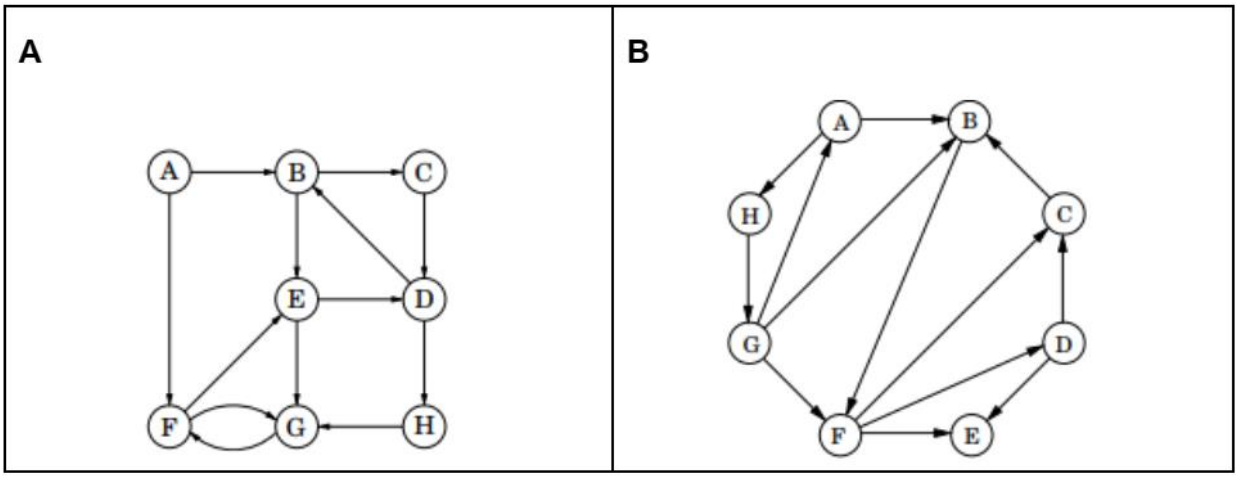
Name: _____ Roll #: _____ Section: _____

Design and Analysis of Algorithms

CS302

Total Marks = 60, each question carries 10 marks.

Q1) Run the DFS algorithm on the following graph. Draw the DFS tree (or forest). Label each node with pre and post numbers. Indicate the backward, forward and cross edges (See slides on DFS to learn about edge classificaitons). Wherever multiple options are available, visit the nodes in the alphabetic order.



Q2) You are given a DAG G=(V, E) of the courses in your curriculum (nodes), with the edges between them showing the pre-requisite relationship, i.e. an edge going from A to B means that A must be studied before B: so you take the course A in one semester and then B must come in the following semester. Imagine that other than the prerequisite relationship, there is no bound on the number of courses you can take in a semester. Design an O(|V|+|E|) algorithm to find the minimum number of semesters needed to graduate.

Hint: notice that this problem asks you to find the longest path in the DAG, since there will be at least that many semesters needed to graduate.

Q3) The reverse of a directed graph G = (V, E) is another directed graph $G^R = (V, E^R)$ on the same vertex set, but with all edges reversed; that is, $E^R = \{(v, u): (u, v) \in E\}$. Give a linear-time algorithm for computing the reverse of a graph in the adjacency list format. Give a clearly specified pseudo-code.

Name: _____ Roll #: _____ Section: _____

Q4) Recall that 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 paths between s and x, where s is the source vertex and x is any other vertex of the graph. But shortest path may not be unique. Suppose we want to count the number of distinct shortest paths between s and x for each x, where xϵV. 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 A: 1

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

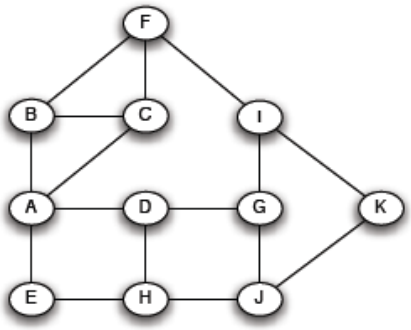
From A to G: 1

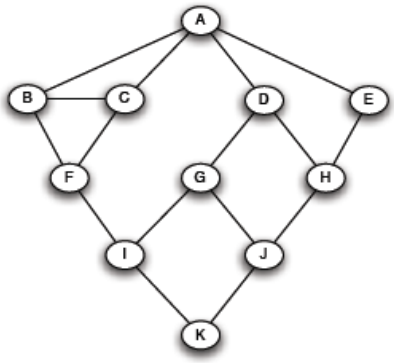
From A to H: 2

From A to I: 3

From A to J: 3

From A to K: 6





Name: _____ Roll #: _____ Section: _____

You are required to design a linear time algorithm that counts the number of distinct shortest paths between a given source vertex s and all the other vertices in the graph. You can assume that graph is represented using adjacency list representation.

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

Give detailed explanation of your algorithm

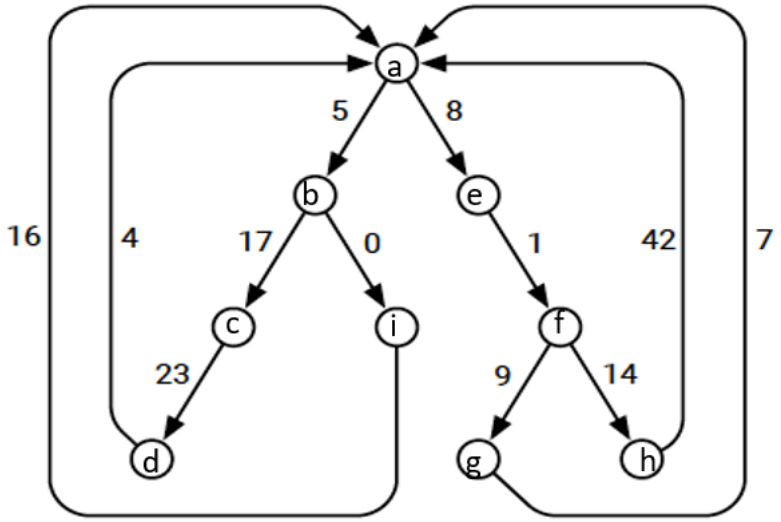
Solution

Use the following idea: Start BFS with the start node but instead of saving only one vertex as parent, store list of parents if more than one nodes will give the same distance. Number of unique shortest paths from s to s is 1. For all other vertices, number of distinct shortest paths are sum of distinct shortest paths of their parents.

```
Modified_BFS(G(V,E), s)

        distinct_paths[1..v]
        color[1..v]
        pi[1..v] //each element is a list
        dist[1..v]
        for all v in V
                distinct_paths[v]=0
                color[v]=white
                dist[v]=inf
                pi[v]=null

        color[s]=grey
        dist[s]=0
        distinct_path[v]=1
        make FIFO queue Q
        Q.insert(s)
        while(Q is not empty)
                u = Q.remove()
                for each v adj to u
                        if(color[v]==white)
                                color[v]=grey
                                dist[v] = dist[u]+1
                                pi[v].insert(u)
                                distict_path[v] += distint_path[u]
                                Q.insert(v)
                        else
                                if(dist[v] == dist[u+1])
                                        pi[v].insert(u)
                                        distict_path[v] += distint_path[u]

                color[u]=black
```

Q5) The following picture shows an example of a Korchoff Graph. It is a weighted, directed graph. As you can see the graph has a binary tree component at is center. Moreover,there are edges going from each leaf of the binary tree

backto the root. All edges are weighted and the weights are non-negative. We wish to compute single-source shortest paths in a Korchoff graph.



For example, the shortest path costs from vertex 'b' to all other vertices are as follows:
b -> a: 16, (b, i, a)
b -> c: 17 (b,c)
b -> i: 0 (b,i)
b -> d: 40 (b,c,d)
b -> e: 24, (b, i, a, e)
b -> f: 25, (b, i, a, e, f)
b -> g: 34, (b, i, a, e, f, g)
b -> h: 39, (b, i, a, e, f, h)

a) Is the following statement true or false? Justify your answer:
Dijkstra's shortest path algorithm takes O(|V|lg|V|) time on a Korchoff graph.

Solution
True.
Usually, Dijkstra's algorithm takes O((|E|+|V|)lg|V|).
For a Korchoff graph,
|E| = #(edges in the tree part) + #(edges going from leaves to root)
#(edges in the tree part) = |V| - 1
#(edges going from leaves to root) <= |V|
Therefore, |E| < 2|V|
Hence, a call to Dijkstra takes only O(|V|lg|V|).

b) Design a Single Source shortest path algorithm for the Korchoff graph which works in linear time. The call to the algorithm looks like: spKorchoff(G=(V, E), s, r), where G is a Korchoff graph, s is the source node used to measure the shortest paths, and r is the root node of the binary tree component of the graph. First describe the algorithm in few lines (or steps) in English, then write the pseudocode.

Solution
Step 1: Find path from s to all its descendants in the tree using a dfs starting from s and ignoring the (x, r) type edges back to root.

 Step 2: Find shortest path from s to r in linear time: this can be done in the same dfs as above by specially treating (x, r) type edges (updating a minSoFar for r with each (x, r) type edge).
 Step 3: Find paths from r to every other node using a dfs (ignoring s and its descendants, i.e. ignoring any (x, s) and (x, r) type edges in the traversal).

Q6) Agent Bob is living in his houseat A in the enemy territory. There are n other agents in the area, stationed at hotels h_1, h_2, …, h_n. They all must visit Bob's house at A for a top secret meeting. However, the area is constantly monitored and there is a risk of being monitored associated with every road. You have a map, G=(V, E), of the n hotels h_1, h_2, …, h_n and A. This map shows every road between any two hotels or between a hotel and A and also shows its associated risk value, which is a positive weight. The total risk of a path taken by an agent is the sum of the risk values of all the edges in that path. You need to write an algorithm that produces paths for all agents, such that the total risk for each agent is minimized. Your algorithm should work in O((|V|+|E|)lg|V|).