S. B. JAIN INSTITUTE OF TECHNOLOGY,

MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 5

Aim: Develop a program that finds the optimal solution for the Traveling Salesperson problem using dynamic programming.

Name of Student: Shrutika Pradeep Bagdi

Roll No: CS22130

Semester/Year: V/III

Academic Session:2024-2025

Date of Performance:

Date of Submission:

AIM: Develop a program that finds the optimal solution for the Traveling Salesperson problem using dynamic programming.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- To understand the concepts of Travelling Salesperson Problem.
- To implement the Travelling Salesperson Problem using dynamic programming.

THEORY:

Dynamic programming is a technique that breaks the problems into sub-problems, and saves the result for future purposes so that we do not need to compute the result again. The subproblems are optimized to optimize the overall solution is known as optimal substructure property. The main use of dynamic programming is to solve optimization problems. Here, optimization problems mean that when we are trying to find out the minimum or the maximum solution of a problem. The dynamic programming guarantees to find the optimal solution of a problem if the solution exists.

The definition of dynamic programming says that it is a technique for solving a complex problem by first breaking into a collection of simpler subproblems, solving each subproblem just once, and then storing their solutions to avoid repetitive computations.

Travelling Salesperson Problem:

Given a set of cities and the distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.

Salespersons are required to travel frequently as a part of their job, with the aim of maximizing their business opportunities. However, their daily schedules are often subject to sudden changes due to scheduled and last-minute customer appointments, making it difficult for them to plan optimal travel routes. As a result, salespersons are forced to take longer routes to reach their destinations, which can result in missed appointments and lost business opportunities.

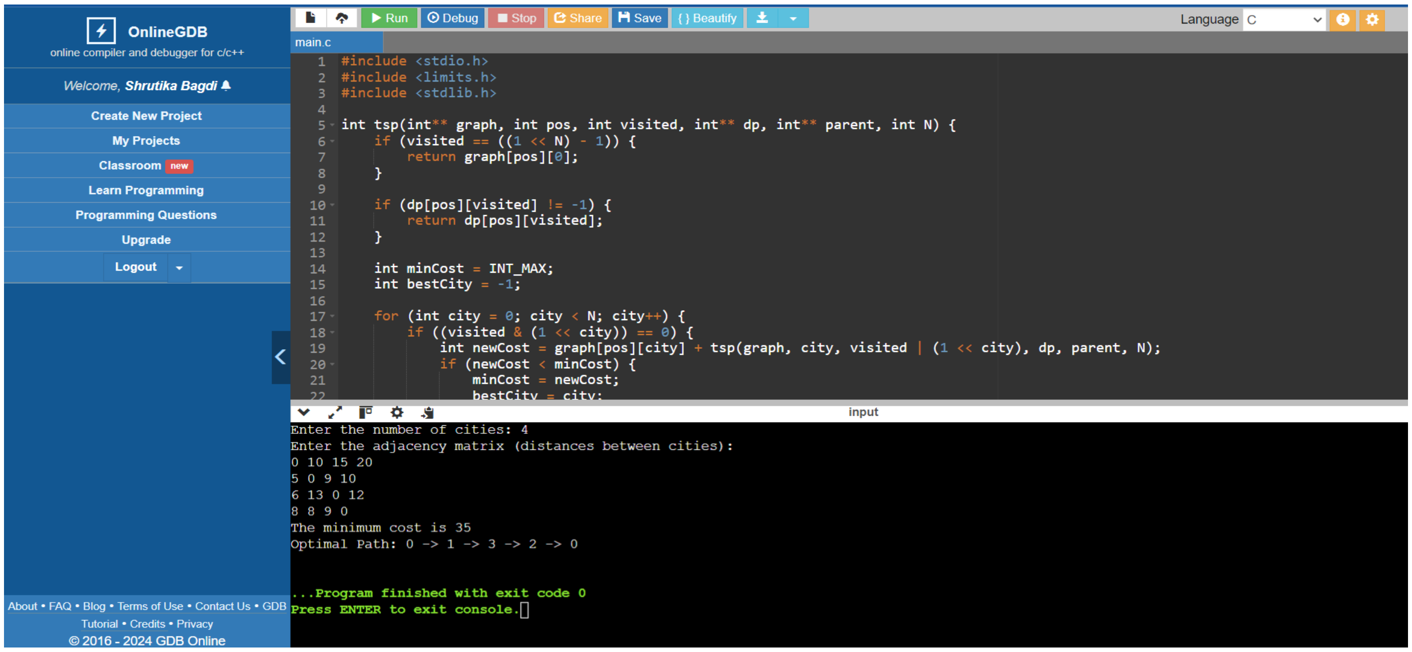
- The TSP problem is **highly applicable in the logistics sector**, particularly in route planning and optimization for delivery services. TSP solving algorithms help to reduce travel costs and time.

- **Real-world applications often require adaptations because they involve additional constraints** like time windows, vehicle capacity, and customer preferences.

CODE:

```
#include <stdio.h>
#include <limits.h>
#include <stdlib.h>
int tsp(int** graph, int pos, int visited, int** dp, int** parent, int N) {
    if (visited == ((1 << N) - 1)) {
        return graph[pos][0];
    }
    if (dp[pos][visited] != -1) {
        return dp[pos][visited];
    }
    int minCost = INT_MAX;
    int bestCity = -1;
    for (int city = 0; city < N; city++) {
        if ((visited & (1 << city)) == 0) {
            int newCost = graph[pos][city] + tsp(graph, city, visited | (1 << city), dp, parent, N);
            if (newCost < minCost) {
                minCost = newCost;
                bestCity = city;
            }
        }
    }
    parent[pos][visited] = bestCity;
    return dp[pos][visited] = minCost;
}
void printPath(int** parent, int N) {
    int visited = 1;
    int city = 0;
    printf("Optimal Path: 0 -> ");
    while (visited != (1 << N) - 1) {
        city = parent[city][visited];
        printf("%d -> ", city);
        visited |= (1 << city);
    }
    printf("0\n");
}
int main() {
    int N;
    printf("Enter the number of cities: ");
    scanf("%d", &N);
    int** graph = (int**)malloc(N * sizeof(int*));
    for (int i = 0; i < N; i++) {
        graph[i] = (int*)malloc(N * sizeof(int));
    }
    int** dp = (int**)malloc(N * sizeof(int*));
    int** parent = (int**)malloc(N * sizeof(int*));
    for (int i = 0; i < N; i++) {
        dp[i] = (int*)malloc((1 << N) * sizeof(int));
        parent[i] = (int*)malloc((1 << N) * sizeof(int));
    }
    printf("Enter the adjacency matrix (distances between cities):\n");
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            scanf("%d", &graph[i][j]);
        }
    }
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < (1 << N); j++) {
            dp[i][j] = -1;
            parent[i][j] = -1;
        }
    }
    int result = tsp(graph, 0, 1, dp, parent, N);
    printf("The minimum cost is %d\n", result);
    printPath(parent, N);
    for (int i = 0; i < N; i++) {
        free(graph[i]);
        free(dp[i]);
        free(parent[i]);
    }
    free(graph);
    free(dp);
    free(parent);
    return 0;
}
```

INPUT & OUTPUT WITH DIFFERENT TEST CASES:



```
#include <stdio.h>
#include <limits.h>
#include <stdlib.h>

int tsp(int** graph, int pos, int visited, int** dp, int** parent, int N) {
    if (visited == ((1 << N) - 1)) {
        return graph[pos][0];
    }

    if (dp[pos][visited] != -1) {
        return dp[pos][visited];
    }

    int minCost = INT_MAX;
    int bestCity = -1;

    for (int city = 0; city < N; city++) {
        if ((visited & (1 << city)) == 0) {
            int newCost = graph[pos][city] + tsp(graph, city, visited | (1 << city), dp, parent, N);
            if (newCost < minCost) {
                minCost = newCost;
                bestCity = city;
```

```
Enter the number of cities: 4
Enter the adjacency matrix (distances between cities):
0 10 15 20
5 0 9 10
6 13 0 12
8 8 9 0
The minimum cost is 35
Optimal Path: 0 -> 1 -> 3 -> 2 -> 0


...Program finished with exit code 0
Press ENTER to exit console.
```

CONCLUSION:

DISCUSSION AND VIVA VOCE:

- Explain the Travelling Salesperson Problem.
- Discuss the dynamic programming approach..
- Discuss the complexity of Travelling Salesperson Problem.

REFERENCES:

- *https://www.geeksforgeeks.org/travelling-salesman-problem-using-dynamic-programming/*
- *https://www.tutorialspoint.com/data_structures_algorithms/travelling_salesman_problem_dynamic_programming.htm*
- *https://www.javatpoint.com/travelling-salesman-problem*
- *https://www.baeldung.com/cs/tsp-dynamic-programming*