CS 4102

Basic Programming 2

Joseph Lee

Wiring Programming Assignment – Writeup

For this assignment, Kruskal algorithm was used to find the minimum spanning tree of a connected weighted graph. At the end, my algorithm was running at Θ(E log V) runtime, which is the expected time complexity for this assignment. I am using labels to refer j1, l1, o1, etc. and light, breaker, light, etc. as names. I decided to create a 6 functions to implement Kruskal's algorithm: find, makeSet, union, validateLink, lightSwitchCheck, and sumSwitch. For find-union, I created a private method called find(Map<String, Segment> segments, String label) which traverse recursively through segments until it finds the parent of the label. The parent of the label can be either root parent or one that is assigned by union method. The hashmap of segments (Map<String, Segement>) is created in the private method called makeSet where the label is a key and the segment (which stores the parent and rank) as the value. Then the find method is called in a private method called union, which is using union by rank, to find the parents of two vertices of and edge. Then the union method compares their parent's rank to assign new parent and increment the rank. Ultimately, KruskalMST(List<Edge> newEdges, int newVertices, Map<String, String> labelItemMap) uses find, makeSet, and union to find the minimum spanning tree. kruskalMST first sort the edges in ascending order of weight and calls union only when the edge is valid and the parents of two vertices of the edge are not the same and if they are the same then it creates a cycle which makes it not valid. There may be a chance that the label of a component does not match its name (for instance, light can be o1) so I decided to use a hash map called labelItemMap which stores the label as a key and the name as value. This will ensure that whenever I check the validity of an edge, it will check for the relation in the names not the labels. For validity of an edge to be checked, I created two private methods: validateLink and sumSwitch. First validateLink checks for any invalid cases like light to breaker, breaker to light, switch to switch, etc. This method is called when the input edges are read. And then the validity of edges between a switch and a light or a light and light are checked in the main after all the edges are read. Finally, the private method called sumSwitch is called to find the sum of the shortest path after the kruskalMST are called twice for each phases. The sumSwitch go through every edge between a switch and other components beside light and adds the smallest value to sum before it gets to the next switch. The shortest path is then calculated by adding the sum of two phases of kruskalMST and the sumSwitch. At the end, the total runtime of this program is Θ(E log V) as we were taught in the class.