TRAVELING SALESMAN PROBLEM

i/ Optimality

- The code I've provided is an implementation of the Held-Karp algorithm, which is a dynamic programming approach to solve the Traveling Salesman Problem (TSP).
- The Held-Karp algorithm always finds the optimal solution to the TSP, unlike heuristic approaches such as the Nearest Neighbor or 2-opt algorithms, which can find good solutions quickly but do not guarantee to find the optimal solution.

ii/ Time complexity

- The time complexity of this approach is O(n^2 * 2^n), where n is the number of cities. It is more efficient than the brute-force approach of trying all possible permutations of the cities.
- Here's why:
    + The function iterates over all subsets of the set of cities, which is 2^n (since each city can either be included or not included in a subset).
    + For each subset, it iterates over all cities to find the minimum cost path that visits all cities in the subset and ends at a particular city. This is an O(n) operation.
    + Therefore, the total time complexity is O(n * 2^n).
    + However, since this operation is done for all cities, the final time complexity is O(n^2 * 2^n).

iii/ Space complexity

- The space complexity of this approach is O(n * 2^n), where n is the number of cities.
- Here's why:
    + The function uses two 2D arrays, Value and previousCity, each of size n * 2^n. This is because the

function needs to keep track of the minimum cost (Value) and the path (previousCity) for all subsets of cities, and for each subset, for all ending cities.
    + Therefore, the space complexity of each array is n * 2^n.

iv/ Subproblem overlap

- The key insight of the Held-Karp algorithm is that the TSP has overlapping subproblems. For example, the shortest route that visits a particular set of cities and ends at a particular city does not depend on the order in which the other cities are visited. This allows the algorithm to solve each subproblem once and store its result, rather than re-computing the result every time the subproblem is encountered.

v/ Bitmasking

- The Held-Karp algorithm uses a technique called bitmasking to represent subsets of cities. Each bit in the bitmask represents whether a city is in the subset (1) or not (0). This allows the algorithm to quickly and efficiently iterate over all subsets of cities.

vi/ Limitations

- Despite its efficiency compared to the brute-force approach, the Held-Karp algorithm is still not practical for large instances of the TSP due to its exponential time and space complexity. For large instances, heuristic or approximation algorithms are typically used.