1. (30 points) Below is a weighted undirected graph. (a) Show the DFS (depth-first search) tree found by the recursive DFS algorithm and list the vertices in the order of adding to the DFS tree. (b) Show the DFS tree found by the non-recursive DFS algorithm and list the vertices in the order of adding to the DFS tree. (c) Show the BFS (breadth-first search) tree found by the BFS algorithm and list the vertices in the order of adding to the BFS tree. (d) Show the MST (minimum spanning tree) found by Prim’s algorithm and list the vertices in the order of adding to the MST. (e) Compute the shortest paths from vertex A to all other vertices using Dijkstra’s algorithm and list the vertices in the order of adding to the cloud. For all the above questions, we start with vertex A and ties are broken by alphabet order of vertices.

2. (30 points) Given two strings X and Y, the edit distance between X and Y, D(X, Y), is the minimal number of operations performed on X so that X becomes Y. The allowed operations are: delete a letter, add a letter, or change a letter. For example, D(“hurry”, “carry”) = 2, because “hu” can be changed to “ca” by two operations. Please design an efficient algorithm to compute D(X, Y) and analyze its complexity.

3. (30 points) In a company, the supervisor-supervisee relation can be represented by a single tree T, with the president being the root of the tree. Given the tree T, you are asked to compute the maximal number of employees that can be invited to a party such that an employee and his/her immediate supervisor cannot be invited at the same time. Please design an efficient algorithm for this problem and analyze its time complexity.

4. (10 points) A greedy algorithm for the Vertex Cover problem works as follows: Always move a vertex with the highest degree into the vertex cover, and then delete all edges incident to this vertex in the graph. Repeat the above operation until no more edges in the graph. Please provide a counter example to that this algorithm is not a 2-approximation algorithm for the Vertex Cover problem.