DAY 19:

ASSIGNMENT 2:

Task 2: Traveling Salesman Problem

Create a function int FindMinCost(int[,] graph)
that takes a 2D array representing the graph where graph[i][j] is the cost to travel from city i to city j. The function should return the minimum cost to visit all cities and return to the starting city.
Use dynamic programming for this solution.

ANSWER:

```
import java.util.Arrays;


public class TravelingSalesman {


    public static int FindMinCost(int[][] graph) {

        int n = graph.length;

        int[][] dp = new int[n][1 << n];

        for (int[] row : dp) {

            Arrays.fill(row, -1);

        }

        return tsp(0, 1, graph, dp);

    }


    private static int tsp(int currentCity, int visitedMask, int[][] graph, int[][] dp) {

        int n = graph.length;


        // Base case: all cities visited, return cost to go back to starting city

        if (visitedMask == (1 << n) - 1) {

            return graph[currentCity][0];

        }


        // Return already computed result
        if (dp[currentCity][visitedMask] != -1) {
            return dp[currentCity][visitedMask];
        }

        int minCost = Integer.MAX_VALUE;

        // Try to go to an unvisited city
        for (int nextCity = 0; nextCity < n; nextCity++) {
            if ((visitedMask & (1 << nextCity)) == 0) { // If nextCity is not visited
                int newVisitedMask = visitedMask | (1 << nextCity);
                int cost = graph[currentCity][nextCity] + tsp(nextCity, newVisitedMask, graph, dp);
                minCost = Math.min(minCost, cost);
            }
        }

        // Store and return the minimum cost
        dp[currentCity][visitedMask] = minCost;
        return minCost;
    }

    public static void main(String[] args) {
        // Example usage
        int[][] graph = {
            {0, 10, 15, 20},
            {10, 0, 35, 25},
            {15, 35, 0, 30},
            {20, 25, 30, 0}
        };

        System.out.println("The minimum cost to visit all cities and return to the starting city is: " +
FindMinCost(graph));
```

```
    }
}
```

Explanation:

1. Initialization:

- dp is a 2D array where dp[currentCity][visitedMask] represents the minimum cost to visit all cities in the subset represented by visitedMask starting from currentCity.
- visitedMask is a bitmask indicating the set of visited cities.

2. Recursive Function tsp:

- currentCity is the current city being visited.
- visitedMask keeps track of the cities visited so far.
- Base case: If all cities are visited (visitedMask == (1 << n) - 1), return the cost to return to the starting city.
- If the result for this state is already computed, return the cached result.
- Iterate over all cities, recursively compute the cost for unvisited cities, and update the minimum cost.

3. Main Function FindMinCost:

- Initializes the dp array and starts the TSP function from city 0 with the initial mask 1 (indicating that city 0 is visited).

4. Example Usage:

- The main method provides an example graph and prints the minimum cost to visit all cities and return to the starting city.