My algorithm starts by taking in all critical points of the badge ranges, which are the min and max of each badge range. Any duplicates are taken out. The ranges will be the critical point by themselves and the numbers in between since the numbers in between are basically the same. So given [1,10] and [7,14], the ranges made will be [1,1],[2,6], [7,7], [8,9],[10,10],[11,13],[14,14]. Each of the ranges will be run through a DFS on the start node, only traveling down if the range fits that lock's badge range. Once the destination node is found, stop checking that unique badge range we made to prevent over counting and duplicates. Since DFS is $\Theta(E + V)$ and we do DFS $\Theta(l)$ times, so the overall run-time is $\Theta(l * (V + E))$.

In collaboration with Parth Raut, plr6uqx in coming up with how to find the unique badge ranges