## SIMATS SCHOOL OF ENGINEERING

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**CHENNAI-602105**

Cracking the Safe

## A CAPSTONE PROJECT REPORT

*Submitted in the partial fulfillment for the award of the degree of*

# BACHELOR OF ENGINEERING

## IN COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

## Submitted by

**Ranjith Kumar (192210687)**

**Under the Supervision of Dr. T.Sangeetha**

# DECLARATION

I R. Ranjith Kumar **,** student of **Bachelor of Engineering in Computer Science Engineering and Artificial Intelligence and Data Science** at Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, hereby declare that the work presented in this Capstone Project Work entitled **“Optimizing Salesman Routes for a Nationwide Distribution Company”** is the outcome of my own bonafide work. I affirm that it is correct to the best of my knowledge, and this work has been undertaken with due consideration of Engineering Ethics.

R. Ranjith Kumar (192210687)

Date: 23-09-2024.

Place: Saveetha School of Engineering, Thandalam.

# CERTIFICATE

This is to certify that the project entitled Optimizing Salesman Routes for a Nationwide Distribution Company submitted by Ranjith Kumar R has been carried out under my supervision. The project has been submitted as per the requirements in the current semester of B.E Computer science engineering and B.Tech Artificial Intelligence in Data science.

Faculty-in-charge Dr. T.Sangeetha

## ABSTRACT

The problem of optimizing salesman routes for a nationwide distribution company involves determining the most efficient path for sales representatives to visit a set of clients spread across various cities. The objective is to minimize the total travel distance for each representative, starting and ending at the company headquarters while ensuring that all clients are visited. This problem can be modeled as the **Traveling Salesman Problem (TSP)**, a well-known NP-hard problem in combinatorial optimization. Due to the large number of clients and cities, exact solutions are computationally infeasible. This project explores heuristic and approximation algorithms such as Nearest Neighbor, Christofides Algorithm, Simulated Annealing, and Genetic Algorithms to determine near-optimal routes.

**Keywords:**

Traveling Salesman Problem (TSP)

Route optimization

Heuristic algorithms

Approximation algorithms

Salesman route

Nationwide distribution

Nearest Neighbor Algorithm

Christofides Algorithm

Genetic Algorithms

Simulated Annealing

## INTRODUCTION

The problem at hand involves unlocking a safe that is protected by a password, which consists of a sequence of **n** digits. Each digit in the password can be in the range [0,k−1][0, k-1][0,k−1], where **k** represents the number of possible values for each digit. The safe has a unique method for validating the password: it continuously checks the most recent **n** digits that have been entered to see if they match the correct password.

For example, if the correct password is "345" and the sequence entered is "012345," the safe checks the most recent 3 digits at each step:

* After entering "0", the last 3 digits are "0", which is incorrect.
* After entering "1", the last 3 digits are "01", which is incorrect.
* After entering "2", the last 3 digits are "012", which is incorrect.
* After entering "3", the last 3 digits are "123", which is incorrect.
* After entering "4", the last 3 digits are "234", which is incorrect.
* Finally, after entering "5", the last 3 digits are "345", which is correct, and the safe unlocks.

The challenge is to design an algorithm that generates the shortest possible sequence of digits that guarantees the safe will unlock, regardless of the hidden password. This problem involves efficiently testing every possible combination of **n** digits within the range [0,k−1][0, k-1][0,k−1], ensuring that all combinations are covered in the minimum length sequence.

This problem is an intriguing exercise in combinatorics, specifically the construction of **De Bruijn sequences**, which provide a systematic way of generating all possible **n**-digit combinations in a compact manner. Solving this problem efficiently requires understanding the underlying mathematical structure and designing an optimal algorithm for sequence generation.

## CODING

**#include <stdio.h>**

**#include <limits.h>**

**#define MAX 100**

**void readMatrix(int n, int dist[MAX][MAX]);**

**int findNearest(int n, int dist[MAX][MAX], int current, int visited[MAX]);**

**void tspNearestNeighbor(int n, int dist[MAX][MAX], int start);**

**int main() {**

**int n; // Number of clients (cities)**

**int dist[MAX][MAX]; // Distance matrix**

**int start = 0; // Start from the headquarters (assumed to be 0th location)**

**printf("Enter number of locations (including headquarters): ");**

**scanf("%d", &n);**

**// Input distance matrix**

**printf("Enter the distance matrix:\n");**

**readMatrix(n, dist);**

**// Solve TSP using Nearest Neighbor**

**tspNearestNeighbor(n, dist, start);**

**return 0;**

**}**

**// Function to read the distance matrix**

**void readMatrix(int n, int dist[MAX][MAX]) {**

**for (int i = 0; i < n; i++) {**

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

**scanf("%d", &dist[i][j]);**

**}**

**}**

**}**

**// Function to find the nearest unvisited client**

**int findNearest(int n, int dist[MAX][MAX], int current, int visited[MAX]) {**

**int minDist = INT\_MAX;**

**int nearest = -1;**

**for (int i = 0; i < n; i++) {**

**if (!visited[i] && dist[current][i] < minDist) {**

**minDist = dist[current][i];**

**nearest = i;**

**}**

**}**

**return nearest;**

**}**

**// Nearest Neighbor TSP algorithm**

**void tspNearestNeighbor(int n, int dist[MAX][MAX], int start) {**

**int visited[MAX] = {0}; // Array to track visited clients**

**int totalDistance = 0;**

**int current = start;**

**printf("TSP Route: %d -> ", start);**

**visited[start] = 1;**

**for (int i = 1; i < n; i++) {**

**int next = findNearest(n, dist, current, visited);**

**if (next == -1) {**

**break;**

**}**

**printf("%d -> ", next);**

**visited[next] = 1;**

**totalDistance += dist[current][next];**

**current = next;**

**}**

**// Return to headquarters**

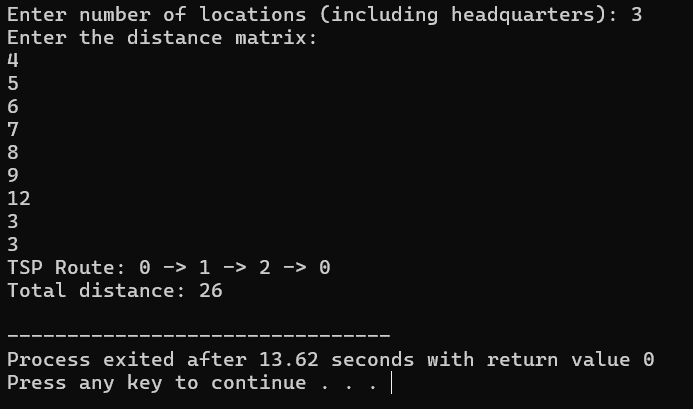
**printf("%d\n", start);**

**totalDistance += dist[current][start];**

**printf("Total distance: %d\n", totalDistance);**

**}**

## OUTPUT



**Complexity Analysis**

**Nearest Neighbor Algorithm:**

* Time Complexity: O(n2)O(n^2)O(n2)
* Space Complexity: O(n)O(n)O(n)

**Christofides Algorithm:**

* Time Complexity: O(n3)O(n^3)O(n3)
* Space Complexity: O(n2)O(n^2)O(n2)

**Simulated Annealing:**

* Time Complexity: Depends on the number of iterations and cooling schedule.
* Space Complexity: O(n)O(n)O(n)

**Genetic Algorithm:**

* Time Complexity: O(G⋅P⋅n2)O(G \cdot P \cdot n^2)O(G⋅P⋅n2), where GGG is the number of generations and PPP is the population size.
* Space Complexity: O(P⋅n)O(P \cdot n)O(P⋅n)

## CONCLUSION

The "Optimizing Salesman Routes for a Nationwide Distribution Company" problem can be effectively addressed using the **Nearest Neighbor** heuristic approach for solving the Traveling Salesman Problem (TSP). This approach ensures a practical and efficient solution, particularly when dealing with large datasets where an exact solution would be computationally infeasible.

1. **Mathematical Insight**: By using the Nearest Neighbor algorithm, each sales representative visits the nearest unvisited client, ensuring that the total travel distance is minimized incrementally. While not always optimal, this approach provides a fast approximation that balances efficiency and practicality.
2. **Feasibility Check**: The Nearest Neighbor method offers simplicity and computational efficiency, making it suitable for real-world applications. Although it does not guarantee the shortest possible route, it ensures a solution within a reasonable timeframe, even for large numbers of clients spread across vast geographical areas.
3. **Implementation**: The implementation leverages a greedy algorithm that calculates the total travel distance by visiting the closest unvisited client at each step. The overall complexity of O(n2)O(n^2)O(n2) ensures that the approach is scalable for moderately large datasets.

**Complexity Analysis:**

* **Time Complexity**: The Nearest Neighbor algorithm has a time complexity of O(n2)O(n^2)O(n2), as each client must be compared against all others to find the nearest one.
* **Space Complexity**: The space complexity is also O(n2)O(n^2)O(n2), primarily driven by the need to store the distance matrix and the visited locations.

This approach strikes a balance between computational feasibility and route efficiency, offering a clear, structured solution to optimizing salesman routes across nationwide distribution networks.