**Practice Quiz 3**

## Problem 1 True and False with Justification

1.1 The shortest path between two nodes is necessarily a part of some MST.
False: Can prove by contradiction (Multiple acceptable answers)

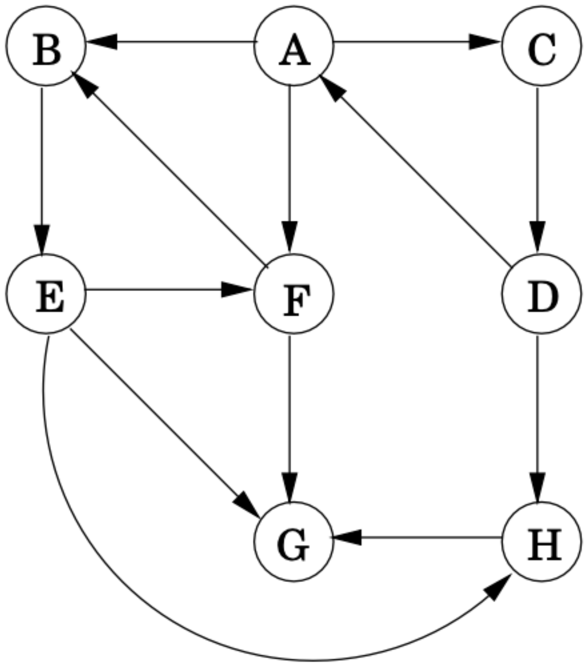
1.2 Prim's algorithm can be applied on a graph with negative edges.
True: Order of adding edges is preserved and cut property is applicable to negative edges.

1.3 Kruskal's algorithm can be applied on a graph with negative edges.
True: Order of adding edges is preserved and cut property is applicable to negative edges.

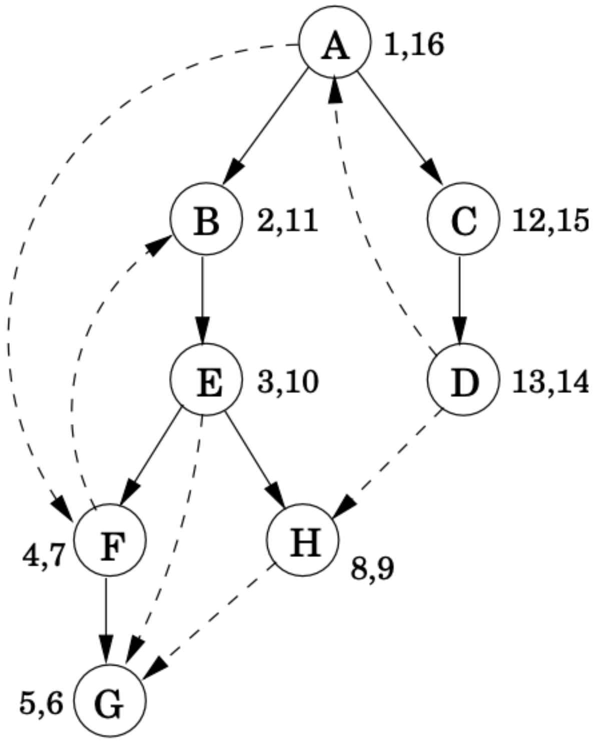
1.4 There can be multiple MSTs for the same graph G.
True: A graph can have more than one MST in the case where both trees have the same overall weight but different paths to complete the tree (Proved in lecture)

1.5 If all edge weights in a connected graph G are distinct, then G has a unique minimum spanning tree.
True: Prove by contradiction. (https://jeffe.cs.illinois.edu/teaching/algorithms/book/07-mst.pdf) or using plain logic that running prim's or kruskal will produce a unique MST since there is only one edge that can be selected at each step.

**Problem 2 Consider the graph below for the following subproblems:**

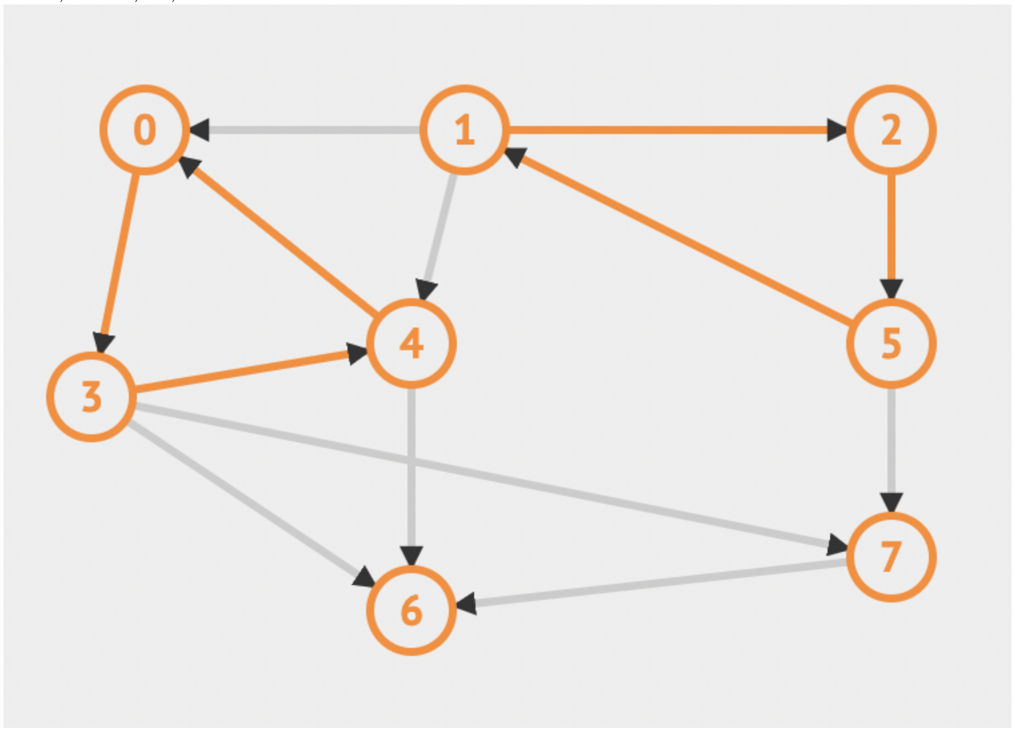


2.1 Perform a depth-first search and provide the pre and post labels for the vertices A-H below



2.2 How many strongly connected components exist in the graph above? (Only list the number; no work needed) **4**

2.3 List the strongly connected components in the graph above
BEF, ACD, G, H



(can run in https://visualgo.net/en/dfsbfs for visualization)

## Problem 3 Computopia

The police department in the mythical city of Computopia has made all streets one-way. The mayor contends that there is still a way to drive legally from any intersection in the city to any other intersection, but the opposition is not convinced. Furthermore, the city elections are coming up soon, and there is only enough time to run a linear time algorithm.

**(a)** Formulate this problem as a graph-theoretic problem, and explain why this problem can indeed be solved by a linear-time algorithm (i.e., linear in the number of intersections and streets).

Let G be the graph whose vertices are the intersections and edges are the corresponding oneway streets. There is a way to legally drive from any intersection to any other intersection if all vertices in G are in the same strongly connected component (by the definition). Let n be the number of intersections and m be the number of one-way streets. We can solve the problem at hand by finding the strongly connected components of G in O(n + m) time, and then checking whether or not there is one strongly connected component.

**(b)** Suppose now that the mayor needs to show that, even though her original claim was false, something weaker does hold: If you start navigating one-way streets from town hall, there is always a way to drive legally back to the town hall. Formulate this weaker property as a graph-theoretic problem and show that this can also be solved in linear time.

Let r be the number of vertices reachable from town hall. Note that r can easily be computed using a depth-first search in O(n + m) time. Next, find the strongly connected components of G like in part (3a), and check whether the size of the component containing the town hall is r. If the size of the component is r, then all vertices reachable from town hall can also return to town hall, and the claim is true. Otherwise, there exists a vertex reachable from town hall that cannot return to town hall, and the claim is false. Computing strongly connected components and determining the size of each component can be done in O(n+m) time, which proves the claim about the linear time algorithm.

## Problem 4 Batman and Buzz Boi

Tanmoy is graduating soon and has decided to take up a job as Batman's new sidekick in Gotham city: Buzz Boi. New to the job, Tanmoy is trying to minimize his run-ins with the city's numerous villains.

The Batman provides Tanmoy with a list of n neighborhoods (labeled 0 to n-1) on a map (with marked distances between each neighborhood) of Gotham City. Each of the neighborhoods is connected by a two-way highway. Tanmoy is also provided with a solar-powered BuzzMobile that can travel up to 2d miles a day.

Given that every neighborhood within d miles from Tanmoy's house will be his responsibility, help him come up with an algorithm to pick the neighborhood for him to choose his house so that he has the fewest other neighborhoods to look after. Assume no two neighborhoods have the same number of other neighborhoods within distance d.

**(a)** Describe (in words) the algorithm for finding the required neighborhood.

1. Convert the map into a graph where each of the neighborhoods are the vertices and the two-way highways connecting them are the edges with the distances as the weights.
2. Now, starting from vertex 0, run Dijkstra's Shortest Path Algorithm using the graph and the selected node as the input. Assume Dijkstra's uses the Priority Heap implementation.
3. Using the output from dijkstra's (the distances of each vertex from the source) count the number of vertices that are less than d units away from this source and store it in a tuple/dictionary/hashmap corresponding to the source vertex.
4. Repeat steps 2 and 3 for all vertices from vertex 1 to vertex n-1 so that you have the values for each vertex.
5. Using the tuple/hashmap/dictionary of number of vertices that are less than d units away from each vertex as the source, return the vertex corresponding to minimum value.

**(b)** What is the runtime of the algorithm that you gave above?

Assuming |V| = n and |E| = m, Step 1 takes O(n+m) time. Step 2 takes O((n+m)logn) time. Step 3 takes O(n) time. Step 4 loops on steps 2 and 3, n times. This makes the final runtime O( (n+m) + n ( (n+m)logn + n) ) which can be reduced to O(n((n+m)logn)) since the n ( (n+m)logn + n) term dominates (n+m) and (n+m)logn term dominates n. This is asymptotically the same as running as running Dijkstra's n times (for each vertex).

Food for thought: Why can't we use DFS and BFS for this question? Extra: Try checking out the Floyd–Warshall algorithm.

## Problem 5 MST Check

Suppose we are given an undirected graph G = (V,E) with positive weights w(e) for each edge e in E. We are also given a minimum spanning tree T. Now suppose that for every edge e in E, we replace its edge weight w(e) by w(e) + 1. Is the tree T still guaranteed to be a minimum spanning tree for this new weighted graph? Explain.

Yes, T is still guaranteed to be a minimum spanning tree for the re-weighted graph. Suppose the weight of T with the initial weights is a. It follows that the weight of T in the re-weighted setting is a + n - 1. If T is not a minimum spanning tree of the re-weighted graph, then there exists a spanning tree T' with weight b < a + n - 1. However, this implies that the weight of T' in the initial setting is b - (n - 1) < a, contradicting the fact that T is a minimum spanning tree. Therefore, T is a minimum spanning tree in the re-weighted graph.