**Attempted Algorithms Summary**

We have tried four different approaches to this project. First, we implemented a pure greedy algorithm, which always drives to the closest home from the current location of the car, so eventually no one walks. The next algorithm we tried is similar but optimized by TSP. We still made Rao drop off everyone at their houses, but we used TSP to find the optimal cycle instead. The third one was built upon the second one with heuristics which are based on whether we should let the next TA walk home along the way to the next of the next TA’s home. Finally, we had a completely different approach, in which we used Mean-Shift clustering to group all homes. Then Rao only dropped the TAs off at the center points of these clusters.

**Algorithms’ explanations**

1. **Pure Greedy Algorithm**
   1. **How it works**

From the starting location, use Dijkstra’s to find all the shortest paths to all homes. Then drive to the closest home. Repeat this process until Rao drops everyone off. Finally, Rao drives back to the starting location.

* 1. **Performance**

The algorithm works fine for well-distributed houses because we use the most efficient way (driving cost is slower) to get to all houses. However, in the case of some houses which are far away from the others, driving there and back would cost more than a person walking there. Therefore, it might not work well in general and we got a quality of 40.725 for both of our own inputs and all inputs.

1. **Greedy TSP Algorithm**
   1. **How it works**

The main idea is to reduce the graph of all locations to the graph of only houses. To do this, we connect edges between all homes with weight being the corresponding length of the shortest path in between. Now, we still want to find a way to travel all the homes and return to the starting location, so it reduces to a Travelling Salesman Problem. As a result, we run an approximation algorithm for TSP on the newly constructed graph to get the desired route.

* 1. **Performance**

The algorithm performs better than the last one since the TSP algorithm looks at the entire graph and returns the optimal route. While the pure greedy algorithm only looks at the best option on the correct location. Thus, we have the quality of 38.958 for both of our own inputs and all inputs. Though this might still not be optimal because sometimes walking home which is far away from the residual would cost less than driving all the way in.

1. **Smart TSP Algorithm**
   1. **How it works**

This is basically an improved version of the last algorithm. First, run TSP approximation algorithms on the graph of homes. Then prune the route using a heuristic. If the cost (drive to the next TA’s home) + cost (drive to the next of the next TA’s home) < cost (drive directly to the next of the next TA’s home) + cost (next TA walk home), then drives to both. Otherwise, it would let the next TA get off at the current location and only drive to the next of the next TA’s home.

* 1. **Performance**

Theoretically, this approach should work better than the last one because we let some of the TAs walk instead of driving them home if it results in a lower cost. Thus, we got a quality of 38.786 across our own inputs which perform slightly better. Though we had 40.141 across all inputs, result in worse performance in general.

1. **Mean Shift Clustering Algorithm**
   1. **How it works**

Run Mean Shift algorithm with a hyperparameter r which defines the radius of a cluster to cluster all locations. Then for every cluster, we select the node that is closest to the center as the dropoff point for all TAs living inside the cluster. Finally we collect all center points and feed them into the TSP approximation algorithm to retrieve a traversal for them. We implemented this by using *sklearn* from package *MeanShift* to automatically detect the best value of r so that running TSP approximation algorithms on it will give us the best route.

* 1. **Performance**

This is considered a failed attempt, and the performance is worse than all other approaches. We believe that the reason is that the houses are generally sparse, so dropping off TAs at the center and making everyone walk one would result in much higher costs.

**Computing resources**

No other computing resources were used other than our own laptops.

**Final Algorithm**

We decided to submit the Smart TSP Algorithm as our final solver since it gives us the lowest quality across our own inputs. In addition, it generally makes sense to have a choice of letting TA walk home or driving them.