# 14. GRAPH AND GRAPH MODELS

**Prepared By Mrs. A. A. Daptardar**

The previous part brought forth the different tools for reasoning, proofing and problem solving. In this part, we will study the discrete structures that form the basis of formulating many a real-life problem.

The two discrete structures that we will cover are graphs and trees. A graph is a set of points, called nodes or vertices, which are interconnected by a set of lines called edges. The study of graphs, or **graph theory** is an important part of a number of disciplines in the fields of mathematics, engineering and computer science.

## What is a Graph?

**Definition:** A graph (denoted as G = (V, E)) consists of a non-empty set of vertices or nodes V and a set of edges E.

**Example:** Let us consider, a Graph is G = (V, E) where V = {a, b, c, d} and E = {{a, b}, {a, c}, {b, c},{c, d}}

**Even and Odd Vertex:** If the degree of a vertex is even, the vertex is called an even vertex and if the degree of a vertex is odd, the vertex is called an odd vertex.

**Degree of a Vertex:** The degree of a vertex V of a graph G (denoted by deg (V)) is the number of edges incident with the vertex V.

| Vertex | Degree | Even / Odd |
|---|---|---|
| a | 2 | even |
| b | 2 | even |
| c | 3 | odd |
| d | 1 | odd |

**Degree of a Graph:** The degree of a graph is the largest vertex degree of that graph. For the above graph the degree of the graph is 3.

**The Handshaking Lemma:** In a graph, the sum of all the degrees of vertices is equal to twice the number of edges.

## Types of Graphs

There are different types of graphs, which we will learn in the following section.

### Null Graph

A null graph has no edges. The null graph of n vertices is denoted by $N_n$

### Simple Graph

A graph is called simple graph/strict graph if the graph is undirected and does not contain any loops or multiple edges.

### Multi-Graph

If in a graph multiple edges between the same set of vertices are allowed, it is called Multi-graph.

### Directed and Undirected Graph

A graph G = (V, E) is called a directed graph if the edge set is made of ordered vertex pair and a graph is called undirected if the edge set is made of unordered vertex pair.

### Connected and Disconnected Graph

A graph is connected if any two vertices of the graph are connected by a path and a graph is disconnected if at least two vertices of the graph are not connected by a path. If a graph G is unconnected, then every maximal connected subgraph of G is called a connected component of the graph G.

### Regular Graph

A graph is regular if all the vertices of the graph have the same degree. In a regular graph G of degree r, the degree of each vertex of G is r.

### Complete Graph

A graph is called complete graph if every two vertices pair are joined by exactly one edge. The complete graph with n vertices is denoted by $K_n$

### Cycle Graph

If a graph consists of a single cycle, it is called cycle graph. The cycle graph with n vertices is denoted by $C_n$

### Bipartite Graph

If the vertex-set of a graph G can be split into two sets in such a way that each edge of the graph joins a vertex in first set to a vertex in second set, then the graph G is called a bipartite graph. A graph G is bipartite if and only if all closed walks in G are of even length or all cycles in G are of even length.

### Complete Bipartite Graph

A complete bipartite graph is a bipartite graph in which each vertex in the first set is joined to every single vertex in the second set. The complete bipartite graph is denoted by $K_{r,s}$ where the graph G contains x vertices in the first set and y vertices in the second set.

## Representation of Graphs

There are mainly two ways to represent a graph:

## Adjacency Matrix

An Adjacency Matrix A[V][V] is a 2D array of size V×V where V is the number of vertices in a undirected graph. If there is an edge between $V_x$ to $V_y$ then the value of $A[V_x][V_y]=1$ and $A[V_y][V_x]=1$, otherwise the value will be zero. And for a directed graph, if there is an edge between $V_x$ to $V_y$, then the value of $A[V_x][V_y]=1$, otherwise the value will be zero.

### Adjacency Matrix of an Undirected Graph

Let us consider the following undirected graph and construct the adjacency matrix:

Adjacency matrix of the above undirected graph will be:

|   | **a** | **b** | **c** | **d** |
|---|---|---|---|---|
| **a** | 0 | 1 | 1 | 0 |
| **b** | 1 | 0 | 1 | 0 |
| **c** | 1 | 1 | 0 | 1 |
| **d** | 0 | 0 | 1 | 0 |

## Adjacency Matrix of a Directed Graph

Let us consider the following directed graph and construct its adjacency matrix:

Adjacency matrix of the above directed graph will be:

|   | **a** | **b** | **c** | **d** |
|---|---|---|---|---|
| **a** | 0 | 1 | 1 | 0 |
| **b** | 0 | 0 | 1 | 0 |
| **c** | 0 | 0 | 0 | 1 |
| **d** | 0 | 0 | 0 | 0 |

## Adjacency List

In adjacency list, an array (A[V]) of linked lists is used to represent the graph G with V number of vertices. An entry A[$V_x$] represents the linked list of vertices adjacent to the V**x-**th vertex. The adjacency list of the graph is as shown in the figure below:

## Planar vs. Non-planar graph

**Planar graph:** A graph G is called a planar graph if it can be drawn in a plane without any edges crossed. If we draw graph in the plane without edge crossing, it is called embedding the graph in the plane.

**Non-planar graph:** A graph is non-planar if it cannot be drawn in a plane without graph edges crossing.

## Isomorphism

If two graphs G and H contain the same number of vertices connected in the same way, they are called isomorphic graphs (denoted by G≅H).

It is easier to check non-isomorphism than isomorphism. If any of these following conditions occurs, then two graphs are non-isomorphic:

- The number of connected components are
- different Vertex-set cardinalities are different
- Edge-set cardinalities are different

### Example

The following graphs are isomorphic:

## Homomorphism

A homomorphism is an isomorphism if it is a bijective mapping. Homomorphism always preserves edges and connectedness of a graph. The compositions of homomorphisms are also homomorphisms. To find out if there exists any homomorphic graph of another graph is a NP-complete problem.

## Euler Graphs

A connected graph G is called an Euler graph, if there is a closed trail which includes every edge of the graph G. An Euler path is a path that uses every edge of a graph exactly once. An Euler path starts and ends at different vertices.

An Euler circuit is a circuit that uses every edge of a graph exactly once. An Euler circuit always starts and ends at the same vertex. A connected graph G is an Euler graph if and only if all vertices of G are of even degree, and a connected graph G is Eulerian if and only if its edge set can be decomposed into cycles.

The above graph is an Euler graph as “a 1 b 2 c 3 d 4 e 5 c 6 f 7 g” covers all the edges of the graph.

## Hamiltonian Graphs

A connected graph G is called Hamiltonian graph if there is a cycle which includes every vertex of G and the cycle is called Hamiltonian cycle. Hamiltonian walk in graph G is a walk that passes through each vertex exactly once.

If G is a simple graph with n vertices, where $n \geq 3$ If $\deg(v) \geq 1/2\ n$ for each vertex v, then the graph G is Hamiltonian graph. This is called **Dirac's Theorem**.

If G is a simple graph with n vertices, where $n \geq 2$ if $\deg(x) + \deg(y) \geq n$ for each pair of non-adjacent vertices x and y, then the graph G is Hamiltonian graph. This is called **Ore's theorem**.

# 15. TREES

**Tree** is a discrete structure that represents hierarchical relationships between individual elements or nodes. A tree in which a parent has no more than two children is called a binary tree.

## Tree and its Properties

**Definition:** A Tree is a connected acyclic graph. There is a unique path between every pair of vertices in G. A tree with N number of vertices contains (N-1) number of edges. The vertex which is of 0 degree is called root of the tree. The vertex which is of 1 degree is called leaf node of the tree and the degree of an internal node is at least 2.

**Example:** The following is an example of a tree:

## Centers and Bi-Centers of a Tree

The center of a tree is a vertex with minimal eccentricity. The eccentricity of a vertex X in a tree G is the maximum distance between the vertex X and any other vertex of the tree. The maximum eccentricity is the tree diameter. If a tree has only one center, it is called Central Tree and if a tree has only more than one centers, it is called Bi-central Tree. Every tree is either central or bi-central.

### Algorithm to find centers and bi-centers of a tree

**Step 1:** Remove all the vertices of degree 1 from the given tree and also remove their incident edges.

**Step 2:** Repeat step 1 until either a single vertex or two vertices joined by an edge is left. If a single vertex is left then it is the center of the tree and if two vertices joined by an edge is left then it is the bi-center of the tree.

## Problem 1

Find out the center/bi-center of the following tree:

*Tree T1*

## Solution

At first, we will remove all vertices of degree 1 and also remove their incident edges and get the following tree:

Again, we will remove all vertices of degree 1 and also remove their incident edges and get the following tree:

Finally we got a single vertex ‘c’ and we stop the algorithm. As there is single vertex, this tree has one center ‘c’ and the tree is a central tree.

## Problem 2

Find out the center/bi-center of the following tree:

*A tree T2*

## Solution

At first, we will remove all vertices of degree 1 and also remove their incident edges and get the following tree:

Again, we will remove all vertices of degree 1 and also remove their incident edges and get the following tree:

Finally, we got two vertices 'c' and 'd' left, hence we stop the algorithm. As two vertices joined by an edge is left, this tree has bi-center 'cd' and the tree is bi-central.

## Labeled Trees

**Definition:** A labeled tree is a tree the vertices of which are assigned unique numbers from 1 to n. We can count such trees for small values of n by hand so as to conjecture a general formula. The number of labeled trees of n number of vertices is $n^{n-2}$. Two labelled trees are isomorphic if their graphs are isomorphic and the corresponding points of the two trees have the same labels.

### Example

*A labeled tree with two vertices*

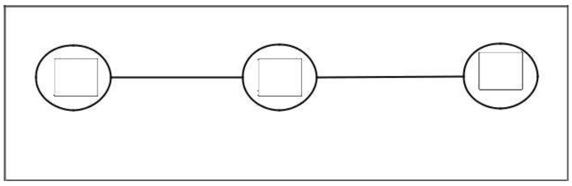
*Three possible labeled tree with three vertices*

## Unlabeled trees

**Definition:** An unlabeled tree is a tree the vertices of which are not assigned any numbers. The number of labeled trees of n number of vertices is (2n)! / (n+1)!n!

### Example

*An unlabeled tree with two vertices*


*An unlabeled tree with three vertices*

## Rooted Tree

A rooted tree G is a connected acyclic graph with a special node that is called the root of the tree and every edge directly or indirectly originates from the root. An ordered rooted tree is a rooted tree where the children of each internal vertex are ordered. If every internal vertex of a rooted tree has not more than m children, it is called an m-ary tree. If every internal vertex of a rooted tree has exactly m children, it is called a full m-ary tree. If m = 2, the rooted tree is called a binary tree.

## Binary Search Tree

Binary Search tree is a binary tree which satisfies the following property:

- X in left sub-tree of vertex V, Value(X) ≤ Value (V)
- Y in right sub-tree of vertex V, Value(Y) ≥ Value (V)

So, the value of all the vertices of the left sub-tree of an internal node V are less than or equal to V and the value of all the vertices of the right sub-tree of the internal node V are greater than or equal to V. The number of links from the root node to the deepest node is the height of the Binary Search Tree.

### Example

### Algorithm to search for a key in BST

```
BST_Search(x, k)

if ( x = NIL or k = Value[x] )
        return x;
if ( k < Value[x])
        return BST_Search (left[x], k);
else
        return BST_Search (right[x], k)
```

### Complexity of Binary search tree

| | **Average Case** | **Worst case** |
|---|---|---|
| **Space Complexity** | O(n) | O(n) |
| **Search Complexity** | O(log n) | O(n) |
| **Insertion Complexity** | O(log n) | O(n) |
| **Deletion Complexity** | O(log n) | O(n) |

A spanning tree of a connected undirected graph G is a tree that minimally includes all of the vertices of G. A graph may have many spanning trees.

## Example

## Minimum Spanning Tree

A spanning tree with assigned weight less than or equal to the weight of every possible spanning tree of a weighted, connected and undirected graph G, it is called minimum spanning tree (MST). The weight of a spanning tree is the sum of all the weights assigned to each edge of the spanning tree.

### Example

## Kruskal's Algorithm

Kruskal's algorithm is a greedy algorithm that finds a minimum spanning tree for a connected weighted graph. It finds a tree of that graph which includes every vertex and the total weight of all the edges in the tree is less than or equal to every possible spanning tree.

### Algorithm

**Step 1:** Arrange all the edges of the given graph G (V,E) in non-decreasing order as per their edge weight.

**Step 2:** Choose the smallest weighted edge from the graph and check if it forms a cycle with the spanning tree formed so far.

**Step 3:** If there is no cycle, include this edge to the spanning tree else discard it.

**Step 4:** Repeat Step 2 and Step 3 until (V-1) number of edges are left in the spanning tree.

## Prim's Algorithm

Prim's algorithm, discovered in 1930 by mathematicians, Vojtech Jarnik and Robert C. Prim, is a greedy algorithm that finds a minimum spanning tree for a connected weighted graph. It finds a tree of that graph which includes every vertex and the total weight of all the edges in the tree is less than or equal to every possible spanning tree. Prim’s algorithm is faster on dense graphs.

## Algorithm

1. Create a vertex set V that keeps track of vertices already included in MST.

2. Assign a key value to all vertices in the graph. Initialize all key values as infinite. Assign key value as 0 for the first vertex so that it is picked first.

3. Pick a vertex ‘x’ that has minimum key value and is not in V.

4. Include the vertex U to the vertex set V.

5. Update the value of all adjacent vertices of x.

6. Repeat step 3 to step 5 until the vertex set V includes all the vertices of the graph.

## Problem

Suppose we want to find minimum spanning tree for the following graph G using Prim’s algorithm.