VISHWAKARMA INSTITUTES VI	Bansilal Ramnath Agarwal Charitable Trust's Vishwakarma Institute of Information Technology **Department of** **Artificial Intelligence and Data Science**	
Name: Siddhesh Dilip Khairnar		
Class: TY	Division: B	Roll No: 372028
Semester: V	Academic Year: 2023-24	
Subject Name & Code: Design and Analysis of Algorithm: ADUA31202		
Title of Assignment: Implement any scheme to find the optimal solution for the Traveling Salesperson problem and then solve the same problem instance using different algorithmic strategies and determine the optimal solution.		

ASSIGNMENT NO. 8

DAA Assignment 08

Name: Siddhesh Dilip Khairnar
Roll no: 372028
PRN no: 22110398

Aim: ~~A~~ Implement any scheme to find the optimal solution for the Traveling salesperson problem and then solve the same problem instance using different algorithmic strategies and determine the optimal solution.

Learning objective:
1. Gain knowledge about the ~~Fe~~ Traveling salesperson problem (TSP) and its status as an NP-hard problem, which is a class of difficult
2. Learn various algorithm strategies for tackling the TSP, including exact algorithm like brute force & dynamic programming, approximation algorithm like nearest neighbor and christofides, and optimization algorithm like genetic algorithm.

Theory: →
TSP is an NP-hard problem, and finding the optimal solution for large instances can be computationally intensive. Here, I'll outline a common approaches and some algorithmic strategies to solve it.

1) Exact algorithm:
A. Brute force: The most straightforward method is to calculate the distance for all possible permutation of city order. This guarantees the optimal solⁿ, but it's not feasible for large instances due to its factorial time complexity.
B. Dynamic programming: The Held-karp algorithm is a dynamic programming approach that optimizes the computation of subproblem solution. It's more efficient than brute force but still has high time complexity.

Page No.
Date

2. Approximation Algorithm : →
a). Nearest Neighbor : start from an arbitrary city and at each step, choose the nearest unvisited city ~~until~~ until all cities are visited. It's a simple & fast heuristic but doesn't guarantee optimality
b) Christofides Algorithm : This is more advanced heuristic that provides a solution guarantee to be within 3/2 times the optimal solution for TSP instances with Euclidean distances.

Algorithm for Dynamic programming approach : →
If size of S is 2, then S must be {1, i}
C(S, i) = dist(1, i)
Else if size of S is greater than 2.
C(S, i) = min{C(S-{i}, j) + dist (j, i)} where j belong to S, j != i & j != 1.

Algorithm for BackTracking Approach : →
- Consider city1 as the starting & ending point, since route is cyclic, we can consider any point as starting point.
- Start traversing from the source to its adjacent nodes in dfs manner.
- Calculate cost of every traversal & keep track of minimum cost & keep on updating the value of minimum cost stored value.
- Return the permutation with minimum cost.

Conclusion : In this assignment we successfully found the optimal soln for the Travelling salesperson problem. We then solved the same problem instance using Dynamic programming approach & Backtracking approach & determined the optimal soln.

Aim:
To implement any scheme to find the optimal solution for the Traveling Salesperson problem and then solve the same problem instance using different algorithmic strategies and determine the optimal solution.

Problem Statement:
Implement any scheme to find the optimal solution for the Traveling Salesperson problem and then solve the same problem instance using different algorithmic strategies and determine the optimal solution.

Software Requirements:
1. Text Editor: VS Code
2. Online GDB Compiler
3. Environment: GCC C++

Background Information:
The traveling salesman problems abide by a salesman and a set of cities. The salesman must visit every one of the cities starting from a certain one (e.g., the hometown) and to return to the same city. The challenge of the problem is that the traveling salesman needs to minimize the total length of the trip.

Suppose the cities are x_1 x_2..... x_n where cost c_{ij} denotes the cost of travelling from city x_i to x_j. The travelling salesperson problem is to find a route starting and ending at x_1 that will take in all cities with the minimum cost.

Using Dynamic Programming Approach:

Let the given set of vertices be {1, 2, 3, 4,….n}. Let us consider 1 as starting and ending point of output. For every other vertex I (other than 1), we find the minimum cost path with 1 as the starting point, I as the ending point, and all vertices appearing exactly once. Let the cost of this path cost (i), and the cost of the corresponding Cycle would cost (i) + dist(i, 1) where dist(i, 1) is the distance from I to 1. Finally, we return the minimum of all [cost(i) + dist(i, 1)] values. This looks simple so far.

Let us define a term C(S, i) be the cost of the minimum cost path visiting each vertex in set S exactly once, starting at 1 and ending at i. We start with all subsets of size 2 and calculate C(S, i) for all subsets where S is the subset, then we calculate C(S, i) for all subsets S of size 3 and so on. Note that 1 must be present in every subset.

If size of S is 2, then S must be {1, i},

C(S, i) = dist(1, i)

Else if size of S is greater than 2.

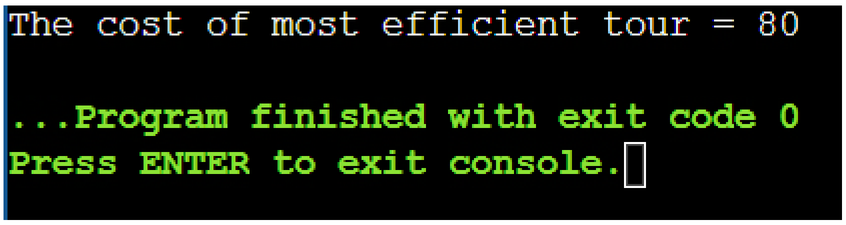
C(S, i) = min { C(S-{i}, j) + dist(j, i)} where j belongs to S, j != i and j != 1.

Algorithm:

```
If size of S is 2, then S must be {1, i},
 C(S, i) = dist(1, i)
Else if size of S is greater than 2.
 C(S, i) = min { C(S-{i}, j) + dist(j, i)} where j belongs to S, j != i and j != 1.
```

Program Code:

Output:



```
The cost of most efficient tour = 80

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

Using Back Tracking Approach:

Algorithm:

Consider city 1 (let say 0th node) as the starting and ending point. Since route is cyclic, we can consider any point as starting point.

Start traversing from the source to its adjacent nodes in dfs manner.

Calculate cost of every traversal and keep track of minimum cost and keep on updating the value of minimum cost stored value.

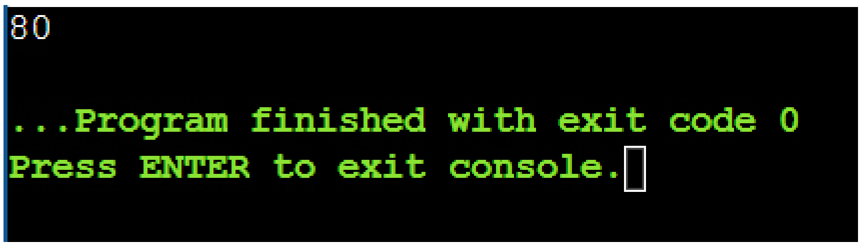
Return the permutation with minimum cost.

Program Code:

```cpp
#include <bits/stdc++.h>
using namespace std;
#define V 4

void tsp(int graph[][V], vector<bool>& v, int currPos,
        int n, int count, int cost, int& ans)
{

    if (count == n && graph[currPos][0]) {
        ans = min(ans, cost + graph[currPos][0]);
        return;
    }

    for (int i = 0; i < n; i++) {
        if (!v[i] && graph[currPos][i]) {


            v[i] = true;
            tsp(graph, v, i, n, count + 1,
                cost + graph[currPos][i], ans);
            v[i] = false;
        }
    }
};


int main()
{

    int n = 4;

    int graph[][V] = {
        { 0, 10, 15, 20 },
        { 10, 0, 35, 25 },
        { 15, 35, 0, 30 },
        { 20, 25, 30, 0 }
    };

    vector<bool> v(n);
    for (int i = 0; i < n; i++)
```

```
        v[i] = false;

    v[0] = true;
    int ans = INT_MAX;
    tsp(graph, v, 0, n, 1, 0, ans);
    cout << ans;
    return 0;
}
```

Output:



```
80

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

Conclusion:

In this assignment we successfully found the optimal solution for the Traveling Salesperson problem. We then solved the same problem instance using Dynamic Programming Approach and Backtracking Approach and determined the optimal solution.