1.) For the team members it’s only me!

2.) The Case I decided to solve and provide information was Case #1 which was the Inland Solar Sales Travel and provide which is the best and shortest route the salesperson can take.

\* Mainly the problem I'm solving for this case is what is the best and shortest route for the salesperson

\* The solution to implementing in this case would be more like a fast and aggressive approach for this code. An approach like permutation which can be applied here. The reason being is there can be multiple routes that can be found in this solution, an algorithm. Take a Rubic cube for example, there can be different ways the cube can be solved but by using permutation mathematics it uses a linear order or sequence of set items that can arranged. The cube can have different sequences and orders to solve the cube. The same can be applied to this case, different sets of orders or “routes” in this case to find the best and most efficient route for the salesperson. Different combinations that can be used in order to find the best route. This Brute force approach is used to find all the combinations and break down the best one

* A graph in this case can be used as well, the 4 cities will be called nodes, and the edges would be the distance and total cost between each city. For the graph it will be an adjacency matrix. The adjacency matrix is being used here because it uses a 0-1 in each position or row indicating which two vertices or cities in this case are adjacent to each other. For example, let's take Row 0 as Riverside; so, 0 in this case would be riverside since there is no distance between the same cities. The other figures would be the distance from Riverside to that city, starting at 10, which would be Moreno Valley, 15 is Perris and 20 is Hemet. These numbers represent the distance between each city from each row.
* This would increase from each point, which would be Riverside.
* For example, Riverside → Moreno Valley → Perris → Hemet → Riverside: *10+35+30+20=95*
* 95 units will call it in total if that route is chosen, we are trying to find the shortest route or units in this so there are 6 variations of trips that can be used to find the shortest route or units.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 10 | 15 | 20 |

* The distance calculation would be this:
* The sum of edge weights for all the cities + the distance back to the starting point.
* Distance=i=0∑n−1 distances[i][i+1] +distances[n−1][0]

N would be the number of nodes, and the distances[i][j] would be distance between each city so i and i

* 0 and 1 would be the distance from the first city and the next
* 1 and 2 would be the distance from the second city to the next
* This would go on and would return 0 and 1 since it will be back to the starting point which would be riverside.
* There are also 6 different variations that can be used in this trip so the formula would be n!=3!=6 variations of the trip:
* Riverside → Moreno Valley → Perris → Hemet
* Riverside → Moreno Valley → Hemet → Perris
* Riverside → Perris → Moreno Valley → Hemet
* Riverside → Perris → Hemet → Moreno Valley
* Riverside → Hemet → Moreno Valley → Perris
* Riverside → Hemet → Perris → Moreno Valley
* By using these variations and the graph as well we can find the shortest route or least units. That would be the 2 option which is:
* Riverside -> Moreno Valley -> Hemet -> Perris ->Riverside since its back to the starting point.
* This would be 10+25+30+15= 80 units, this would be considered the shortest and best route.
* The purpose of this program was supposed to help the user aid decision making to find the best route there is. The user doesn’t really need to interact with the program since it’s already pre-defined from the graph or matrix.
* Discrete structures are used here by using a variety of approaches and techniques that are used in discrete math like permutation approach.
* The limitations would be this program is meant to solve for a small number of cities not a large number. This allows the program to be more brute force in way but is limited to certain areas like flexibility and scalability. To help improve these limitations on a bigger scale and approach needs to be used like dynamic programming which would help with time complexity.





