1. **Graph Representation**

   - Given an adjacency-list representation of a graph $G = (V, E)$, write an algorithm to obtain the adjacency-matrix representation of G. Also, obtain the time complexity of this algorithm.
   - Given an adjacency-matrix representation of a graph $G = (V, E)$, write an algorithm to obtain the adjacency-list representation of G. Also, obtain the time complexity of this algorithm.

2. **Degree of Vertices in the Graph**

   - Given an adjacency-list representation of a directed graph $G = (V, E)$, write an algorithm to obtain the in-degree and out- degree of all the vertices. Also, obtain the time complexity of this algorithm.
   - Given an adjacency-matrix representation of a directed graph $G = (V, E)$, write an algorithm to obtain the in-degree and out-degree of all the vertices. Also, obtain the time complexity of this algorithm.
   - Given an adjacency-list representation of an undirected graph $G = (V, E)$, write an algorithm to obtain the degree of all the vertices. Also, obtain the time complexity of this algorithm.
   - Given an adjacency-matrix representation of an undirected graph $G = (V, E)$, write an algorithm to obtain the degree of all the vertices. Also, obtain the time complexity of this algorithm.
   - Given a directed graph $G = (V, E)$, what is the sum of the in-degree and out-degree of all the vertices?
   - Given an undirected graph $G = (V, E)$, what is the sum of the degree of all the vertices?
   - What can be the maximum and minimum degree of a node in case of an undirected graph?
   - What can be the maximum and minimum degree of a node in case of a directed graph?

3. **Transpose of Graph**

   - Given an adjacency-list representation of a graph $G = (V, E)$, write an algorithm to obtain the adjacency-list representation of the transpose of G, *i.e.*, G^T. Also, obtain the time complexity of this algorithm.
   - Given an adjacency-matrix representation of a graph $G = (V, E)$, write an algorithm to obtain the adjacency-matrix representation of the transpose of G, *i.e.*, G^T. Also, obtain the time complexity of this algorithm.

4. **Connected Graph**

   - Given an undirected graph $G = (V, E)$. Write an algorithm to check whether the graph G is connected or not. Analyze the running time of your algorithm.
   - Given a directed graph $G = (V, E)$. Write an algorithm to check whether the graph G is strongly connected or not. Analyze the running time of your algorithm.

- Given a directed graph $G = (V, E)$. Write an algorithm to check whether the graph G is weakly connected or not. Analyze the running time of your algorithm.

5. **Graph Traversal**

   - Given a complete undirected graph $G = (V, E)$. Assume you start your DFS traversal from a vertex $v \in V$. In this case, obtain the number of –
     - Tree edges: $n - 1$
     - Back edges: $0 + 0 + 1 + 2 + 3 + \cdots + (n-2) = \frac{1}{2}(n-1)(n-2)$
     - Forward edges: 0
     - Cross edges: 0
   - Given a complete directed graph $G = (V, E)$. Assume you start your DFS traversal from a vertex $v \in V$. In this case, obtain the number of –
     - Tree edges: $n - 1$
     - Back edges: $0 + 1 + 2 + 3 + \cdots + (n-1) = \frac{1}{2}n(n-1)$
     - Forward edges: $(n-2) + (n-3) + \cdots + \cdots + 1 + 0 + 0 = \frac{1}{2}(n-1)(n-2)$
     - Cross edges: 0
   - Given a connected undirected graph $G = (V, E)$. Obtain the maximum number of cross edges in the DFS traversal of this graph G. Discuss your reasoning.
   - A complete undirected graph $G = (V, E)$ is always connected. **True** / **False**? Discuss reasoning.
   - A complete directed graph $G = (V, E)$ is always strongly connected. **True** / **False**? Discuss reasoning.
   - Why there is no cross edge in the DFS traversal of an undirected graph $G = (V, E)$?
   - Why presence of a back edge in the DFS traversal of a graph $G = (V, E)$ guarantee that there is a cycle?
   - DFS traversal of a directed graph $G = (V, E)$ divides the edges into four types – tree edge, back edge, forward edge, and cross edge. An edge (u, v) is a cross edge if $v.d < v.f < u.d < u.f$ where $u.d$ and $v.d$ are discovery time of vertices u and v. Similarly, $u.f$ and $v.f$ are finish time of vertices u and v. Why the edge (u, v) cannot be cross edge if $u.d < u.f < v.d < v.f$?

6. **Application of BFS and DFS**

   - Given an undirected graph $G = (V, E)$ where the edge weight of all the edges are 1. The diameter of this graph G is defined as the largest shortest path distance in the graph, *i.e.*, diameter $= \max_{u,v \in V} \delta(u, v)$ where $\delta(u, v)$ is the shortest distance between vertices u and v. Give an algorithm to compute the diameter, and analyze the running time of your algorithm.
   - Check whether a given graph $G = (V, E)$ is Bipartite or not.
   - Given an undirected connected graph $G = (V, E)$, check if the graph is 2-edge connected?

7. **Strongly Connected Components**

- Given a directed graph $G = (V, E)$. Obtain all the strongly connected components in this graph G *without using* DFS traversal. Write the algorithm for this and analyze the running time.
- What is the maximum and the minimum number of strongly connected components in a directed graph $G = (V, E)$ and why?
- Given a directed graph $G = (V, E)$ without self loop. Let the number of strongly connected component in this graph G is n where $n = |V|$. Obtain the minimum / maximum number of edges in graph G.
- Given a directed graph $G = (V, E)$ with self loop. Let the number of strongly connected component in this graph G is n where $n = |V|$. Obtain the minimum / maximum number of edges in graph G.
- Given a directed graph $G = (V, E)$. Suppose that G has strongly connected components $C_1, C_2, \ldots, C_k$. The component graph $G^{SCC} = (V^{SCC}, E^{SCC})$. The vertex set of this component graph V^{SCC} is $v_1, v_2, \ldots, v_k$, and it contains a vertex v_i for each strongly connected component C_i of G. There is an edge $(v_i, v_j) \in E^{SCC}$ if G contains an edge (x, y) for some $x \in C_i$ and some $y \in C_j$. The component graph G^{SCC} is a directed acyclic graph. Why?
- Let C and C' be distinct strongly connected components in directed graph $G = (V, E)$. Let $u.f$ represents the finishing time of a vertex $u \in V$. $f(C)$ and $f(C')$ are the latest finishing time of any vertex in C and C' respectively, *i.e.*, $f(C) = \max_{u \in C} \{u.f\}$ and $f(C') = \max_{v \in C'} \{v.f\}$. Suppose that there is an edge $(u, v) \in E$, where $u \in C$ and $v \in C'$. Then prove that $f(C) > f(C')$.

Algorithm 1 CONNECTED(G)

Input: An undirected graph $G = (V, E)$

Output: TRUE: If the graph is connected, FALSE: otherwise

for each vertex $u \in G.V$ **do**
    $L_u \leftarrow \text{DFS}(u)$ ▷ Apply DFS on u and store all the vertices reachable from u (including u) in L_u
    if $|L_u| < |G.V|$ **then** ▷ All the vertices are not reachable from u
        return FALSE ▷ Graph G is not connected
    end if
end for
return TRUE ▷ Graph G is connected

Algorithm 2 STRONGLY-CONNECTED(G)

Input: A directed graph $G = (V, E)$

Output: TRUE: If the graph is strongly connected, FALSE: otherwise

```
for each vertex u ∈ G.V do
    L_u ← DFS(u)                ▷ Apply DFS on u and store all the vertices reachable from u (including u) in L_u
    if |L_u| < |G.V| then       ▷ All the vertices are not reachable from u
        return FALSE            ▷ Graph G is not strongly connected
    end if
end for
return TRUE                     ▷ Graph G is strongly connected
```

Algorithm 3 WEAKLY-CONNECTED(G)

Input: A directed graph $G = (V, E)$

Output: TRUE: If the graph is weakly connected, FALSE: otherwise

```
Obtain the underlying undirected graph G′ from the directed graph G
for each vertex u ∈ G′.V do
    L_u ← DFS(u)                ▷ Apply DFS on u considering G′ and store all the vertices reachable from u (including u) in L_u
    if |L_u| < |G′.V| then      ▷ All the vertices are not reachable from u in G′
        return FALSE            ▷ Graph G is not weakly connected
    end if
end for
return TRUE                     ▷ Graph G is weakly connected
```

- Every connected graph is complete?
- Every complete graph is connected?
- Every strongly connected graph is weakly connected graph?
- Every weakly connected graph is strongly connected graph?