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.

|  |
| --- |
| **package** Day13\_14;  **public** **class** TravelingSalesmanProblem {  **public** **static** **int** findMinCost(**int**[][] graph) {  **int** n = graph.length;  **int**[][] dp = **new** **int**[1 << n][n];  **for** (**int** i = 0; i < (1 << n); i++) {  **for** (**int** j = 0; j < n; j++) {  dp[i][j] = Integer.***MAX\_VALUE***;  }  }  dp[1][0] = 0;  **for** (**int** mask = 1; mask < (1 << n); mask += 2) {  **for** (**int** i = 0; i < n; i++) {  **if** ((mask & (1 << i)) != 0) {  **for** (**int** j = 0; j < n; j++) {  **if** ((mask & (1 << j)) != 0 && i != j) {  dp[mask][i] = Math.*min*(dp[mask][i], dp[mask ^ (1 << i)][j] + graph[j][i]);  }  }  }  }  }  **int** minCost = Integer.***MAX\_VALUE***;  **for** (**int** i = 1; i < n; i++) {  minCost = Math.*min*(minCost, dp[(1 << n) - 1][i] + graph[i][0]);  }  **return** minCost;  }  **public** **static** **void** main(String[] args) {  **int**[][] graph = {  {0, 10, 15, 20},  {10, 0, 35, 25},  {15, 35, 0, 30},  {20, 25, 30, 0}  };  **int** minCost = *findMinCost*(graph);  System.***out***.println("Minimum cost to visit all cities: " + minCost);  }  } |