Below, we'll list out the approaches we've tried and planned TODOs in phase 2, providing our reasoning as well as the potential advantages/disadvantages:

1. **Brute force search** : idea is that if we can find all possibilities, then we can find the optimum.
   - Why? Given any car_route, we can generate an optimal drop-off schema easily with Dijkstra results, so it always gives the optimum if succeeds.
   - Advantage : guarantees optimality - will try to improve it through pruning to reach optimum for smaller inputs.
   - Disadvantage : since the generation is through DFS, it will not work for most graphs with >=20 vertices.
2. **Send all TAs home** : have not implemented for optimality, but this could reduce to a problem similar to TSP, which is the next method we'll describe.
   - Why? Since driving costs less than walking, so this would usually give a better solution than just letting all TAs walk home from Soda. Also an obvious possibility as its the other extreme from approach 0.
   - Advantage : driving home is better than walking home, and if we have 3 people living in around the same area, then driving them home is a good choice.
   - Disadvantage : a bit hard to implement since this is an upgraded version of TSP, and would be bad if TAs all live far away from each other.
3. **Mindless TSP** : as mentioned in approach 2, the problem of sending all TAs home reduce eventually to a variation of TSP.
   - Why? This is an obvious (approximate) reduction given that a TSP solution guarantees to pass through all TAs' homes.
   - Advantage : very fast to solve given the public solver, and the result is a huge improvement from approach 1
   - Disadvantage : we don't really need to reach all vertices (TSP constraint modification), and we should be able to use a vertex twice (very simple counterexample exists). These are problems and edge cases we'll address (**TODO**).
4. **K-cluster TSP** (**TODO**) : as proved in the most recent HW, similarly, if at least 3 TAs live near each other, we should drive them home and drop them off together at some point; otherwise, we should let the TA walk home.
   - Why? A logical approach following the observations from approaches 2 and 3. Clustering the TAs together allows singled out TAs to walk home so that we don't require our TSP (or other algorithm) to pass through that home vertex.
   - Advantage : follows the logic that we should only drive groups >=3 to about the exact position of their homes (choose randomly or by iterating through all possibilities)
   - Disadvantage : hard to implement, i.e. hard to have clusters based on a radius of distances with potentially different number of vertices in groups.