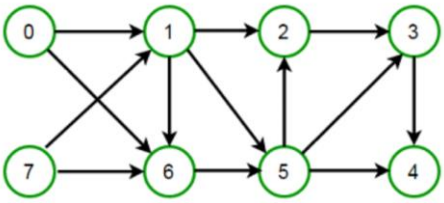
Assignment #4

1. **You can attempt this assignment in a group (max group Limit: 3)**
2. **Submit handwritten, soft copy in pdf format in google classroom and hard copy by 15th May 24.**
3. **Write the name and roll number of all the group members on the title page.**
4. **Attempt the questions in sequence starting from Q#1 and clearly mention Q# and part#. Start each question from a new page.**
5. **Plagiarism will be severely punished and a 50% deduction for not following instruction#4.**

Attempt any part of Q#1

Q#1(a): Given a directed graph in the form of adjacency list, find the total number of routes to reach the destination from a given source with exactly "m" edges. For example, consider the following graph:

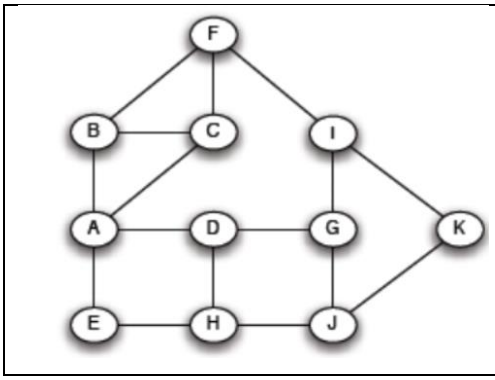


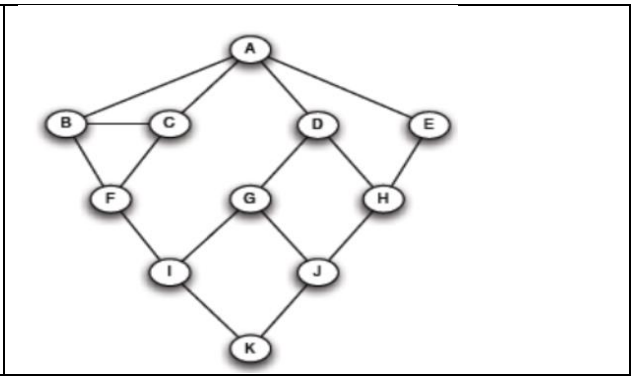
Let source = 0, destination = 3, number of edges 'm' = 4. The graph has 3 routes from source 0 to destination 3 with exactly 4 edges. The solution should return the total number of routes i.e., 3 in this case. Design an efficient algorithm for solving this problem and analyze its time complexity. Explain the main idea in plain English in maximum three to four lines.

Input Parameters: G, src, m, dest

Q#1(b): BFS can be used to solve the problem of single source shortest path for unweighted graph G(V, E). Also, BFS only finds one shortest path between "s" and "d", where "s" is the source vertex and "d" is any other vertex of the graph. But the shortest path may not be unique. Suppose we want to count the number of distinct shortest paths between "s" and for each "d" ϵ G.V. For example, in the graph given below if "A" is the source vertex then the number of distinct shortest paths for each pair are:

From A to B: 1	From A to C: 1	From A to D: 1	From A to E: 1	From A to F: 2
From A to G: 1	From A to H: 2	From A to I: 3	From A to J: 3	From A to K: 6





Design and explain a linear time algorithm to count the number of distinct shortest paths between "s" and all the other vertices in the graph. The graph is available in the form of an adjacency list.

Hint: you can use the tree structure of BFS to count the distinct shortest paths.

Attempt any one part of Q#2

You can assume that the weights of edges are available in "w" and w(u,v) will provide you the weight between any two vertices 'u' and 'v'

Q#2(a): Given a weighted undirected graph G(V, E) and its Minimum Spanning Tree T(V, E'). Both G(V,E) and T(V,E') are represented in the form of adjacency list, where E' represents the edges present in the MST. Assume that an edge (a, b) ∈ E' has been deleted, devise an algorithm to efficiently update the MST after the deletion of (a, b). Assume that all edge weights are distinct, and the graph remains connected after the deletion of edge (a, b). Explain the main idea of your algorithm along with an explanation of its time complexity in plain English as well.

Q#2(b): Given a weighted undirected graph G(V, E) and its Minimum Spanning Tree T(V, E'). Both G(V,E) and T(V,E') are represented in the form of adjacency list, where E' represents the edges present in the MST. Assume that an edge (a, b) ∈ E has been relaxed. There is a strong possibility that MST needs to be updated because of this edge relaxation. Devise an algorithm to efficiently update the MST after the relaxation of (a, b). Your algorithm should have a time complexity better than running a full MST algorithm from scratch. Assume that all edge weights are distinct. Explain the main idea of your algorithm along with an explanation of its time complexity in plain English as well.

Q#3: Can you use the DFS algorithm to compute the number of distinct paths between two given vertices 's' and 't'? Two paths between 's' and 't' are considered distinct if they differ by 1 or more edges. If you think the answer is yes, then provide the pseudo-code for a DFS based algorithm that computes this number in O(|V|+|E|). If you think the answer is no, draw a graph with at-least eight vertices in total, with two of its vertices labeled "s" and "t", in which DFS will fail to compute the number of distinct paths between "s" and "t".

Q#4: The government of Pakistan is planning to build a new town near Lahore to provide houses to homeless people. Initially, it was planned that all the roads will be two way. However, the newly elected government changed the plan and mark all the roads as one way roads. The directions of the roads are marked in a way that that all places are reachable from the Town Hall. Due to this change, not all the places in the town are reachable from each other. In other words there are some pair of places that has no directed path between them. This means some new roads must be constructed to provide path between all those pair of places that are not connected in this one way road network. The government wants to spend minimum amount of money on this project. You are asked to analyze the current road network and add minimum possible new roads such that all pair of places has a path between them. Formulate this problem as a graph problem and give a linear time algorithm that can compute the new edges (roads) given the existing road network.

Q#5: Suppose you are given a weighted directed graph G(V, E) with possibly negative edge weights. Suppose you also know that every path between all pairs of nodes in this graph has at most eight edges. We are interested to compute single source shortest paths in this graph given a source vertex "s". Is it possible to Device an O(V+E) time algorithm for this problem. If "yes" then design and explain the algorithm.

Attempt any one part of Q#6

Q#6(a):

Give a counterexample to the conjecture that if a directed graph G contains a path from u to v, then any depth-first search must result in $v.d \leq u.f$.

Q#6(b):

Give an example of a directed graph $G = (V, E)$, a source vertex $s \in V$, and a set of tree edges $E_\pi \subseteq E$ such that for each vertex $v \in V$, the unique simple path in the graph (V, E_π) from s to v is a shortest path in G, yet the set of edges E_π cannot be produced by running BFS on G, no matter how the vertices are ordered in each adjacency list.

Q#6(c):

Give a counterexample to the conjecture that if a directed graph G contains a path from u to v, and if $u.d < v.d$ in a depth-first search of G, then v is a descendant of u in the depth-first forest produced.