Name	Stephen David Vaz
UID no.	2021700070
Experiment No.	9

AIM:	Approximation algorithms
PROBLEM STATEMENT :	Travelling Salesman Problem
ALGORITHM/ THEORY:	The Traveling Salesman Problem (TSP) is a well-known optimization problem in computer science. It involves finding the shortest possible route that visits each of a given set of cities exactly once and returns to the starting city. One approach to solving the TSP is to use an approximation algorithm. Approximation algorithms provide solutions that are guaranteed to be within a certain factor of the optimal solution. Algorithm: 1. Calculate the minimum spanning tree (MST) of the given graph. 2. Perform a pre-order traversal of the MST, starting from the root node. 3. During the traversal, visit each node exactly once and add it to the tour. 4. Once all nodes have been visited, return to the root node to complete the tour. 5. Calculate the minimum cost by summing up the weights of all the edges in the tour. The time complexity of the MST-based algorithm for the TSP depends on the time complexity of the MST algorithm used to calculate the minimum spanning tree of the input graph. If we use a standard algorithm like Prim's or Kruskal's algorithm to compute the MST, the time complexity of the algorithm would be O(ElogV), where E is the number of edges and V is the number of vertices in the input graph. Performing a pre-order traversal of the MST and visiting each node once takes O(V) time.

	So the overall time complexity of the algorithm would be O(ElogV + V), which is dominated by the time complexity of the MST algorithm. Therefore, the time complexity of the MST-based algorithm for the TSP is typically considered to be O(ElogV).
PROGRAM:	

```c
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#define INF 9999999
#define V 4

int adjG[10][10], visited[10], n, cost = 0;

int min(int c)
{
   int i, nc = 999;
   int min = 999, kmin;

   for (i = 0; i < n; i++)
   {
       if ((adjG[c][i] != 0) && (visited[i] == 0))
           if (adjG[c][i] + adjG[i][c] < min)
           {
               min = adjG[i][0] + adjG[c][i];
               kmin = adjG[c][i];
               nc = i;
           }
   }

   if (min != 999)
       cost += kmin;

   return nc;
}

void userInput()
{
   int i, j;

   printf("Number of villages: ");
```

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

    printf("\nAdjacency Matrix:\n");

    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
            scanf("%d", &adjG[i][j]);

        visited[i] = 0;
    }
}

void tsp(int city)
{
    int i, ncity;

    visited[city] = 1;
    ncity = min(city);

    if (ncity == 999)
    {
        ncity = 0;
        cost += adjG[city][ncity];
        return;
    }
    tsp(ncity);
}

void mst()
{
    int no_edge;
    int selected[V];

    memset(selected, false, sizeof(selected));

    no_edge = 0;

    int sel[10];
    int selz[10];
```

```
selected[0] = true;

int x;
int y;
int z = 0;
int zo = 0;
while (no_edge < V - 1)
{
    int min = INF;
    x = 0;
    y = 0;

    for (int i = 0; i < V; i++)
    {
        if (selected[i])
        {
            for (int j = 0; j < V; j++)
            {
                if (!selected[j] && adjG[i][j])
                {
                    if (min > adjG[i][j])
                    {
                        min = adjG[i][j];
                        x = i;
                        y = j;
                    }
                }
            }
        }
    }

    sel[z] = x + 1;
    selz[zo] = y + 1;
    z++;
    zo++;
    selected[y] = true;
    no_edge++;
}
sel[z] = sel[0];
z++;
for (int i = 0; i < z; i++)
```

```
    {
        if (i == zo)
            printf("%d -> %d", selz[zo - 1], sel[i]);
        else
            printf("%d -> ", sel[i]);
    }
}

int main()
{
    userInput();
    printf("\nRoute:\n");
    mst();
    tsp(0);
    printf("\n\nMinimum cost is %d\n", cost);
    return 0;
}
```

RESULT:

```
 *  Executing task: /usr/bin/clang /Users/stephe

Number of villages: 4

Adjacency Matrix:
0 4 1 3
4 0 2 1
1 2 0 5
3 1 5 0

Route:
1 -> 3 -> 2 -> 4 -> 1

Minimum cost is 7
```

CONCLUSION:	Successfully understood travelling salesman problem in C using Approximation Algorithm