Problem Set 4

All parts are due Tuesday, November 10, 2015 at 11:59PM.

Name: David Walter

Collaborators: Elizabeth Cox

Part A

Problem 4-1.

We can use BFS. We start by going through the edges and at each zero weight edge we remove it, and in doing that, we will combine the two vertices that each zero length edge connects, in a sense creating a multi-vertex. After going through all of the edges, you will be left with only edges of weight two. We can now treat this as the question of finding the shortest path in an undirected graph with edge weights of one. We know from class and the notes that we can use BFS to find the shortest path from a starting vertex, s, to an ending vertex, t, in $O(V + E)$ time.

Problem 4-2.

(a) a) Let c be the number of connected components before the root was removed. We know that after the root is removed we will have $c + (numberofchildren - 1)$ connected components. If there was only one connected component before we removed the root then we have $numberofchildren$ connected components. We know this because if we have a DFS tree, then we know that there are no cross-edges, and any back edges would be in the sub-trees of the children. And any vertex with a back-edge to the root would have it's back-edge deleted after the root was removed. Without any back edges in the children, we know that the only path between one child and another child was through the root, but with the root removed, there is no path between any of the children, and each child of the root is a component of a separate sub-graph.

(b) We know that we cannot have cross edges in a DFS tree, and we will only have tree edges and back edges. If we assume there are no back-edges from the sub-trees of the children of c to any of the ancestors of c, then we know that the only path from the children of c, to the ancestors of c would be through c. This would mean that the subtrees of the children of c would not be connected to any part of the graph that was not part of the sub-tree of c, so we know that removing c would create at least one new connected component, disconnecting G.

(c) Go through the DFS tree, iterating through the vertices. At each vertex you go through all of the edges and check of it is a back edge and if so, check the condition for v.top, setting v.top to either v.d or w.d, as described in the problem description. We also need to make sure our vertices are augmented with the list of their back-edges, so we are able to access the back-edges in the tree. Augmenting the vertices with their back-edges can be done in constant time as you create the DFS tree. You will go through every vertex once, and you will go through each edge twice in the worst case. This makes the run time $O(V + E)$.

(d) Knowing part a, b, and c, we create the DFS tree of G and augment each vertex with v.d and v.top all of which can be done in O(V + E) time as descried in part c. As we are calculating v.top, we can also augment each vertex with a boolean called 'disconnect' that says whether v.top is equal to w.d. If 'disconnect' is True then we know that there is not a back-edge from a descendent of vertex v to any of the ancestors of vertex v. In knowing this, we know from part b that removing v will disconnect G. We also know that 'disconnect' will be set to true for the root as described in part a. Also, we will be keeping track of a variable called 'best' that keeps track of the vertex that will create the most number of disconnected components that will disconnect G. 'best' will first be set to the root of the DFS tree. As we go through the other vertices, if disconnect is true for vertex v, then we will compare the v.n = 'number of children' of v and compare it with 'best.' If v.n is greater than best.n, then best becomes v. After going through every vertex, we will have the vertex with the best v.n value. If if the end, best equals the root and root.n equals 1, then we know that we cannot disconnect G. Otherwise, best will be the vertex that breaks up G into best.n number of subgraphs.

Part B

Problem 4-3.

(a) *Submitted.*

(b) *Submitted.*