Dusty Payne
SID 3033900380

1. Since every node has a guavabot there is no need for scouting. Run DFS and all pairs shortest paths on the graph. Create an array to store the total number of guavabots on each node at any given time. The algorithm starts from the outermost depth and remotes according to the cheapest option amongst its neighbors, which is defined as the (cost of the path * some weight based on the number of bots on the node) where more bots = lighter cost.
2. Since we know where all the bots are, we don't need to scout. Run DFS and all pairs shortest paths. Use a dynamic programming algorithm to move the bots home. The algorithm checks nodes in descending depth value (starts from max depth away from home) and remotes the vertex to the neighbor whose path to home is the cheapest, where path cost is the sum of edge weights, or increasing negative magnitudes corresponding to the number of guavabots if they exist on other nodes, since running multiple guavabots on one path will more than likely be cheaper than the alternative. Once the vertex has been remoted, mark it as visited. The algorithm terminates once all nodes except home have been visited.
3. The most naive solution (and I think only guaranteed correct one) would run remote on every vertex to guarantee the location of each bot. This is at the cost of lots of time though.
4. The solver should have two parts; finding the locations of the bots (with reasonable confidence) by scouting, and moving them home once they've been located. An algorithm to find the bots using scouting(based on the MWU algo) and the strategy in Q2 would make one solver. Another solver, as a safety, would use the strategy in Q1 forgoing any scouting to guarantee all the bots are rescued.
5. The Q1 solver would do terribly on large graphs because of the cost of moving on paths. The Q3/Q2 solver would be much better time wise, but could have trouble on graphs with a high number of guavabots, because of the chance to miss them.