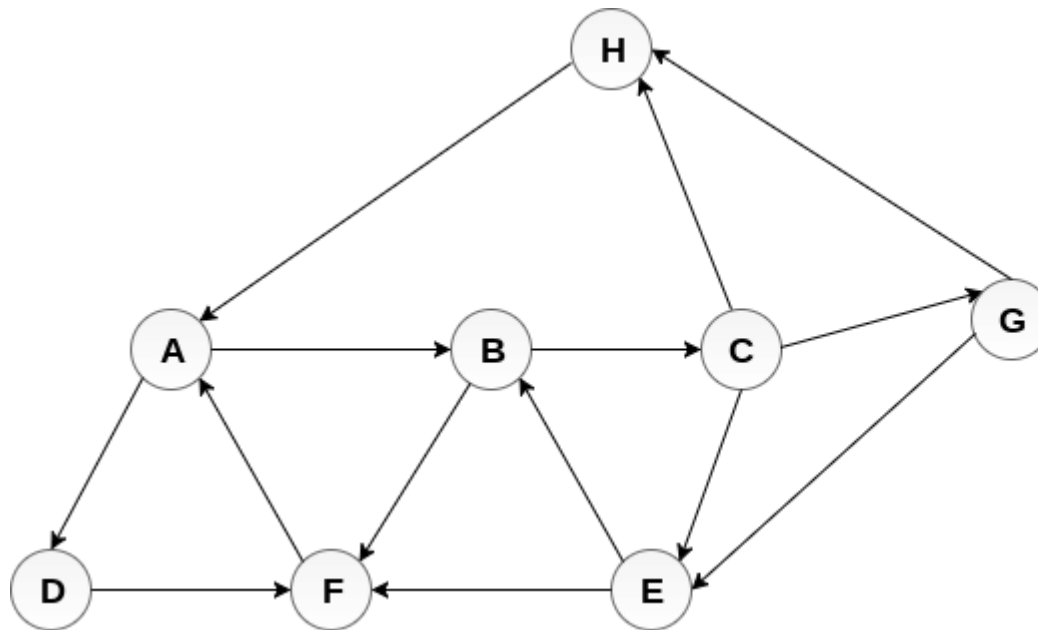
[END OF LOOP]

Step 6: EXIT

Graphs (CO5)

Example :

Consider the graph G along with its adjacency list, given in the figure below. Calculate the order to print all the nodes of the graph starting from node H, by using depth first search (DFS) algorithm.



Adjacency Lists

A : B, D

B : C, F

C : E, G, H

G : E, H

E : B, F

F : A

D : F

H : A

Solution :

Push H onto the stack

1.STACK : H

POP the top element of the stack i.e. H, print it and push all the neighbours of H onto the stack that are in ready state.

2.Print H

3.STACK : A

Pop the top element of the stack i.e. A, print it and push all the neighbours of A onto the stack that are in ready state.

4.Print A

5.Stack : B, D

Pop the top element of the stack i.e. D, print it and push all the neighbours of D onto the stack that are in ready state.

6.Print D

7.Stack : B, F

Pop the top element of the stack i.e. F, print it and push all the neighbours of F onto the stack that are in ready state.

Graphs (CO5)

1.Print F

2.Stack : B

Pop the top of the stack i.e. B and push all the neighbours

3.Print B

4.Stack : C

Pop the top of the stack i.e. C and push all the neighbours.

5.Print C

6.Stack : E, G

Pop the top of the stack i.e. G and push all its neighbours.

7.Print G

8.Stack : E

Pop the top of the stack i.e. E and push all its neighbours.

9.Print E

10.Stack :

Hence, the stack now becomes empty and all the nodes of the graph have been traversed.

The printing sequence of the graph will be :

11.H \rightarrow A \rightarrow D \rightarrow F \rightarrow B \rightarrow C \rightarrow G \rightarrow E

BFS (Breadth First Search)

BFS traversal of a graph produces a **spanning tree** as final result. **Spanning Tree** is a graph without loops. We use **Queue data structure** with maximum size of total number of vertices in the graph to implement BFS traversal.

We use the following steps to implement BFS traversal...

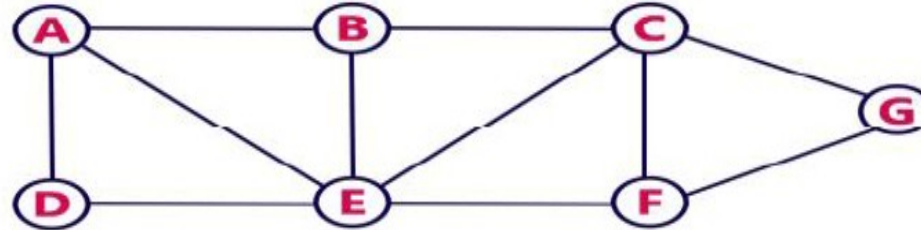
- **Step 1** - Define a Queue of size total number of vertices in the graph.
- **Step 2** - Select any vertex as **starting point** for traversal. Visit that vertex and insert it into the Queue.

Graphs (CO5)

- **Step 3** - Visit all the non-visited **adjacent** vertices of the vertex which is at front of the Queue and insert them into the Queue.
- **Step 4** - When there is no new vertex to be visited from the vertex which is at front of the Queue then delete that vertex.
- **Step 5** - Repeat steps 3 and 4 until queue becomes empty.
- **Step 6** - When queue becomes empty, then produce final spanning tree by removing unused edges from the graph.

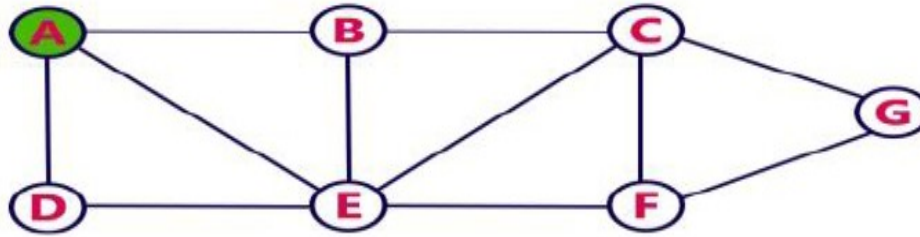
Graphs (CO5)

Consider the following example graph to perform BFS traversal



Step 1:

- Select the vertex **A** as starting point (visit **A**).
- Insert **A** into the Queue.

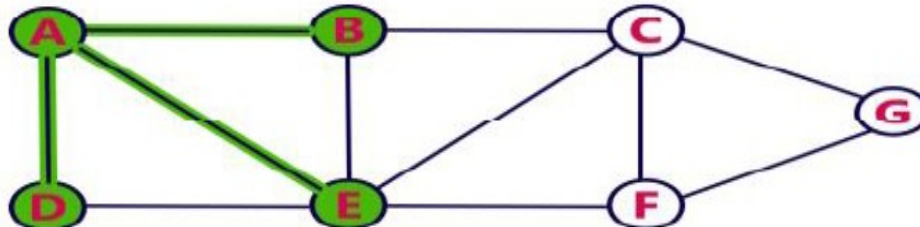


Queue



Step 2:

- Visit all adjacent vertices of **A** which are not visited (**D**, **E**, **B**).
- Insert newly visited vertices into the Queue and delete A from the Queue..



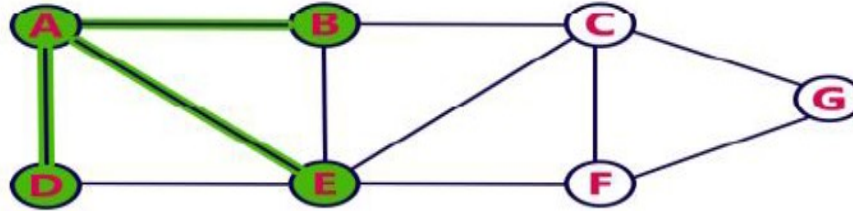
Queue



Graphs (CO5)

Step 3:

- Visit all adjacent vertices of **D** which are not visited (there is no vertex).
- Delete D from the Queue.

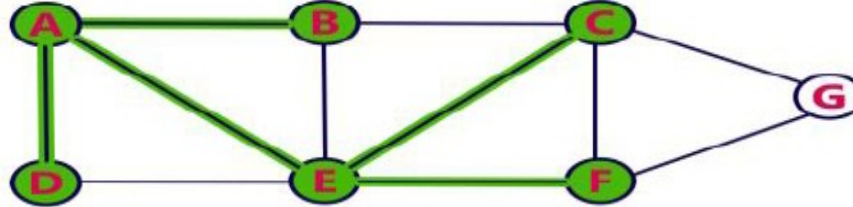


Queue



Step 4:

- Visit all adjacent vertices of **E** which are not visited (**C, F**).
- Insert newly visited vertices into the Queue and delete E from the Queue.

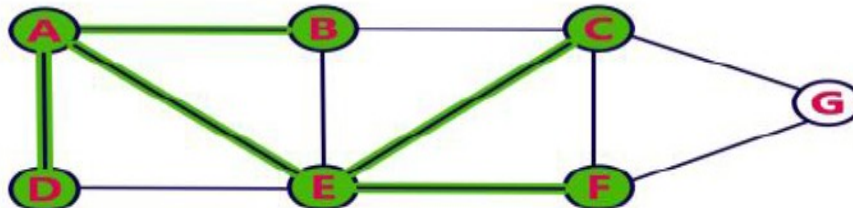


Queue



Step 5:

- Visit all adjacent vertices of **B** which are not visited (**there is no vertex**).
- Delete **B** from the Queue.



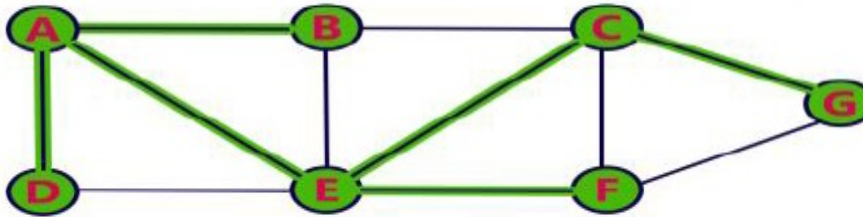
Queue



Graphs (CO5)

Step 6:

- Visit all adjacent vertices of **C** which are not visited (**G**).
- Insert newly visited vertex into the Queue and delete **C** from the Queue.

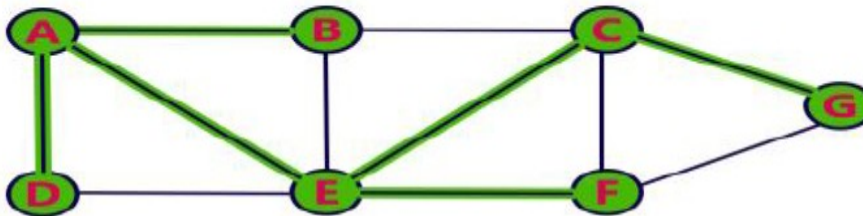


Queue



Step 7:

- Visit all adjacent vertices of **F** which are not visited (**there is no vertex**).
- Delete **F** from the Queue.

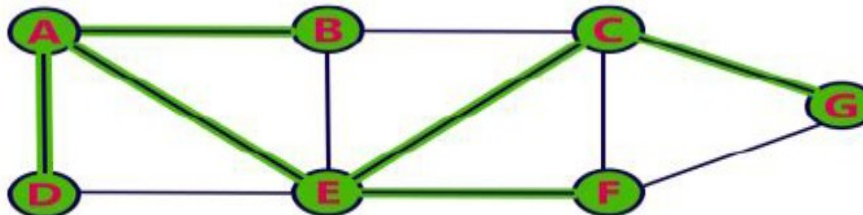


Queue



Step 8:

- Visit all adjacent vertices of **G** which are not visited (**there is no vertex**).
- Delete **G** from the Queue.



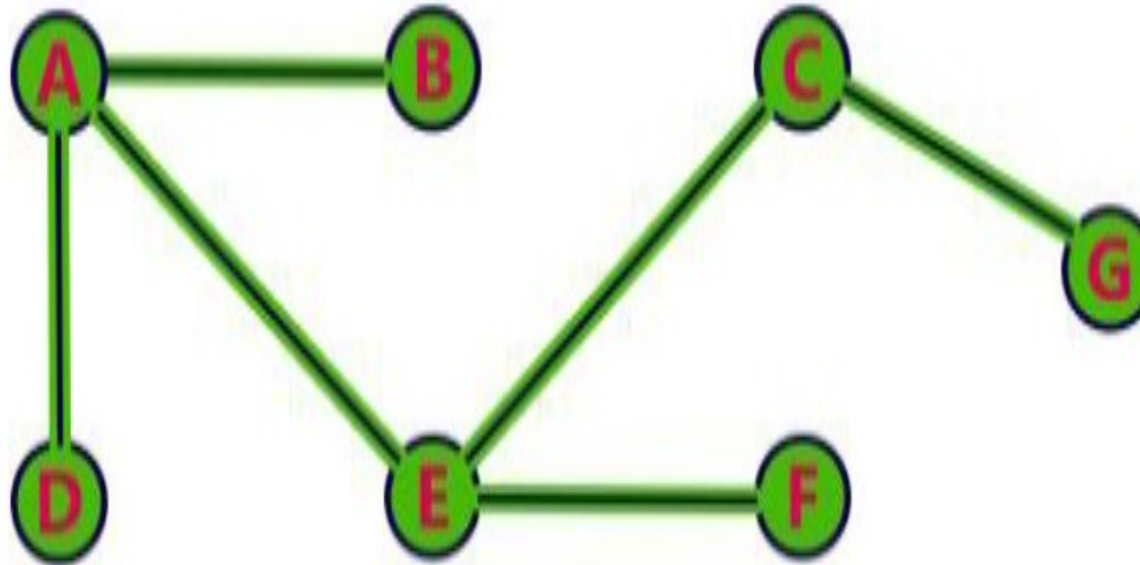
Queue



Graphs (CO5)

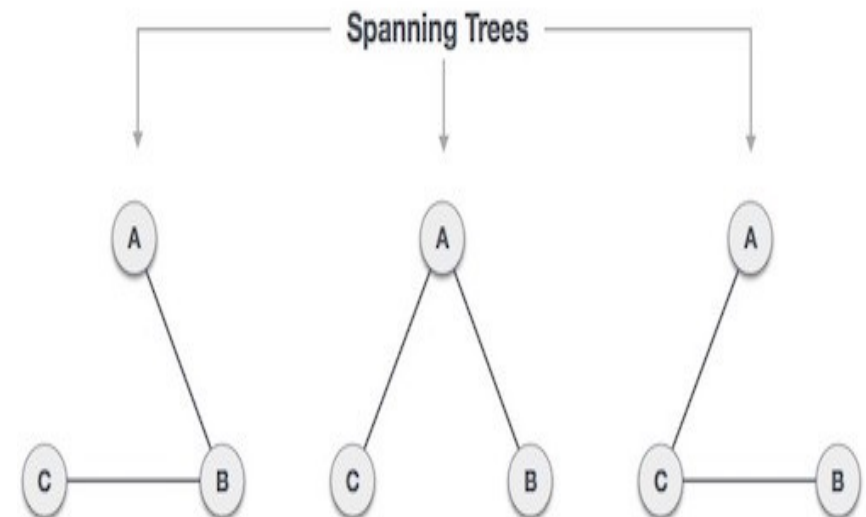
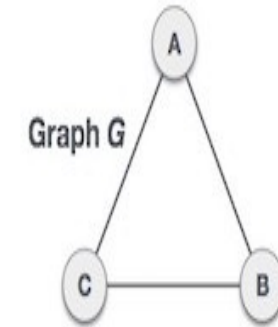
Queue became Empty, So, stop the BFS process.

Final result of BFS is a Spanning tree as shown below



Spanning Tree

- Spanning tree of a graph can be defined as a sub graph that contain all the vertices and is a tree
- In other words, Spanning tree is a non-cyclic sub-graph of a connected and undirected graph G that connects all the vertices together.
- A graph G can have multiple spanning trees.



General Properties of Spanning Tree

- A connected graph G can have more than one spanning tree.
- All possible spanning trees of graph G , have the same number of edges and vertices.
- The spanning tree does not have any cycle (loops).
- Removing one edge from the spanning tree will make the graph disconnected, i.e. the spanning tree is **minimally connected**.
- Adding one edge to the spanning tree will create a circuit or loop, i.e. the spanning tree is **maximally acyclic**.

General Properties of Spanning Tree

- Spanning tree has **$n-1$** edges, where **n** is the number of nodes (vertices).
- From a complete graph, by removing maximum **$e - n + 1$** edges, we can construct a spanning tree.
- A complete graph can have maximum **n^{n-2}** number of spanning trees.

Minimum Spanning Tree:

- A minimum spanning tree is defined for a weighted graph.
- A spanning tree having minimum weight is defined as a minimum spanning tree.
- This weight depends on the weight of the edges.
- In real-world applications, the weight could be the distance between two points, cost associated with the edges or simply an arbitrary value associated with the edges.

Minimum Spanning Tree Algorithms:

There are various algorithms in computer science that help us find the minimum spanning tree for a weighted graph. Some of these algorithms are:

1. Prim's Algorithm
2. Kruskal's Algorithm

Prim's Algorithm

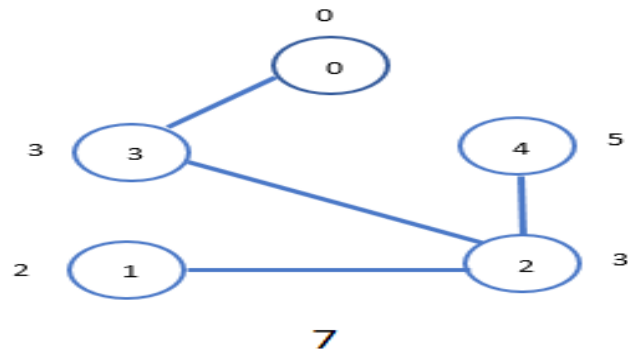
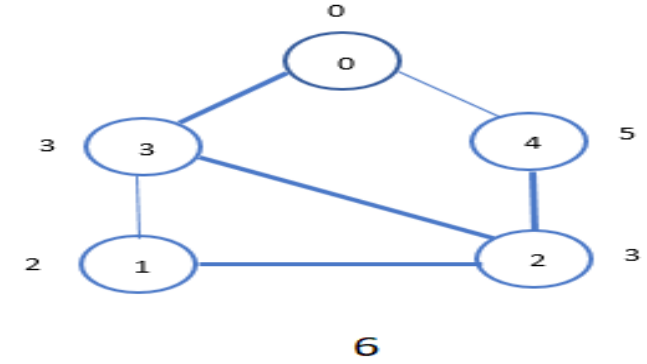
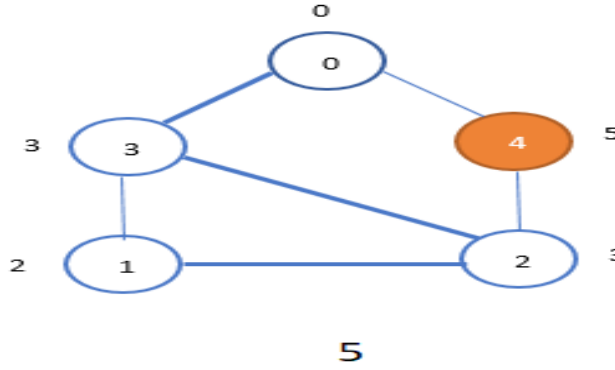
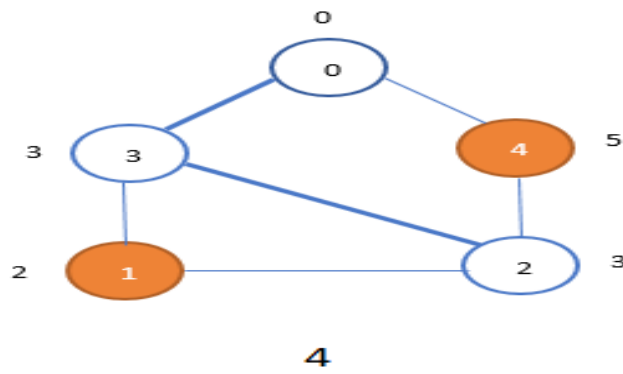
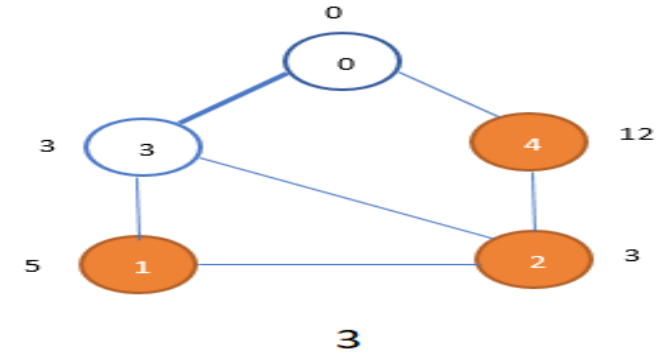
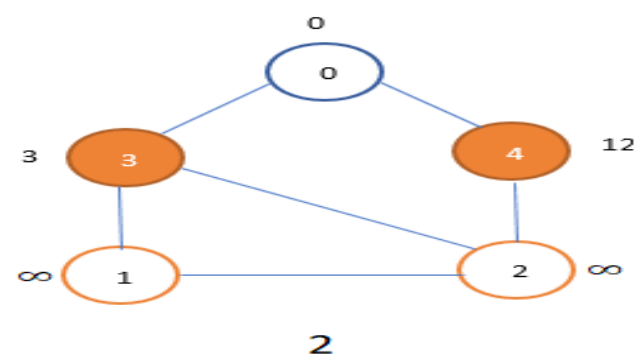
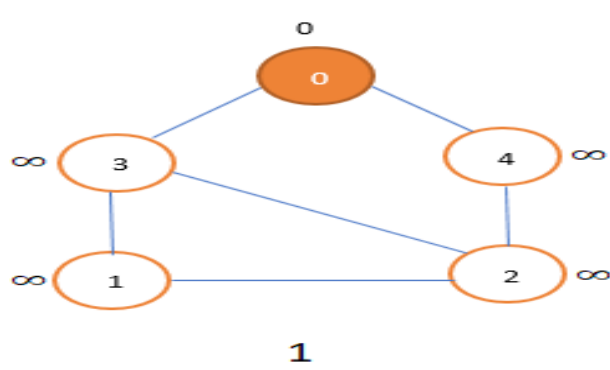
Prim's Algorithm is used to find the minimum spanning tree from a graph. Prim's algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized.

Prim's algorithm starts with the single node and explore all the adjacent nodes with all the connecting edges at every step. The edges with the minimal weights causing no cycles in the graph got selected.




Prim's Algorithm

1. For each vertex U
 1. $U.Key = \infty$
 2. $U.Parent = Null$
2. Any random vertex R
3. $R.Key = 0$
4. Min-Priority Queue $Q = G.V$ (priority on vertex's key)
5. While $Q \neq Empty$
 1. $U = Q.Dequeue()$
 2. For each V in $G.Adj[U]$
 1. If V in Q And $Weight(U, V) < V.Key$
 1. $V.Parent = U$
 2. $V.Key = Weight(U, V)$

Graphs (CO5)



Symbols

-  Not in Queue, not yet processed
-  In Queue
-  Processed

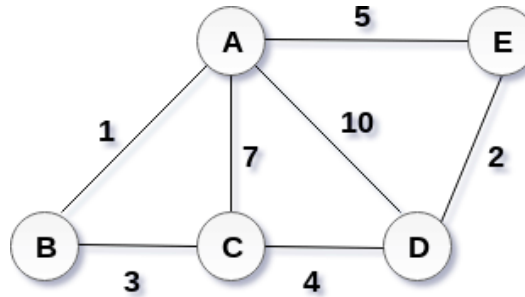
Kruskal's Algorithm: Kruskal's algorithm works on greedy approach, it takes edges first which are smaller in weight.

1. Define an empty List $A = []$
2. For each vertex V
 1. Make-Set(V)
3. Sort edges of graph order by weight
4. For each edge $E(u, v)$
 1. If Find-Set(u) \neq Find-Set(v)
 1. Append $E(u, v)$ in A
 2. Union (u, v)
5. Return A

Graphs (CO5)

Example :

Apply the Kruskal's algorithm on the graph given as follows.



Solution:

the weight of the edges given as:

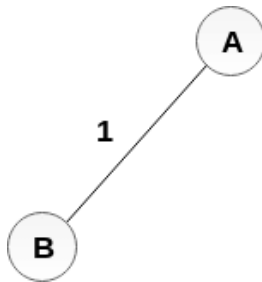
Edge	AE	AD	AC	AB	BC	CD	DE
Weight	5	10	7	1	3	4	2

Graphs (CO5)

Sort the edges according to their weights.

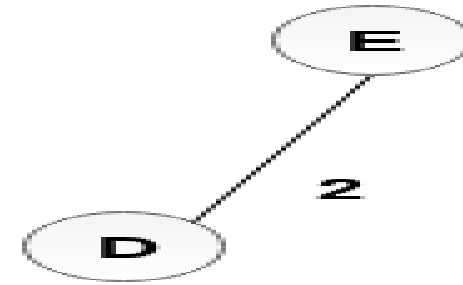
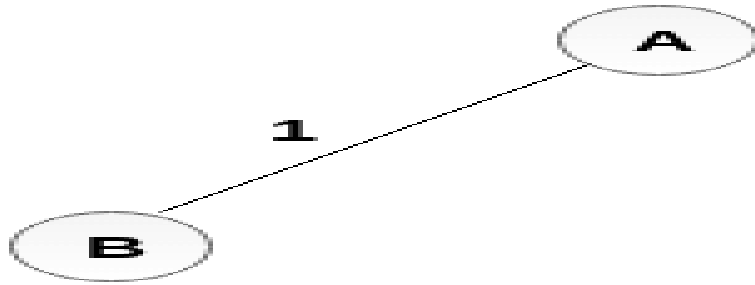
Edge	AB	DE	BC	CD	AE	AC	AD
Weight	1	2	3	4	5	7	10

Start constructing the tree; Add AB to the MST;

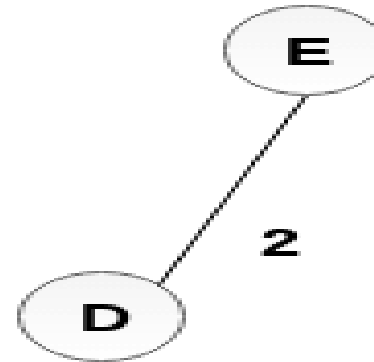
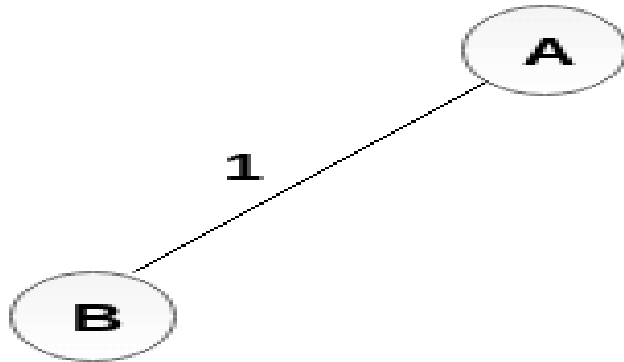


Add DE to the MST;

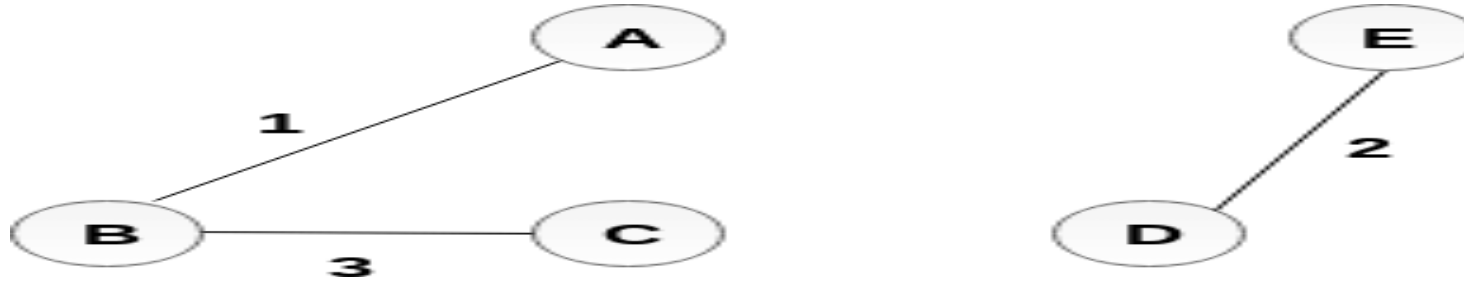
Graphs (CO5)



Add BC to the MST;



Graphs (CO5)



The next step is to add AE, but we can't add that as it will cause a cycle.

The next edge to be added is AC, but it can't be added as it will cause a cycle.

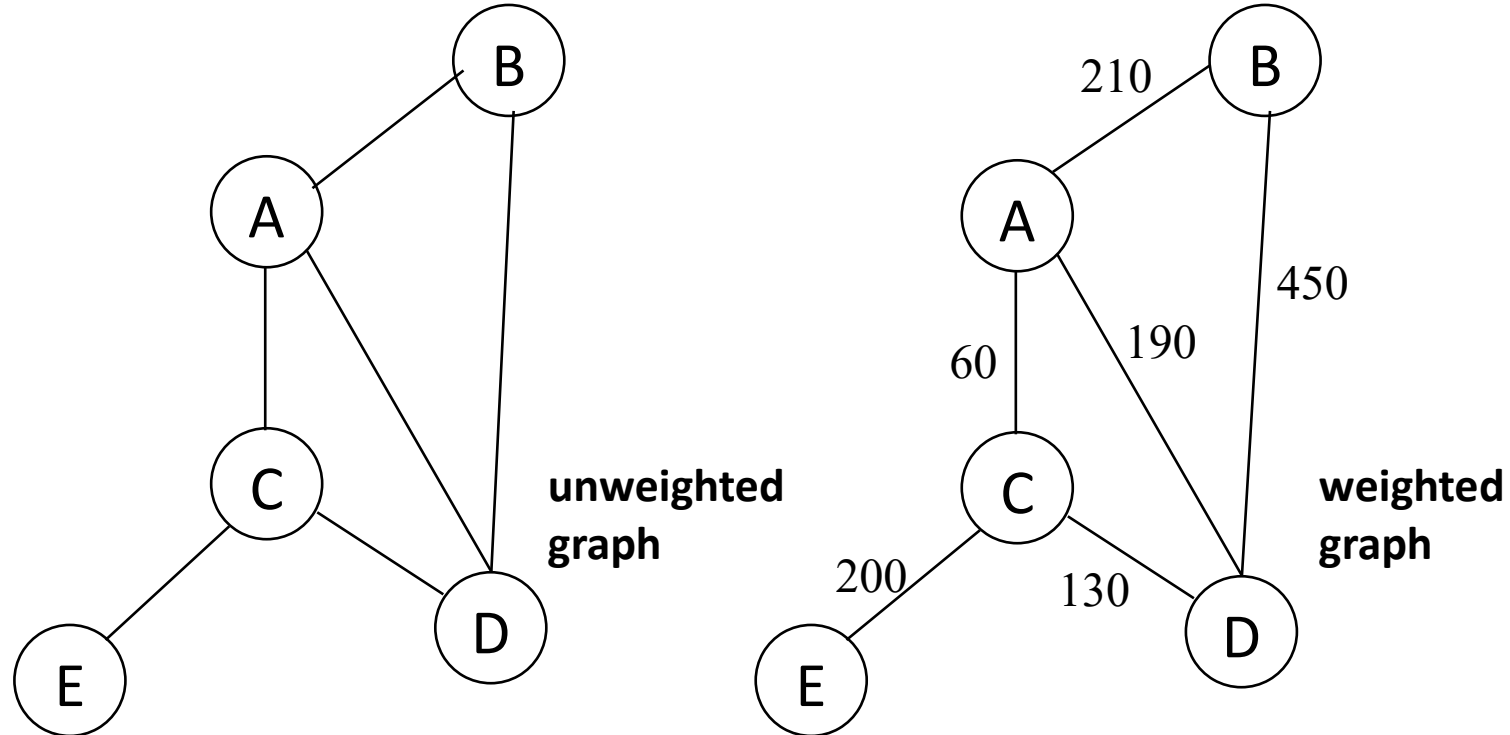
The next edge to be added is AD, but it can't be added as it will contain a cycle.

Hence, the final MST is the one which is shown in the step 4. the cost of MST = $1 + 2 + 3 + 4 = 10$.

Shortest Path Problems (CO5)

What is shortest path ?

- shortest length between two vertices for an unweighted graph:
- smallest cost between two vertices for a weighted graph:



Shortest Path Problems (CO5)

- How can we find the shortest route between two points on a map?
- Model the problem as a graph problem:
 - Road map is a weighted graph:
 - vertices** = cities
 - edges** = road segments between cities
 - edge weights** = road distances
 - Goal: find a shortest path between two vertices (cities)

Variants of Shortest Paths (CO5)

- **Single-source shortest path**
 - $G = (V, E) \Rightarrow$ find a shortest path from a given source vertex s to each vertex $v \in V$
- **Single-destination shortest path**
 - Find a shortest path to a given destination vertex t from each vertex v
 - Reverse the direction of each edge \Rightarrow single-source
- **Single-pair shortest path**
 - Find a shortest path from u to v for given vertices u and v
 - Solve the single-source problem
- **All-pairs shortest-paths**
 - Find a shortest path from u to v for every pair of vertices u and v

Shortest Path Algorithm (CO5)

Alg.: INITIALIZE-SINGLE-SOURCE(V, s)

1. **for** each $v \in V$
2. **do** $d[v] \leftarrow \infty$
3. $\pi[v] \leftarrow \text{NIL}$
4. $d[s] \leftarrow 0$

- All the shortest-paths algorithms start with INITIALIZE-SINGLE-SOURCE.

Dijkstra's Algorithm (CO5)

- Single-source shortest path problem:
 - No negative-weight edges: $w(u, v) > 0 \forall (u, v) \in E$
- Maintains two sets of vertices:
 - S = vertices whose final shortest-path weights have already been determined
 - Q = vertices in $V - S$: min-priority queue
 - Keys in Q are estimates of shortest-path weights ($d[v]$)
- Repeatedly select a vertex $u \in V - S$, with the minimum shortest-path estimate $d[v]$

Dijkstra Algorithm (CO5)

Dijkstra (G, w, s)

1. INITIALIZE-SINGLE-SOURCE(V, s) $\leftarrow \Theta(V)$
2. $S \leftarrow \emptyset$
3. $Q \leftarrow V[G]$ $\leftarrow O(V)$ build min-heap
4. **while** $Q \neq \emptyset$ \leftarrow Executed $O(V)$ times
5. **do** $u \leftarrow \text{EXTRACT-MIN}(Q)$ $\leftarrow O(\lg V)$
6. $S \leftarrow S \cup \{u\}$
7. **for** each vertex $v \in \text{Adj}[u]$
8. **do** RELAX(u, v, w) $\leftarrow O(E)$ times; $O(\lg V)$

Running time: $O(V \lg V + E \lg V) = O(E \lg V)$

Floyd Warshall Algorithm (CO5)

Floyd-Warshall Algorithm is an algorithm for finding the shortest path between all the pairs of vertices in a weighted graph. This algorithm works for both the directed and undirected weighted graphs. But, it does not work for the graphs with negative cycles (where the sum of the edges in a cycle is negative).

Floyd Warshall Algorithm (CO5)

Floyd Warshall Algorithm is as shown below-

Create a $|V| \times |V|$ matrix // It represents the distance between every pair of vertices as given

For each cell (i,j) in M do-

if $i = j$

$M[i][j] = 0$ // For all diagonal elements, value = 0

if (i, j) is an edge in E

$M[i][j] = \text{weight}(i,j)$ // If there exists a direct edge between the vertices, value = weight of edge

Floyd Warshall Algorithm (CO5)

else

$M[i][j] = \text{infinity}$ // If there is no direct edge between the
vertices, value = ∞

for k from 1 to $|V|$

for i from 1 to $|V|$

for j from 1 to $|V|$

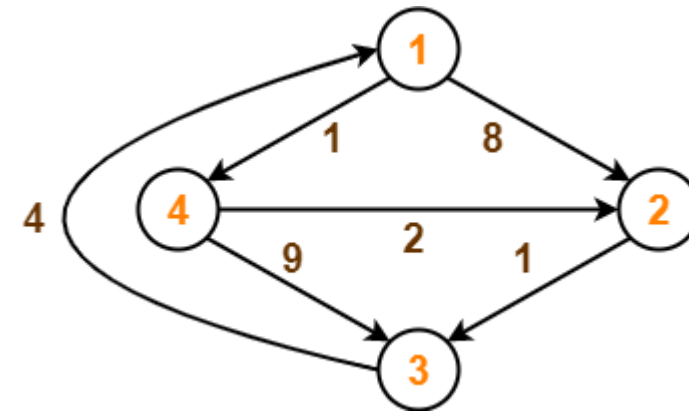
if $M[i][j] > M[i][k] + M[k][j]$

$M[i][j] = M[i][k] + M[k][j]$

Floyd Warshall Problem (CO5)

Problem-

Consider the following directed weighted graph-



Using Floyd Warshall Algorithm, find the shortest path distance between every pair of vertices.

Floyd Warshall Algorithm (CO5)

Solution-

Step-01:

- Remove all the self loops and parallel edges (keeping the lowest weight edge) from the graph.
- In the given graph, there are neither self edges nor parallel edges.

Step-02:

- Write the initial distance matrix.
- It represents the distance between every pair of vertices in the form of given weights.

Floyd Warshall Algorithm (CO5)

Step-02: (Continue)

- For diagonal elements (representing self-loops), distance value = 0.
- For vertices having a direct edge between them, distance value = weight of that edge.
- For vertices having no direct edge between them, distance value = ∞ .

Initial distance matrix for the given graph is-

$$D_0 = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & \infty & 0 & \infty \\ \infty & 2 & 9 & 0 \end{bmatrix} \end{matrix}$$

Floyd Warshall Algorithm (CO5)

Step-03:

Using Floyd Warshall Algorithm, write the following 4 matrices-

$$D_1 = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 9 & 0 \end{bmatrix} \end{matrix}$$

$$D_2 = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 8 & 9 & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 3 & 0 \end{bmatrix} \end{matrix}$$

Floyd Warshall Algorithm (CO5)

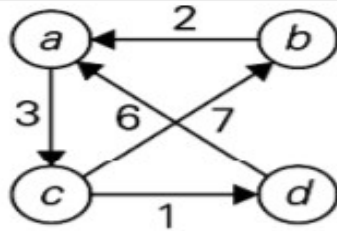
Step-03:

$$D_3 = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 8 & 9 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 12 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix} \end{matrix}$$

The last matrix D_4 represents the shortest path distance between every pair of vertices.

$$D_4 = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 3 & 4 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 7 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix} \end{matrix}$$

Floyd Warshall Algorithm (CO5)



$$D^{(0)} = \begin{matrix} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix} \end{matrix}$$

Lengths of the shortest paths with no intermediate vertices ($D^{(0)}$ is simply the weight matrix).

$$D^{(1)} = \begin{matrix} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \mathbf{5} & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \mathbf{9} & 0 \end{bmatrix} \end{matrix}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 1, i.e. just a (note two new shortest paths from b to c and from d to c).

$$D^{(2)} = \begin{matrix} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ \mathbf{9} & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{bmatrix} \end{matrix}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 2, i.e. a and b (note a new shortest path from c to a).

$$D^{(3)} = \begin{matrix} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{bmatrix} 0 & \mathbf{10} & 3 & \mathbf{4} \\ 2 & 0 & 5 & \mathbf{6} \\ 9 & 7 & 0 & 1 \\ 6 & \mathbf{16} & 9 & 0 \end{bmatrix} \end{matrix}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 3, i.e. a , b , and c (note four new shortest paths from a to b , from a to d , from b to d , and from d to b).

$$D^{(4)} = \begin{matrix} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ \mathbf{7} & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix} \end{matrix}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 4, i.e. a , b , c , and d (note a new shortest path from c to a).

Records and file (CO5)

- **Record**– It is a collection of units of information about a particular entity.
Passenger of an airplane, an employee of an organization, or an article sold from a store.
- **File**-A collection of records involving a set of entities with certain aspects in common and organized for some particular purpose is called a file. For example collection of records of all passengers.

File Organization (CO5)

- File Organization refers to the logical relationships among various records that constitute the file, particularly with respect to the means of identification and access to any specific record.
- In simple terms, Storing the files in certain order is called file Organization.
- **File Structure** refers to the format of the label and data blocks and of any logical control record.

Types of File Organization (CO5)

There are three types of organizing the file:

1. Sequential access file organization
2. Direct access file organization
3. Indexed sequential access file organization

Sequential access file organization (CO5)

- Storing and sorting in contiguous block within files on tape or disk is called as **sequential access file organization**.
- In sequential access file organization, all records are stored in a sequential order. The records are arranged in the ascending or descending order of a key field.
- Sequential file search starts from the beginning of the file and the records can be added at the end of the file.
- In sequential file, it is not possible to add a record in the middle of the file without rewriting the file.

Sequential access file organization (CO5)

Advantages of sequential file –

- It is simple to program and easy to design.
- Sequential file is best use if storage space.

Disadvantages of sequential file –

- Sequential file is time consuming process.
- It has high data redundancy.
- Random searching is not possible.

Direct access file organization (CO5)

- Direct access file is also known as random access or relative file organization.
- In direct access file, all records are stored in direct access storage device (DASD), such as hard disk. The records are randomly placed throughout the file.
- The records does not need to be in sequence because they are updated directly and rewritten back in the same location.
- This file organization is useful for immediate access to large amount of information. It is used in accessing large databases.
- It is also called as **hashing**.

Direct access file organization(CO5)

Advantages of direct access file organization –

- Direct access file helps in online transaction processing system (OLTP) like online railway reservation system.
- In direct access file, sorting of the records are not required.
- It accesses the desired records immediately.
- It updates several files quickly.
- It has better control over record allocation.

Disadvantages of direct access file organization –

- Direct access file does not provide back up facility.
- It is expensive.
- It has less storage space as compared to sequential file.

Indexed sequential access file organization(CO5)

- Indexed sequential access file combines both sequential file and direct access file organization.
- In indexed sequential access file, records are stored randomly on a direct access device such as magnetic disk by a primary key.
- This file have multiple keys. These keys can be alphanumeric in which the records are ordered is called primary key.
- The data can be access either sequentially or randomly using the index. The index is stored in a file and read into memory when the file is opened.

Advantages of Indexed sequential access file organization –

- In indexed sequential access file, sequential file and random file access is possible.
- It accesses the records very fast if the index table is properly organized.
- The records can be inserted in the middle of the file.
- It provides quick access for sequential and direct processing.
- It reduces the degree of the sequential search.

Disadvantages of Indexed sequential access file organization –

- Indexed sequential access file requires unique keys and periodic reorganization.
- Indexed sequential access file takes longer time to search the index for the data access or retrieval.
- It requires more storage space.
- It is expensive because it requires special software.
- It is less efficient in the use of storage space as compared to other file organizations.

1. Dijkstra's Algorithm is used to solve _____ problems.
 - a) All pair shortest path
 - b) Single source shortest path**
 - c) Network flow
 - d) Sorting

2. Which of the following is the most commonly used data structure for implementing Dijkstra's Algorithm?
 - a) Max priority queue
 - b) Stack
 - c) Circular queue
 - d) Min priority queue**

3. Floyd Warshall's Algorithm is used for solving _____

- a) All pair shortest path problems**
- b) Single Source shortest path problems
- c) Network flow problems
- d) Sorting problems

4. Floyd Warshall's Algorithm can be applied on _____

- a) Undirected and unweighted graphs
- b) Undirected graphs
- c) Directed graphs**
- d) Acyclic graphs

Youtube/other Video Links-

- <https://nptel.ac.in/courses/106/106/106106127/>
- <https://nptel.ac.in/courses/106/106/106106145/>

1. What are various file organization techniques?
2. How you can find number of shortest paths from source to destination using Dijkstra's algorithm?
3. Suppose you are given a graph where each edge represents the path cost and each vertex has also a cost which represents that, if you select a path using this node, the cost will be added with the path cost. How can it be solved using Dijkstra's algorithm?
4. Write the algorithm of Floyd Warshal all pair shortest path.
5. Write the algorithm for prims and Kruskal.

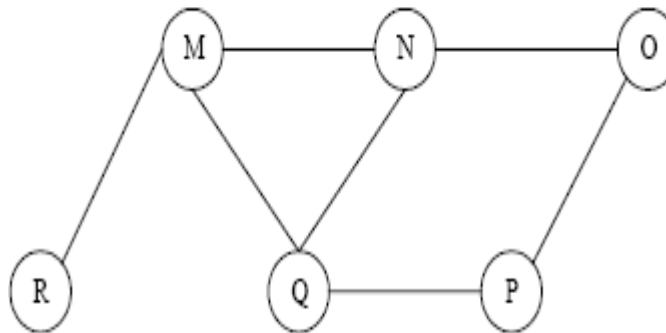
6. What do you mean by minimum spanning tree.
7. Define adjacency list and adjacency matrix.
8. Write the applications of graph.
9. Floyd Warshall Algorithm can be used for finding _____
 - a) Single source shortest path
 - b) Topological sort
 - c) Minimum spanning tree
 - d) Transitive closure**

1. Which of the following algorithms can be used to most efficiently determine the presence of a cycle in a given graph ?
 - a. Depth First Search
 - b. Breadth First Search
 - c. Prim's Minimum Spanning Tree Algorithm
 - d. Kruskal' Minimum Spanning Tree Algorithm
2. Traversal of a graph is different from tree because
 - a. There can be a loop in graph so we must maintain a visited flag for every vertex
 - b. DFS of a graph uses stack, but inorder traversal of a tree is recursive
 - c. BFS of a graph uses queue, but a time efficient BFS of a tree is recursive.
 - d. All of above

3. What are the appropriate data structures for following algorithms?

- a. Breadth First Search
- b. Depth First Search
- c. Prim's Minimum Spanning Tree
- d. Kruskal' Minimum Spanning Tree

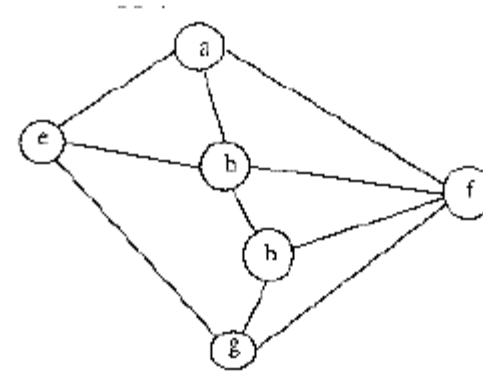
4. The Breadth First Search algorithm has been implemented using the queue data structure. One possible order of visiting the nodes of the following graph is



5. Consider the following graph,
Among the following sequences:

- (I) a b e g h f
- (II) a b f e h g
- (III) a b f h g e
- (IV) a f g h b e

Which are depth first traversals of the above graph?



6. Given two vertices in a graph s and t, which of the two traversals (BFS and DFS) can be used to find if there is path from s to t?

- a) Only BFS
- b) Only DFS
- c) Both BFS and DFS
- d) Neither BFS nor DFS

7. Which of the following is the most commonly used data structure for implementing Dijkstra's Algorithm?

- a. Max priority queue
- b. Stack
- c. Circular queue
- d. Min priority queue

8. What is the time complexity of Dijkstra's algorithm?

- a. $O(N)$
- b. $O(N^3)$
- c. $O(N^2)$
- d. $O(\log N)$

9. Which of the following condition is sufficient to detect cycle in a directed graph?
- a. There is an edge from currently being visited node to an already visited node.
 - b. There is an edge from currently being visited node to an ancestor of currently visited node in DFS forest.
 - c. Every node is seen twice in DFS.
 - d. None of the above
10. Dijkstra's Algorithm is used to solve problems.
- a. All pair shortest path
 - b. Single source shortest path
 - c. Network flow
 - d. Sorting

11. What procedure is being followed in Floyd Warshall Algorithm?

- a) Top down
- b) Bottom up**
- c) Big bang
- d) Sandwich

12. Floyd- Warshall algorithm was proposed by _____

- a) Robert Floyd and Stephen Warshall**
- b) Stephen Floyd and Robert Warshall
- c) Bernad Floyd and Robert Warshall
- d) Robert Floyd and Bernad Warshall

13. Which of the following is false in the case of a spanning tree of a graph G ?

- a) It is tree that spans G
- b) It is a subgraph of the G
- c) It includes every vertex of the G
- d) It can be either cyclic or acyclic**

14. Every graph has only one minimum spanning tree.

- a) True
- b) False**

15. The Data structure used in standard implementation of Breadth First Search is?

- a) Stack
- b) Queue**
- c) Linked List
- d) Tree

16. The Breadth First Search traversal of a graph will result into?

- a) Linked List
- b) Tree**
- c) Graph with back edges
- d) Arrays

17. Which of the following data structure is used to implement DFS?

- a) linked list
- b) tree
- c) stack**
- d) queue

18. Which of the following traversal in a binary tree is similar to depth first traversal?

- a) level order
- b) post order
- c) pre order**
- d) in order

19. What is the time complexity of Dijkstra's algorithm?

- a) $O(N)$
- b) $O(N^3)$
- c) **$O(N^2)$**
- d) $O(\log N)$

20. Bellmann ford algorithm provides solution for _____ problems.

- a) All pair shortest path
- b) Sorting
- c) Network flow
- d) **Single source shortest path**

Weekly Assignment

Q.1 Draw the complete undirected graphs on one, two, three, four and five vertices. Prove that the number of edges in an n vertex complete graph is $n(n-1)/2$.

Q.2 What are the different ways of representing a graph? Represent the following graph using those ways.

Q.3 What are the different ways of representing a graph? Represent some graphs using those ways.

Weekly Assignment

Q.4 Write an algorithm which does depth first search through an un-weighted connected graph. In an un-weighted graph, would breadth first search or depth first search or neither find a shortest path tree from some node? Why?

Q.5 Which are the two standard ways of traversing a graph? Explain them with an example of each.

Q.6 Explain Dijkstra's algorithm for finding the shortest path in a given graph.

Q.7 Write the short note on Tree Traversal.

Previous Year Questions

- Explain warshall's algorithm with the help of an example.

(AKTU SEM III 2019-20)

- Describe the Dijkstra algorithm to find the shortest path.

(AKTU SEM III 2019-20)

- How the graph can be represented in memory? Explain with suitable example.

(AKTU SEM III 2018-19)

- Explain in detail about the graph traversal techniques with suitable examples.

(AKTU SEM III 2018-19)

- Write algorithm for Floyd warshall algorithm, explain with a suitable example.

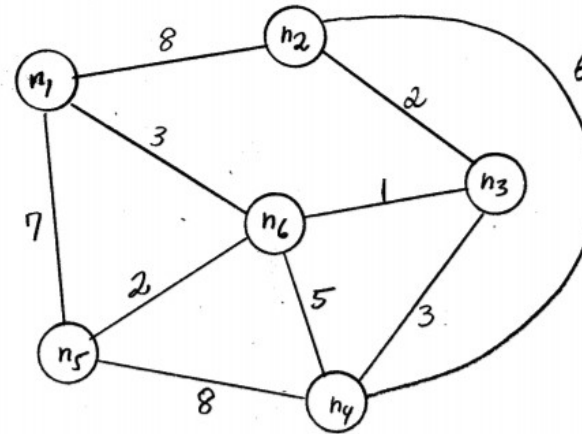
(AKTU SEM III 2017-18)

Expected Questions for University Exam

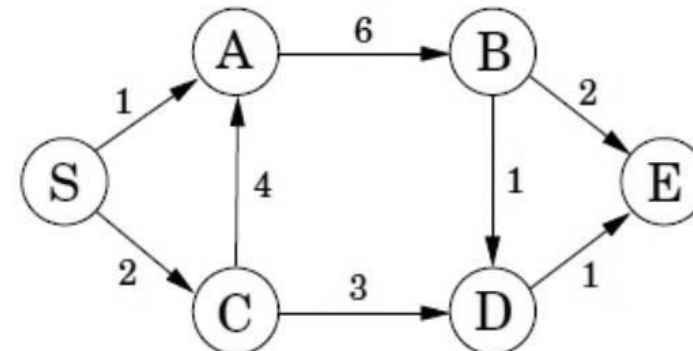
1. Define graph. How a graph is different from a tree?
2. Write Dijkstra algorithm for finding the shortest path from a source vertex.
3. Write the algorithm of Floyd Warshall. Also explain it with example.
4. Define connected graph and complete graph with example.
5. How the graph can be represented in memory? Explain with suitable example.
6. Explain Depth First Search Traversal in Graph with the help of an example.

Expected Questions for University Exam

7. Define Floyd Warshall Algorithm for all pair shortest path and apply the same on following graph:



8. Considering 'S' as source vertex, Apply the Dijkstra's shortest path algorithm on the following graph:



Summary

- File Organization refers to the logical relationships among various records that constitute the file, particularly with respect to the means of identification and access to any specific record.
- Graph theory is an exceptionally rich area for programmers and designers.
- Graphs can be used to solve some very complex problems, such as least cost routing, mapping, program analysis, and so on.
- Network devices, such as routers and switches, use graphs to calculate optimal routing for traffic.

References

- Aaron M. Tenenbaum, Yedidiah Langsam and Moshe J. Augenstein, “Data Structures Using C and C++”, PHI Learning Private Limited, Delhi India
- Horowitz and Sahani, “Fundamentals of Data Structures”, Galgotia Publications Pvt Ltd Delhi India.
- Lipschutz, “Data Structures” Schaum’s Outline Series, Tata McGraw-hill Education (India) Pvt. Ltd.
- Thareja, “Data Structure Using C” Oxford Higher Education.
- AK Sharma, “Data Structure Using C”, Pearson Education India.
- Michael T. Goodrich, Roberto Tamassia, David M. Mount “Data Structures and Algorithms in C++”, Wiley India.
- . P. S. Deshpandey, “C and Data structure”, Wiley Dreamtech Publication.

Thank You