My algorithm uses Kruskal's algorithm in two phases. The first phase would process all the nodes that don't have switches or lights using Kruskal's. This allows the algorithm to bypass the constraint of the outlet having a direct path to the breaker because if there are no switches to be processed, there are no switches to get in between the outlet and the breaker box. The MST cost of the no switch/light nodes are stored, and the second phase begins. In the second phase, the constraints are used to throw out invalid edges, such as lights to lights with different associated switches. That MST cost would be added to the MST cost of the first phase and returned. I used a hashtable to act as my graph where the keys are the names of the nodes and the values the nodes themselves that update their label field when findset and union are called. I had a two separate arraylists to store the edges with and without lights and switches. Kruskal's allowed me to go through without worrying about cycles because of the find-union data structure made. The findset runs O(n), but is expected to run constant because it uses path compression. The union depends on findset, so O(n), but expected constant time as well. Kruskal's algorithm runs on $\Theta(E\log(V))$. The overall runtime is $\Theta(E\log(V))$.