**UCID: 2024510001 BATCH: A NAME: Atharva Vasant Angre**

**EXPERIMENT NO: 07**

**EXPERIMENT TITLE:** To implement Branch and Bound Algorithm (Travelling Salesman)

**Objective:**

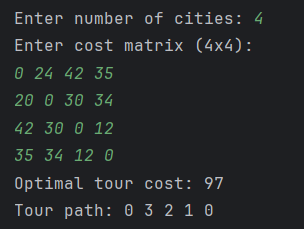
1.To learn how the Branch and Bound method works for solving TSP.

2.To find the shortest possible route by reducing extra calculations.

**Program code:** -

import java.util.\*;  
  
public class TSPBranchAndBound {  
 private int n;  
 private int[][] cost;  
 private boolean[] visited;  
 private int[] bestTour;  
 private int bestCost = Integer.*MAX\_VALUE*;  
  
 public static void main(String[] args) {  
  
 }  
  
 public void readInput() {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter number of cities: ");  
 n = sc.nextInt();  
 cost = new int[n][n];  
 System.*out*.println("Enter cost matrix (" + n + "x" + n + "):");  
 for (int i = 0; i < n; i++)  
 for (int j = 0; j < n; j++)  
 cost[i][j] = sc.nextInt();  
 visited = new boolean[n];  
 bestTour = new int[n + 1];  
 }  
  
 public void solve() {  
 // Start from city 0 arbitrarily  
 visited[0] = true;  
 ArrayList<Integer> path = new ArrayList<>();  
 path.add(0);  
 branchAndBound(0, 1, 0, path);  
 }  
  
 public void branchAndBound(int currentCity, int level, int currentCost, List<Integer> path) {  
 if (level == n) {  
 // Close the tour by returning to start  
 int totalCost = currentCost + cost[currentCity][0];  
 if (totalCost < bestCost) {  
 bestCost = totalCost;  
 for (int i = 0; i < n; i++) bestTour[i] = path.get(i);  
 bestTour[n] = 0;  
 }  
 return;  
 }  
 // Try all next cities  
 for (int next = 0; next < n; next++) {  
 if (!visited[next]) {  
 int tempCost = currentCost + cost[currentCity][next];  
 if (tempCost < bestCost) {  
 visited[next] = true;  
 path.add(next);  
 branchAndBound(next, level + 1, tempCost, path);  
 path.remove(path.size() - 1);  
 visited[next] = false;  
 }  
 }  
 }  
 }  
  
 public void printResult() {  
 System.*out*.println("Optimal tour cost: " + bestCost);  
 System.*out*.print("Tour path: ");  
 for (int city : bestTour) System.*out*.print(city + " ");  
 System.*out*.println();  
 }  
}

**Output:**

****

**Conclusion:**

In this experiment, we successfully implemented the Branch and Bound algorithm to solve the Travelling Salesman Problem (TSP). The algorithm efficiently explored all possible tours while pruning branches that could not yield better results than the current best solution. This helped in reducing the number of computations compared to brute-force methods. Through this experiment, we gained a deeper understanding of optimization techniques and how Branch and Bound can be applied to solve complex combinatorial problems like TSP.