EECS4020 Algorithms

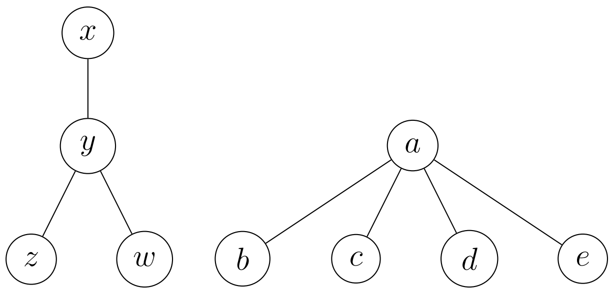
Exam 3

Time: 10:10am – 12:00pm, June 15, 2022

Answer All Questions. Total Marks = 10 + 15 + 15 + 15 + 15 + 15 + 15 + 10 = 110.

Important: (1) When you design an algorithm, you need to show its correctness, and analyze its running time. (2) When you get stuck, try another question first.

1. John has maintained a union-find data structure for a set using a set of trees. The following is the current status of the trees, so that the set is currently partitioned into two subsets:



   (10%) Suppose John is using **union-by-size** strategy to perform `union`, and the **path-compression** heuristic to perform `find`. Describe the resulting trees after (i) `union`(a, x), and (ii) then `find`(z).

2. Suppose we perform Kruskal's algorithm to construct an MST T for the edge-weighted graph in Figure 1, where the label on the edge represents the edge weight (For instance, the weight of the edge $\{\mathsf{A}, \mathsf{S}\}$ is 1 unit). At a certain moment, the algorithm considers each edge one by one in some order, and decides if the edge is in T or not.

   (a) (10%) Describe a possible order of edges that are considered, and for each edge whether it is selected as an edge in T.

   Remark: Be careful! For this question, no partial mark will be given.

   (b) (5%) Is your order in (a) unique? Why or why not?

3. Suppose we perform Dikjstra's algorithm to find the shortest distance from the source node S to each node in the edge-weighted graph in Figure 1, where the label on the edge represents the edge weight (For instance, the weight of the edge $\{\mathsf{A}, \mathsf{S}\}$ is 1 unit). In general, the algorithm would consider the nodes one by one in some order, and for each node considered, the algorithm performs a couple of *relax* operations to update the distances from S to some other nodes.

   (a) (10%) Describe a possible order of nodes that are considered. Also, describe the shortest distance from S to each node.

   (b) (5%) Is your order in (a) unique? Why or why not?

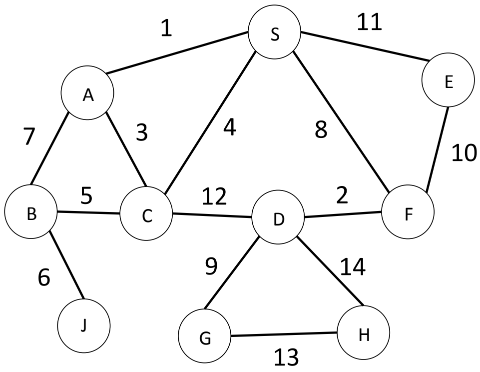


Figure 1: The graph used by Q2 and Q3

4. This question is about SCC (Strongly Connected Component).

   (a) (5%) Find the number of SCCs in Figure 2. For each SCC, mark clearly all nodes that belong to it.

   (b) (10%) Prove or disprove: For any directed graph, if two distinct nodes are in the same SCC, then there exists a *simple* cycle (i.e., a cycle where each node appears at most once) that contains them.

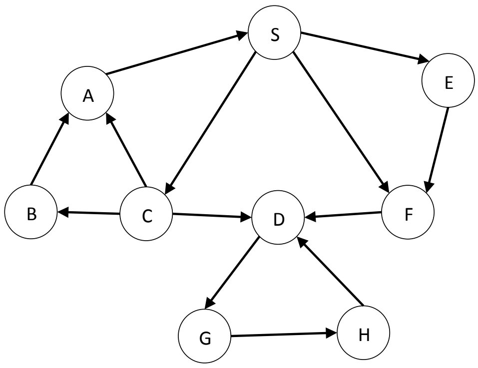


Figure 2: The graph used by Q4

5. Let $G = (V, E)$ be a directed graph, and u, v be two specific vertices in V.

   (15%) Give an $O(|V|+|E|)$-time algorithm to count the number of paths from u to v which use the minimum number of edges.[1]

6. Let $G = (V, E)$ be an edge-weighted directed graph. Running Bellman-Ford algorithm can check, in $O(|V|\,|E|)$ time, whether G contains some negative-weight cycle.

   (15%) Let u, v be two vertices in G. Show how to check if G contains a negative-weight cycle that contains both u and v in $O(|V|\,|E|)$ time.

   Note: The cycle does not need to be simple.

[1] Assume that the desired number of paths is an integer small enough to be represented by a computer word.

7. Let $G = (V, E)$ be a simple, connected and undirected graph. In our homework, we have defined the so-called *articulation points*, which are vertices such that the removal of any of them disconnects the graph.

   For instance, consider Figure 1. The vertex B is an articulation point, and so is the vertex D. On the other hand, all the other vertices are not articulation points.

   In this question, we want to study a related concept, called *bridge*. A bridge in a graph is an edge, such that its removal will disconnect the graph. Take the graph in Figure 1 as an example. The edge (B, J) is a bridge, while all the other edges are not.

   (15%) Give an $O(|V| + |E|)$-time algorithm to find all the bridges in a graph $G = (V, E)$.

   Note: You may assume, without proof, that all articulation points of a graph $H = (U, F)$ can be found in $O(|U| + |F|)$ time.

8. Let $G = (V, E)$ be a simple edge-weighted undirected graph, where each edge in G has a distinct integral weight chosen from $\{1, 2, 3, \ldots, |E|\}$. A path P is called *increasing* if the edges successively used by P have increasing weights.

   For instance, consider the graph in Figure 1. The path $\langle \mathsf{S}, \mathsf{C}, \mathsf{D} \rangle$ uses two edges with weights 4 and 12, successively, and this path is increasing. In contrast, the path $\langle \mathsf{S}, \mathsf{F}, \mathsf{D} \rangle$ uses two edges with weights 8 and 2, successively, and is not increasing.[2]

   Let s be a specific vertex, called *source node*, in G. We want to find out for each node v in G if there is an increasing path from s to v, and if so, what is the minimum weight of such a path. For instance, in the graph in Figure 1, with source node S, the minimum weight of an increasing path from S to B is $1 + 7 = 8$, and from S to J is $4 + 5 + 6 = 15$ (or $1 + 3 + 5 + 6 = 15$), while the minimum weight of an increasing path from S to G is $+\infty$ (since there is no such path).

   (10%) Give an $O(|V| + |E|)$-time algorithm to solve the above problem.

[2] By default, we assume that a path using 0 edges or 1 edge is increasing.