6. Traveling salesman problem

time complexity: O(n!)

Algorithm:

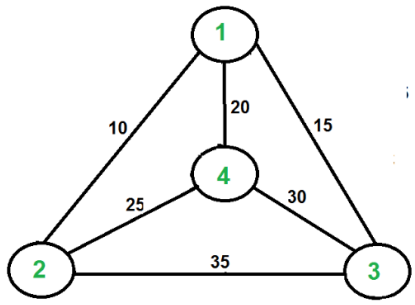
1. Initialize an array **'tour'** to store the sequence of cities visited in the tour.
   Set the first city as the starting city **(0).**
2. Initialize an array **'visited'** to keep track of visited cities.
   Initialize all elements to **0**, indicating **unvisited**.
   (i). For each city **'i'** from 1 to **'n - 1'**:
      a. Set **'minCity'** to **infinity** (INF) and **'minIndex'** to **-1**.
      b. For each **unvisited** city **'j'**:
         - If the distance from the last visited city to **'j'** is **less** than **'minCity'**:
            - Update 'minCity' to the distance from the last city to 'j'.
            - Update 'minIndex' to 'j'.
      c. **Add** the nearest unvisited city ('minIndex') to the tour.
      d. Mark the city **'minIndex'** as **visited**.
   (ii). Print the optimal tour sequence stored in the 'tour' array.
      The tour starts and ends at the starting city (0).
3. Read the number of cities **'n'** from the user.
4. **Read** the **adjacency matrix** representing distances between cities.
5. **Call** the **tsp() functio**n with the graph and 'n' as parameters.
6. Exit.

Code:

```
#include <stdio.h>
#define MAX_CITIES 10
#define INF 9999

void tsp(int graph[MAX_CITIES][MAX_CITIES], int n)
{
   int tour[MAX_CITIES];
   int visited[MAX_CITIES] = {0}; // Initializing all to 0 means false

   tour[0] = 0; // Start from the first city
   int totalCost = 0;

   for (int i = 1; i < n; i++)
         {
      int minCost = INF;
      int minIndex = -1;
```

```
        for (int j = 0; j < n; j++)
                    {
            if (!visited[j] && graph[tour[i - 1]][j] < minCost)
                                {
                minCost = graph[tour[i - 1]][j];
                minIndex = j;
            }
        }

        tour[i] = minIndex;
        visited[minIndex] = 1; // Mark as visited
        totalCost += minCost;
    }
    // Complete the tour by returning to the starting city
    totalCost += graph[tour[n - 1]][tour[0]];
    // Print the tour
    printf("Optimal Tour: ");
    for (int i = 0; i < n; i++)
        printf("%d -> ", tour[i]);
    printf("%d\n", tour[0]); // Back to the starting city

    // Print the total cost
    printf("Total Cost: %d\n", totalCost);
}

int main()
{
    int n;
    printf("Enter the number of cities: ");
    scanf("%d", &n);

    int graph[MAX_CITIES][MAX_CITIES];
    printf("Enter the weighted adjacency matrix (use INF for unreachable cities):\n");
    for (int i = 0; i < n; i++)
            {
        for (int j = 0; j < n; j++)
                    {
            scanf("%d", &graph[i][j]);
        }
    }
  tsp(graph, n);
    return 0;
}
```

Graph:



Input/output:

```
Enter the number of cities: 4
Enter the weighted adjacency matrix (use INF for unreachable cities):
0 10 15 20
10 0 35 25
15 35 0 30
20 25 30 0
Optimal Tour: 0 -> 1 -> 3 -> 2 -> 0
Total Cost: 80
```