Algorithms Illuminated Chapter 16

Problem 15.1

The basic setup is that we have a graph (G) with distinct, nonnegative edge costs. We also have a MST (T) and arbitrary shortest-path (P) for the graph. Which is true if we increase all edge lengths by 1? (G')

a) T must be an MST and P must be a shortest s-t path.

(False). See the counter example above. A,B,C is a shortest path in G but not in G'!

b) (True). Theorem 15.6 states that MBP → MST. Incrementing all edge lengths by 1 does not change whether T is a spanning tree. Further, with a uniform increment to all edges, it's not possible for an edge that met MBP to no longer do so, because all other candidate edges increase by the same amount.

15.2

The output of this algorithm will never have a cycle, because that would imply some edge that was tested could have been removed while maintaining a connected graph.

The output of this algorithm will always be connected because it starts connected and only removes edges if the graph remains fully connected.

Is the output a MST? First, the output will be a spanning tree, per the two previous paragraphs. Does it meet the MBP? Well, suppose the algorithm$_{(v,w)}$ does not satisfy the MBP for an edge. That means, at some point, there was more than one path between v and w, and the path that was indirect had only strictly lower edge weights, yet was broken instead of removing v and w. However, that's not possible with this algorithm, because (v,w) would be checked first and found to be removable without violating connectedness. Therefore, the MBP must be met.

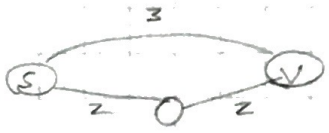
Per the arguments above, (d is true).

15.3

a) (True), Simply use $\frac{1}{\text{weight}}$ or $-1 \cdot \text{weight}$ for each edge with Prim or Kruskal.

b) Take spanning tree weights $\{w_1, w_2, \ldots, w_n\}$. If a MST, no weight can be replaced with a smaller weight while maintaining a spanning tree. All spanning trees must have the same number of edges $(n-1)$. There is no way to increase a weight, w_i, while decreasing $\sum_{i=1}^{n} w_i$ if all w_i are positive. (True)

c) (False). See below for a counter-example



d) Remove a minimum cost set of edges such that the remaining graph is acyclic. The MST problem ensures a connected, acyclic graph, but only ensures the remaining edges have minimum costs. If all edges have total weight S, then $S - S_p = S_{MST}$. Therefore to minimize the total of edges removed, need to invert the weights and solve the maximum-cost spanning tree; then the excluded edges solve the intended problem. (True)

15.4 Prove that, if T is an MST of graph with real-valued edge costs, every edge of T satisfies the MBP.

Try proof by contradiction. Assume there is a MST with an edge that does not satisfy the MBP. That means that an edge exists between (v, w), while there is a path between (v, w) with all edge lengths less than the weight of (v, w). That means that both v, w have other incident edges with lower weight than (v, w). As every other vertex must also be connected by the MST, there is no reason to choose (v, w) over either of the lower-weight incident edges, because it would no longer be an MST in that case. Therefore, it is not possible to violate MBP with a MST.

15.5 Removing non-distinct edges only changes the fact that there can be multiple paths between two vertices, (v, w), with the same maximum bottleneck weight. Let's say there is an edge (v, w) with weight w^*. There is also a path (v, w), not using that edge, that has a maximum bottleneck of w^*. If the edge is removed, the two resulting connected components can be joined by an edge of at most w^*. Therefore, it is still an MST, with no net change in total cost. We can break "ties" arbitrarily, this relates the MBP to require an edge to have cost $\leq$ the maximum of all other v, w paths.

15.7

a) To connect a graph, there must be at least one crossing edge for every partition. The min. for each such partition must be in the MST, else, we could substitute the crossing edge for one with less cost and get a lower-cost tree than the "MST."

b) Prim's will always consider the lowest cost edges first. It also explicitly maintains a partition at each iteration, and chooses the minimum crossing edge for that "cut".

c) For Kruskal, an explict bipartite graph is not maintained. However, every edge selected by Kruskal meets the minimum crossing edge of a cut property. The edge selected is the global minimum for all UCC's at a given iteration. All UCC's, except the two being joined, can be arbitrarily added to the two joining UCCs without affecting which edge is selected, showing the selected edge is the minimum crossing edge for a valid partition.