Graph Algorithm Design

Here is a video walkthrough of the solution.

For each of the following scenarios, write a brief description for an algorithm for finding the MST in an undirected, connected graph G.

(a) If all edges have edge weight 1. Hint: Runtime is O(V+E)

The key idea here is that any tree which connects all nodes is an MST. We can run DFS and take the DFS tree. You could also take a BFS tree, or run Prim's algorithm with a queue or stack instead of a priority queue (this would be equivalent to BFS/DFS). Unfortunately, a modified Kruskal's will be slightly slower, because even if we don't need to sort edges, the union-find operations will take additional time.

(b) If all edges have edge weight 1 or 2. Hint: Use your algorithm from part (a)

Remove weight 2 edges from the graph so only weight 1 edges remain. Now run an algorithm from part (a) as far as possible (e.g. find a DFS forest). We will have some number of connected components. Use these connected components as nodes in a new graph G*. Look at the weight 2 edges in G. For each edge, if the nodes containing the two endpoints are not already connected in G*, add an edge between the two containing nodes in G*. Now we can run our algorithm from part (a) again to complete the MST.