	Pimpri Chinchwad Education Trust's **Pimpri Chinchwad College of Engineering (PCCoE)** (An Autonomous Institute) Affiliated to Savitribai Phule Pune University (SPPU) ISO 21001:2018 Certified by TUV	
Department: Information Technology	**Academic Year**: 2025-26	**Semester:** V
DESIGN AND ANALYSIS OF ALGORITHMS LABORATORY		
Name : Avishkar Jadhav	**PRN** : 123B1F032	

ASSIGNMENT NO. 8

Aim:
To develop and implement a solution for the Travelling Salesman Problem (TSP) using the Branch and Bound technique, in order to determine the optimal route with the minimum total travel cost while visiting all cities exactly once and returning to the starting point.

Problem Statement:

Scenario: Optimizing Delivery Routes for a Logistics Company

A leading logistics company, SwiftShip, is responsible for delivering packages to multiple cities. To minimize fuel costs and delivery time, the company needs to find the shortest possible route that allows a delivery truck to visit each city exactly once and return to the starting point.

The company wants an optimized solution that guarantees the least cost route, considering:

- Varying distances between cities.
- Fuel consumption costs, which depend on road conditions.
- Time constraints, as deliveries must be completed within a given period.

Since there are N cities, a brute-force approach checking all (N-1)!permutations is infeasible for large N (e.g., 20+ cities). Therefore, you must implement an LC (Least Cost) Branch and Bound algorithm to find the optimal route while reducing unnecessary computations efficiently.

Code:

```
#include <bits/stdc++.h>
using namespace std;
#define INF 9999999
```

```
struct Node {
    int level;
    int pathCost;
    int reducedCost;
    int vertex;
    vector<int> path;
    vector<vector<int>> reducedMatrix;

    bool operator<(const Node &other) const {
        return reducedCost > other.reducedCost; // for min-heap
    }
};

// Copy matrix
void copyMatrix(const vector<vector<int>> &src, vector<vector<int>> &dest) {
    dest = src;
}

// Reduce the cost matrix and return the reduction cost
int reduceMatrix(vector<vector<int>> &matrix, int n) {
    int reductionCost = 0;

    // Row reduction
    for (int i = 0; i < n; i++) {
        int rowMin = INF;
        for (int j = 0; j < n; j++)
            rowMin = min(rowMin, matrix[i][j]);

        if (rowMin != INF && rowMin != 0) {
            reductionCost += rowMin;
            for (int j = 0; j < n; j++)
                if (matrix[i][j] != INF)
                    matrix[i][j] -= rowMin;
        }
    }
```

```
    // Column reduction
    for (int j = 0; j < n; j++) {
        int colMin = INF;
        for (int i = 0; i < n; i++)
            colMin = min(colMin, matrix[i][j]);

        if (colMin != INF && colMin != 0) {
            reductionCost += colMin;
            for (int i = 0; i < n; i++)
                if (matrix[i][j] != INF)
                    matrix[i][j] -= colMin;
        }
    }

    return reductionCost;
}

// Create child node
Node createNode(vector<vector<int>> parentMatrix, vector<int> path, int level, int i, int j, int n) {
    Node node;
    node.reducedMatrix = parentMatrix;

    for (int k = 0; level != 0 && k < n; k++)
        node.reducedMatrix[i][k] = INF;

    for (int k = 0; k < n; k++)
        node.reducedMatrix[k][j] = INF;

    if (level + 1 < n)
        node.reducedMatrix[j][0] = INF;

    node.path = path;
    node.path.push_back(j);
    node.level = level;
    node.vertex = j;
    return node;
```

```
}

// Solve TSP
void solveTSP(vector<vector<int>> costMatrix, int n) {
    priority_queue<Node> pq;
    vector<int> path = {0};

    Node root;
    root.reducedMatrix = costMatrix;
    root.path = path;
    root.level = 0;
    root.vertex = 0;
    root.pathCost = 0;
    root.reducedCost = reduceMatrix(root.reducedMatrix, n);

    pq.push(root);
    int minCost = INF;
    vector<int> finalPath;

    while (!pq.empty()) {
        Node min = pq.top();
        pq.pop();

        int i = min.vertex;

        if (min.level == n - 1) {
            min.path.push_back(0);
            int totalCost = min.pathCost + costMatrix[i][0];
            if (totalCost < minCost) {
                minCost = totalCost;
                finalPath = min.path;
            }
            continue;
        }

        for (int j = 0; j < n; j++) {
```

```
            if (min.reducedMatrix[i][j] != INF) {
                Node child = createNode(min.reducedMatrix, min.path, min.level + 1, i, j, n);
                child.pathCost = min.pathCost + costMatrix[i][j];
                child.reducedCost = child.pathCost + reduceMatrix(child.reducedMatrix, n);
                pq.push(child);
            }
        }
    }

    cout << "\nOptimal Delivery Route (SwiftShip): ";
    for (int v : finalPath) cout << v << " ";
    cout << "\nMinimum Total Delivery Cost: " << minCost << endl;
}

int main() {
    int n;
    cout << "Enter number of cities: ";
    cin >> n;

    vector<vector<int>> costMatrix(n, vector<int>(n));
    cout << "Enter cost matrix (use large number for no direct route):\n";
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            cin >> costMatrix[i][j];

    solveTSP(costMatrix, n);
    return 0;
}
```

Output:

```
Enter number of cities: 4
Enter cost matrix (use large number for no direct route):
9999999 10 15 20
10 9999999 35 25
15 35 9999999 30
20 25 30 9999999

Optimal Delivery Route (SwiftShip): 0 1 3 2 0
Minimum Total Delivery Cost: 80
```

Conclusion:

In this assignment, the **Travelling Salesman Problem (TSP)** was successfully implemented using the **Branch and Bound algorithm** to determine the most cost-effective route for visiting all cities exactly once and returning to the starting point. The algorithm efficiently explored different possible paths and eliminated non-optimal routes early, reducing the overall computation time compared to brute-force methods. The program produced the **optimal delivery route and minimum total cost**, demonstrating the effectiveness of Branch and Bound in solving complex **optimization and routing problems** commonly encountered in logistics and transportation systems.