1. Given a graph G = <V, E>, what is to be the running time of the depth-first search algorithm, as a function of the number of nodes n = |V| and edges m = |E|, if the input graph is represented by an adjacency matrix instead of an adjacency list?
   a) Runtime = O(n^2)
   b) If input graph is adjacency matrix instead.

2. Kolade is at a train station in a foreign town. He wants to select a hotel that has the maximum number of shortest paths from the train station. He thinks that this should reduce the risk of getting lost. Suppose he gives you a city map represented via a graph G = <V , E> with n = |V| locations and m = |E| edges connecting the locations. Each edge connecting a pair of directly connected locations has a unit distance, say 1. Help Kolade to find a proper hotel by designing a O(n + m) runtime algorithm that finds the number of shortest paths between the train station, located at node s and every hotel on the map. Note that Kolade expects you to convince him that your algorithm is correct and it does find all possible shortest paths. Your solution can be in the form of a pseudocode.

```
While Arrive the address
{
    Compare all of neighbor Length m
    Couse the min length
    Remember correct node
    If next node equals last node
    {
        Choose another one
    }
    Return path Length
}
Compare all path Length Choose the Min one
```

3. A communication network, such as the Internet, can be modelled as an undirected graph G = <V, E>. Here, the vertices V are the computers on the network, and the edge set E consists of one edge for each pair of computers that are directly connected. We assume that the edges of G are undirected, that is, if there is a direct connection from computer u to computer v, then there is also a direct connection from computer v to computer u.

It is highly desirable for a communication network graph to be connected, so that every computer on the network can communicate, possibly through a series of relays, with any other computer. But networks can change, with some computers failing and other computers being added to the network. It is useful to have a testing algorithm that collects information about the current network graph (vertices and edges) at designated times, and determines properties related to connectivity.

Describe, in words and pseudocode, a testing algorithm that given an undirected graph G = <V, E> representing the current network decides whether or not the network is connected. A graph (network) is connected if there is a path from any node to any other node in the graph.

You may assume that G is given in adjacency list format. Your algorithm must run in O(n + m) time, where n = |V| is the number of computers on the network, and m = |E| is the number of connecting edges.

```
Algorithm BFS(G, V)
    Input: A graph G and a vertex V of G
    Output: A labeling of the edges in the connected component of V as discovery edge and
cross edges
    Create an empty list, L0
    Mark V as explored and insert V into L0
    I == 0
    While Li is not empty
    {
        Create an empty list, L i+1
        For each vertex, v, in Li
        {
            For each edge, e = (v, w), incident on v in G
            {
                If edge E is unexplored
                {
                    If vertex w is unexplored
                    {
                        Label E as a discovery edge
                        Mark w as explored and insert w into L i+1
                    }
                    Else
                    {
                        Label E as a cross edge
                    }
                }
            }
        }
        I = I + 1
    }
```