### Graph

* + - A graph G consist of a non-empty set V called the set of nodes (points, vertices) of the graph, a set E which is the set of edges and a mapping from the set of edges E to a set of pairs of elements of V.
    - It is also convenient to write a graph as G=(V,E).
    - Notice that definition of graph implies that to every edge of a graph G, we can associate a pair of nodes of the graph. If an edge X Є E is thus associated with a pair of nodes (u,v) where u, v Є V then we says that edge x connect u and v.

### Adjacent Nodes

* + - Any two nodes which are connected by an edge in a graph are called adjacent node.

### Directed & Undirected Edge

* + - In a graph G=(V,E) an edge which is directed from one end to another end is called a directed edge, while the edge which has no specific direction is called undirected edge.

### Directed graph (Digraph)

* + - A graph in which every edge is directed is called directed graph or digraph.

### Undirected graph

* + - A graph in which every edge is undirected is called undirected graph.

### Mixed Graph

* + - If some of the edges are directed and some are undirected in graph then the graph is called mixed graph.

### Loop (Sling)

* + - An edge of a graph which joins a node to itself is called a loop (sling).

### Parallel Edges

* + - In some directed as well as undirected graphs, we may have certain pairs of nodes joined by more than one edges, such edges are called Parallel edges.

### Multigraph

* + - Any graph which contains some parallel edges is called multigraph.

### Weighted Graph

* + - A graph in which weights are assigned to every edge is called weighted graph.

### Isolated Node

* + - In a graph a node which is not adjacent to any other node is called isolated node.

### Null Graph

* + - A graph containing only isolated nodes are called null graph. In other words set of edges in null graph is empty.

### Path of Graph

* + - Let G=(V, E) be a simple digraph such that the terminal node of any edge in the sequence is the initial node of the edge, if any appearing next in the sequence defined as path of the graph.

### Length of Path

* + - The number of edges appearing in the sequence of the path is called length of path.

### Degree of vertex

* + - The no of edges which have V as their terminal node is call as indegree of node V
    - The no of edges which have V as their initial node is call as outdegree of node V
    - Sum of indegree and outdegree of node V is called its Total Degree or Degree of vertex.

### Simple Path (Edge Simple)

* + - A path in a diagraph in which the edges are distinct is called simple path or edge simple.

### Elementary Path (Node Simple)

* + - A path in which all the nodes through which it traverses are distinct is called elementary path.

### Cycle (Circuit)

* + - A path which originates and ends in the same node is called cycle (circuit).

### Directed Tree

## What is spanning tree?

* A Spanning tree of a graph is an undirected tree consisting of only those edges necessary to connect all the nodes in the original graph
* A spanning tree has the properties that
  + For any pair of nodes there exists only one path between them
  + Insertion of any edge to a spanning tree forms a unique cycle
* The particular Spanning for a graph depends on the criteria used to generate it.
* If DFS search is use, those edges traversed by the algorithm forms the edges of tree, referred to as Depth First Spanning Tree.
* If BFS Search is used, the spanning tree is formed from those edges traversed during the search, producing Breadth First Search Spanning tree.

V0

V1

V2

V3

V4

V5

V6

V7

V0

V1

V2

V3

V4

V5

V6

V7

V0

V1

V2

V3

V4

V5

V6

V7

**DFS Spanning Tree BFS Spanning Tree**

## Consider the graph shown in Fig Find depth-first and breadth first traversals of this graph starting at A

**A**

**B**

**C**

**E**

**D**

**F**

**A**

**B**

**C**

**E**

**D**

**F**

**A**

**B**

**C**

**E**

**D**

**F**

**DFS : A B D C F E BFS : A B C D F E**

## Define spanning tree and minimum spanning tree. Find the minimum spanning tree of the graph shown in Fig.

**Using Prim’s Algorithm:**

**A**

4

5

**B**

3

**E**

6

2

5

6

7

**C**

1

**D**

Let X be the set of nodes explored, initially X = { A }

~~A – B | 4~~ A – E | 5 A – C | 6 A – D | 6 ~~B – E | 3~~ ~~B – C | 2~~ C – E | 6 ~~C – D | 1~~ D – E | 7

|  |  |
| --- | --- |
| **Step 1:** Taking minimum weight edge of all Adjacent edges of **X = { A }**  4 **A**  **B X = { A , B }** | **Step 2:** Taking minimum weight edge of all Adjacent edges of **X = { A , B }**  4 **A**  **B**  **X = { A , B , C }**  2  **C** |
| **Step 3:** Taking minimum weight edge of all Adjacent edges of **X = { A , B , C }**  4 **A**  **B**  **X = { A , B , C, D }**  2  **C** 1 **D** | **Step 4:** Taking minimum weight edge of all Adjacent edges of **X = { A , B , C , D }**  4 **A**  **B** 3 **E**  2  **C** 1 **D**  **X = { A , B , C, D, E }** |

All nodes of graph are there with set X, so we obtained minimum spanning tree of cost**: 4 + 2 + 1 + 3 = 10**

# Using Kruskal’s Algorithm

**A**

4

5

**B**

3

**E**

6

2

5

6

7

**C**

1

**D**

**Step 1:** Taking min edge (C,D)

**Step 2:** Taking next min edge (B,C)

**Step 3:** Taking next min edge (B,E)

**B**

**B**

3

**E**

2

2

**C**

1

**D**

**C**

1

**D**

**C**

1

**D**

**Step 4:** Taking next min edge (A,B)

4

**A**

**Step 5:** Taking next min edge (A,E) it forms cycle so do not consider

**Step 6:** Taking next min edge (C,E) it forms cycle so do not consider **Step 7:** Taking next min edge (A,D) it forms cycle so do not consider **Step 8:** Taking next min edge (A,C) it forms cycle so do not consider

**Step 9:** Taking next min edge (E,D) it forms cycle so do not consider

**B**

3

**E**

2

**C**

1

**D**

All edges of graph has been visited,

so we obtained minimum spanning tree of cost**: 4 + 2 + 1 + 3 = 10**

## Give example and applications of directed and undirected graphs. Find the adjacency matrix for the graph shown in Fig.

### Adjacency matrix for the given graph

**1**

**6**

**2**

**3**

**4**

**5**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 0 | **1** | 0 | 0 | 0 | 0 |
| **2** | 0 | 0 | 0 | 1 | 0 | 0 |
| **3** | **1** | 0 | 0 | 0 | 0 | 0 |
| **4** | 0 | 0 | **1** | 0 | **1** | 0 |
| **5** | 0 | 0 | **1** | 0 | 0 | **1** |
| **6** | **1** | 0 | **1** | 0 | 0 | 0 |

###### Applications of graph:

* Electronic Circuits
  + Printed Circuit Board
  + Integrated Circuit
* Transportation networks
  + Highway networks

Modeling a road network with vertexes as towns and edge costs as distances.

* + Water Supply networks

Modeling a water supply network. A cost might relate to current or a function of capacity and length. As water flows in only 1 direction, from higher to lower pressure connections or downhill, such a network is inherently an acyclic directed graph.

* + Flight network

Minimizing the cost and time taken for air travel when direct flights don't exist between starting and ending airports.

* Computer networks
  + Local Area Network
  + Internet

Dynamically modeling the status of a set of routes by which traffic might be directed over the Internet.

* + Web

Using a directed graph to map the links between pages within a website and to analyze ease of navigation between different parts of the site.

* Databases
  + Entity Relationship Diagram

## Apply Dijkstra’s algorithm to find shortest path between vertex A and

**vertex F5 for the graph shown in Fig.**

**1**

**B**

**2**

**D**

**6**

**A**

**4**

**1**

**7**

**F**

**3**

**C**

**2**

**5**

**E**

**Step 1:** Traverse all adjacent node of A

**1 ∞**

**∞**



**2**

**Step 2:** Traverse all adjacent node of B

**1 3**



**2**

**0 A**

**1 B**

**4**

**1**

**3**

**3 C 5**

**D 6**

**7 F ∞**

**2**

###### E

**∞**

**0 A**

**1 B**

**4**

**1**

**3**

**3 C 5**

**D 6**

**7 F ∞**

**2**

###### E

**5**

**Step 3:** Traverse all adjacent node of C

**1 3**

**2**

**Step 4:** Traverse all adjacent node of D

**1 3**



**2**



**0 A**

**1 B**

**4**

**1**

**3**

**3 C 5**

**D 6**

**7 F ∞**

**2**

###### E

**5**

**0 A**

**1 B**

**4**

**1**

**3**

**3 C 5**

**D 6**

**7 F 9**

**2**

###### E

**5**

**Step 5:** Traverse all adjacent node of E

**1 3**



**0 A**

**2**

**1 B**

**4**

**1**

**3**

**3 C 5**

**D 6**

**7 F 7**

**2**

###### E

**5**

* Shortest path from node **A to F** is :

**A – B – E – F** as shown in step 5

* Length of path is **7**