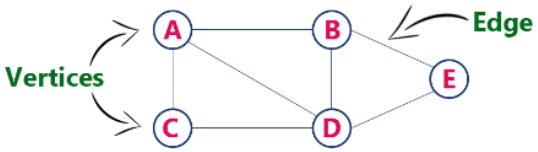
MODULE 5

GRAPHS

Graph: Graph is a collection of vertices and edges which connects vertices in the graph.
The notation G = (V, E) is used to represent a graph.
V = set of vertices
E = set of edges

Example:
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)}.



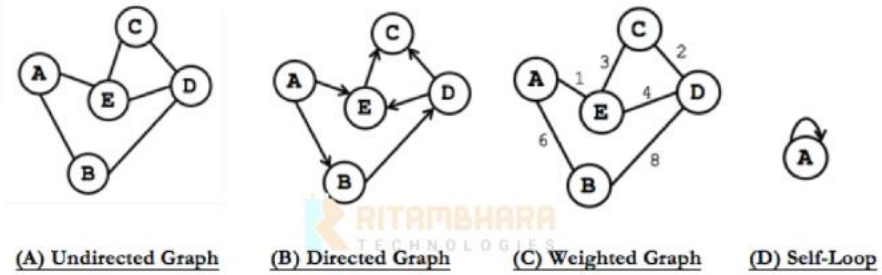
3.1 Graph Terminologies:

Vertex
A 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. In above example graph, there are 7 edges (i.e., (A,B), (A,C), (A,D), (B,D), (B,E), (C,D), (D,E)).

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

Directed Graph (Digraph)
A graph with only directed edges is said to be directed graph.



Weighted Graph: A Graph is termed weighted graph if all the edges in it are labeled with some weights.

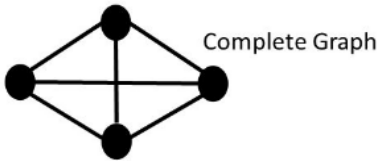
Adjacent vertices: If there is an edge between vertices A and B then both A and B are said to be adjacent.

Self loop: If there is an edge whose starting and end vertices are same, that is, (v_i, v_i) is an edge, then it is called a self-loop.

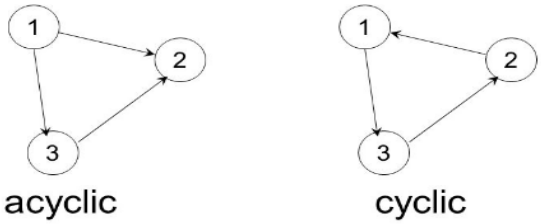
Parallel edges: If there is more than one edge between same pair of vertices, then they are known as parallel edges.

Simple graph: A graph if it does not have any self loop or parallel edges is called a simple graph.

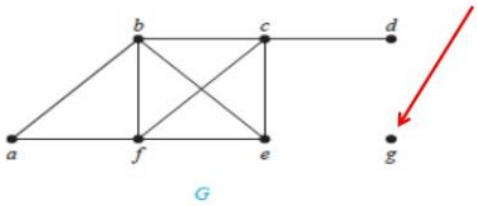
Complete graph: A graph is said to be complete if there are edges from any vertex to all other vertices.



Acyclic Graph: If there is a path containing one or more edges which starts from a vertex v_i and terminates into the same vertex is called cyclic graph. If the graph does not have any cycle is called acyclic graph.



Isolated vertex: A vertex is isolated if there is no edge connected from any other vertex to the vertex.



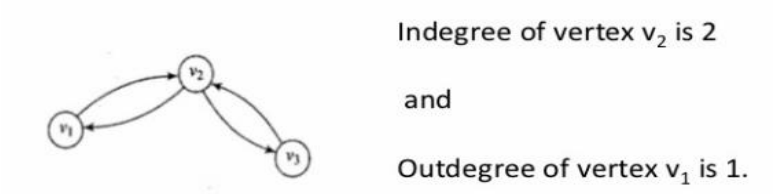
Degree of vertex: Total number of edges connected to a vertex is said to be degree of that vertex.

Indegree

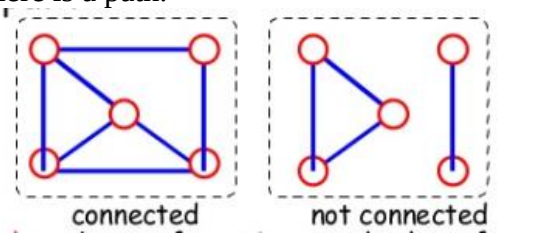
Total number of incoming edges connected to a vertex is said to be indegree of that vertex.

Outdegree

Total number of outgoing edges connected to a vertex is said to be outdegree of that vertex.



Pendant vertex: A vertex v_i is pendant if its indegree = 1 and outdegree = 0.
Connected graph: A graph G is said to be connected if for every pair of distinct vertices v_i, v_j in Graph G, there is a path.

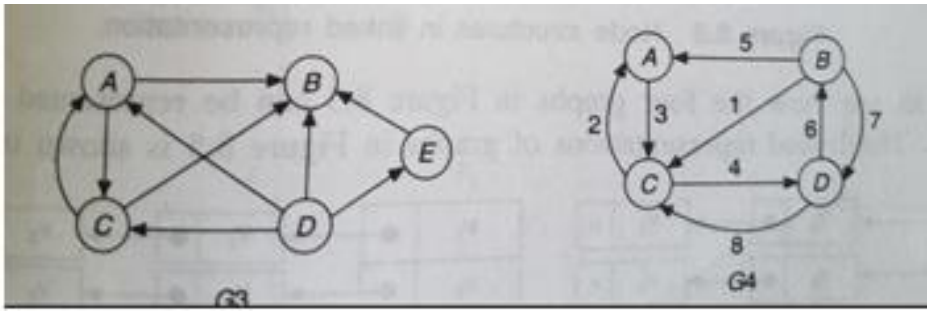


3.2 Representation of graph:

A graph can be represented in many ways. Some of the representations are:

- Set representation
- Linked representation
- Sequential or matrix representation

Set representation: This is one of the straightforward methods of representing a graph. With this method, two sets are maintained: (i) V, the set of vertices and (ii) the set of edges.



Graph G3

$V(G3) = \{A, B, C, D, E\}$

$E(G3) = \{(A, B), (A, C), (C, B), (C, A), (D, A), (D, B), (D, C), (D, E), (E, B)\}$

Graph G4

$V(G4) = \{A, B, C, D\}$

$E(G4) = \{(3, A, C), (5, B, A), (1, B, C), (7, B, D), (2, C, A), (4, C, D), (6, D, B), (8, D, C)\}$

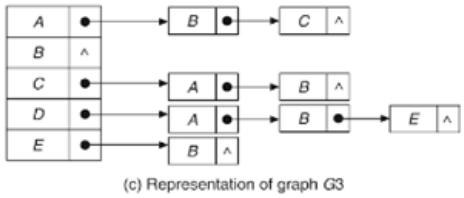
allow to store

Linked representation: It is another space-saving way of graph representation. In this representation, there are types of node structures.

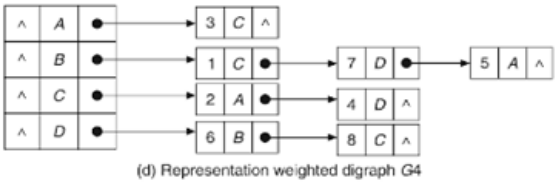
Node structure for non-weighted graph

Node structure for weighted graph

Linked representation of graphs is shown below:



(c) Representation of graph *G3*

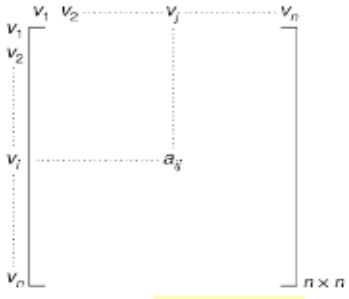


(d) Representation weighted digraph *G4*

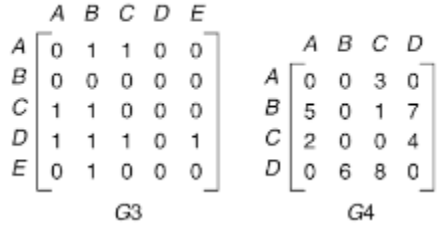
Matrix representation:

It is the most useful way of representing any graph. This representation uses a square matrix of order n x n, where n is number of vertices in the graph. This matrix is also known as adjacency matrix.

General representation is shown below:



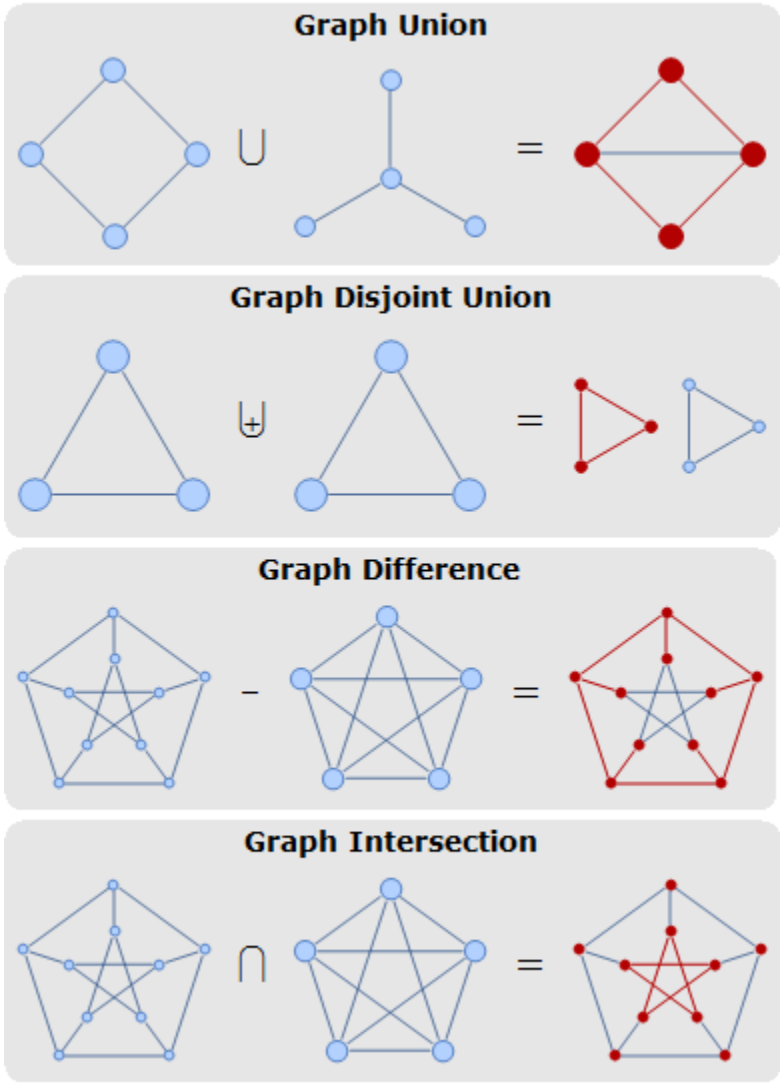
Adjacency matrix representation of graphs G1, G2, G3 and G4 is shown below:



G3 *G4*

GRAPH OPERATIONS

Elementary operations or editing operations, which are also known as graph edit operations, create a new graph from one initial one by a simple local change, such as addition or deletion of a vertex or of an edge, merging and splitting of vertices, edge contraction, etc.



3.6 Shortest path problem

The shortest path problem is about finding a path between 2 vertices in a graph such that the total sum of the edges weights is minimum. The following three algorithms are discussed in shortest path problem.

- Warshall's algorithm
- Floyd's algorithm
- Dijkstra's algorithm

3.7 Dijkstra's algorithm

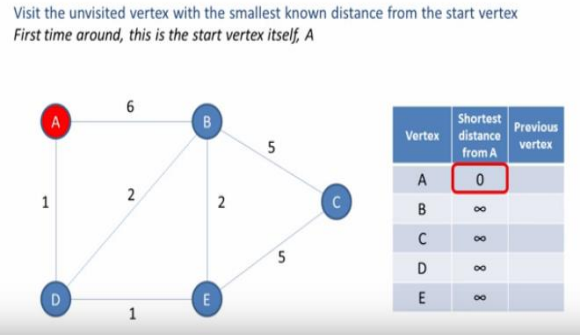
Dijkstra's algorithm is an iterative algorithm that finds the shortest path from source vertex to all other vertices in the graph.

Algorithm:

Step 1: Initially, distance of start vertex =0
Step 2: Distance of all other vertices from start vertex = infinity (∞)
Step 3: visit the unvisited vertex with the smallest known distance from the start vertex
Step 4: For the current vertex, examine its unvisited neighbor's
Step 5: For the current vertex, calculate the distance each neighbor from start vertex
Step 6: If the calculated distance of vertex is less than the known distance, update the shortest distance
Step 7: update the previous vertex for each of the updated distances.
Step 8: Add current vertex to the list of visited vertices.
Step 9: Repeat steps from 3 to 8 until all vertices visited.

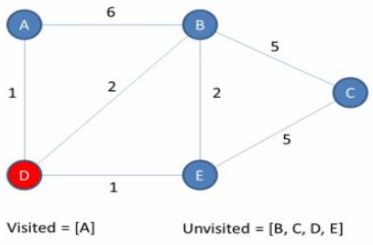
Example:

Step 1: consider start vertex A and distance of start vertex =0 and Distance of all other vertices from start vertex = infinity (∞)



Vertex	Shortest distance from A	Previous vertex
A	0	
B	∞	
C	∞	
D	∞	
E	∞	

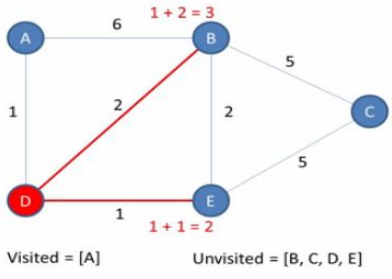
Step 2: For vertex A, calculate the distance of each neighbor.



Vertex	Shortest distance from A	Previous vertex
A	0	
B	6	A
C	∞	
D	1	A
E	∞	

Step 3: Vertex D has shortest distance from A. so, consider D.

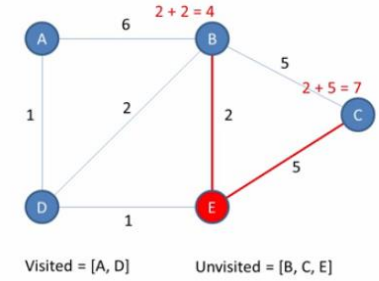
For the current vertex D, calculate the distance of each neighbor from the start vertex.



Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	∞	
D	1	A
E	2	D

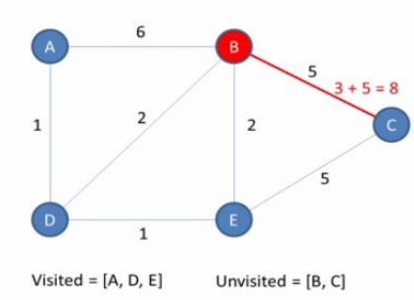
Step 4: Vertex E has shortest distance from A. So, consider E.

For the current vertex E, calculate the distance of each neighbor from the start vertex.



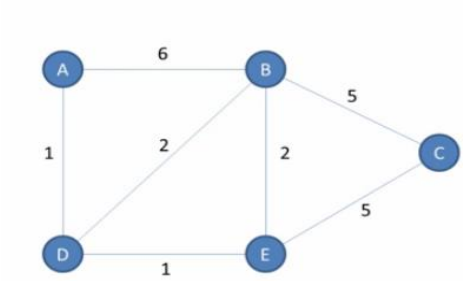
Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	∞	
D	1	A
E	2	D

Step 5: Vertex B has shortest Distance from A. So, consider B.
For the vertex B, calculate the distance of each neighbor from the start vertex. The calculated value from B to C is 8, which is greater than the value from E to C is 7. So, there is no need to update distance to C.



Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Step 6: Now, visit C and currently there are no unvisited neighbors from C.



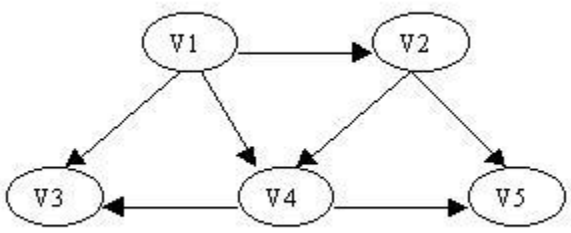
Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

3.8 Topological sorting: This is an ordering of the vertices in a directed acyclic graph, such that if there is a path from **u** to **v**, then **v** appears after **u** in the ordering.

Algorithm

1. Compute the indegrees of all vertices
2. Find a vertex **U** with indegree 0 and print it (store it in the ordering)
3. If there is no such vertex then there is a cycle and the vertices cannot be ordered and terminate.
4. Else
5. Remove **U** and all its edges **(U,V)** from the graph.
6. Update the indegrees of the remaining vertices.
7. Repeat steps 2 through 4 while there are vertices to be processed.

Example



Sol:

Compute the indegrees:

V1: 0
V2: 1
V3: 2
V4: 2
V5: 2

Find a vertex with indegree 0: V1

Output V1 , remove V1 and update the indegrees:

Sorted: V1

Updated indegrees:

V2: 0
V3: 1
V4: 1
V5: 2

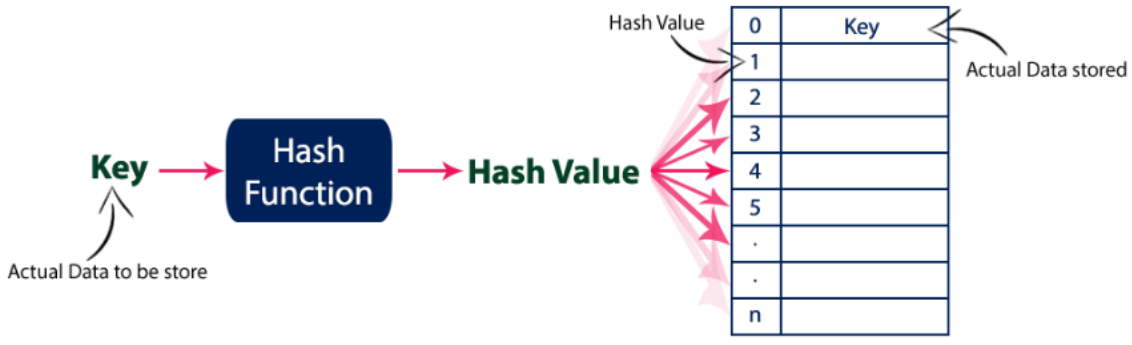
2. Repeat the process until all the vertices are processed.
3. Final topological order on a directed acyclic graph is
   One possible sorting: V1, V2, V4, V3, V5
   Another sorting: V1, V2, V4, V5, V3

Minimum Spanning Trees – Kruskal's and Prim's algorithms.

HASH TABLES

Tables: Hash tables:

- Hashing is the process of indexing and retrieving element (data) in a data structure to provide faster way of finding the element using the hash key.
- Hash key is a value which provides the index value where the actual data is likely to store in the data structure.
- In this data structure, we use a concept called Hash table to store data.
- All the data values are inserted into the hash table based on the hash key value.
- Hash key value is used to map the data with index in the hash table.
- And the hash key is generated for every data using a hash function.
- That means every entry in the hash table is based on the key value generated using a hash function.



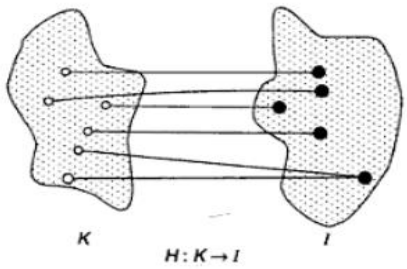
To achieve a good hashing mechanism, It is important to have a good hash function with the following basic requirements:

1. Easy to compute
2. Uniform distribution
3. Less collision

STATIC HASHING

Hashing Techniques:

The main idea behind any Hashing technique is to find a one-to-one correspondence between a index value and an index in the hash table where the key value can be placed.
In the following figure, K denotes a set of key values, I denote a range of indices and H denotes the mapping function from K to I.



The following are the hashing techniques:

1. Division method
2. Mid suqare method
3. Folding method
4. Digit Analysis method

Division Hash method:

The key K is divided by some number m and the remainder is used as the Hash address of K.

$$h(k) = k \bmod m.$$

This gives indexes in the range 0 to m-1so the Hash table should be of size m.

This is an example of uniform hash function if value of m is chosen carefully.

H(x) = x mod 10

21, 56, 72, 39, 48

986, 13

H(21) = 21 mod 10 = 1

H(56) = 56 mod 10 = 6

H(72) = 2

H(39) = 9

H(48) = 48 mod 10 = 8

0	
1	21
2	72
3	
4	
5	
6	56
7	
8	48
9	39

Folding method:

- The key K is partitioned into a number of parts, each of which has the same length as the required address with the possible exception of the last part.
- The parts are then added together, ignoring the final carry, to form an address
- There are two types in folding method. They are:
- Fold-shift hashing
- Fold-boundary hashing

Fold-shift hashing:

Key value is divided into parts whose size matches the size of the required address.

Example: K= 123456789 (K is divided into three equal parts and added)

123 + 456 + 789 = 1368

Remove 1

Therefore, h(k) = 368

Fold-boundary hashing:

Left and right number are folded on a fixed boundary between them.

Example: K = 123456789 (Here, 123 and 789 are reversed).

321 + 456 + 987 = 1764

Remove 1

Therefore, h(k) = 764

Mid-Square method:

- The key K is multiplied by itself and the address is obtained by selecting an appropriate number of digits from the middle of the square.
- The number of digits selected depends on the size of the table.
- Example: If K= 123456
  - then K^2 = 15241383936
- If three digit addresses is required, positions 5 to 7 is chosen giving address 138.

Digit Analysis method

- In the digit analysis method, the index is formed by extracting and then manipulating specific digits from the key.
- For example, if our key is 1234567, we might select the digits in positions 2 through 4 yielding 234
- The manipulations can then take many forms:
  - Reversing the digits (432)

- Performing a circular shift to the right (423)
- Performing a circular shift to the left (342)
- Swapping each pair of digits (324)

DYNAMIC HASHING:

Motivation for Dynamic Hashing Traditional hashing schemes as described in the previous section are not ideal. This follows from the fact that one must statically allocate a portion of memory to hold the hash table. This hash table is used to point to the pages used to hold identifiers, or it may actually. hold the identifiers themselves. In either case, if the table is allocated to be as large as possible. then space can be wasted. If it is allocated to be too small, then when the data exceed the capacity of the hash table, the entire file must be restructured. a time-consuming process. The purpose of dynamic hashing (also referred to as extendible hashing) is to retain the fast retrieval time of conventional hashing while extending the technique so that it can accommodate dynamically increasing and decreasing tile size without penalty.