|  |  |
| --- | --- |
|  | |
| Assignment No: 07     1. A program of Travelling Salesman Problem using Branch & Bound. | |
| Date of Performance: 01/09/2019  Date of Submission: 08/09/2019 | Student ID: 17-02-04-058  Group: B1 |

**No.1:**

#include <bits/stdc++.h>

using namespace std;

const int N = 4;

int final\_path[N+1];

bool visited[N];

int final\_res = INT\_MAX;

void copyToFinal(int curr\_path[])

{

for (int i=0; i<N; i++)

final\_path[i] = curr\_path[i];

final\_path[N] = curr\_path[0];

}

int firstMin(int adj[N][N], int i)

{

int min = INT\_MAX;

for (int k=0; k<N; k++)

if (adj[i][k]<min && i != k)

min = adj[i][k];

return min;

}

int secondMin(int adj[N][N], int i)

{

int first = INT\_MAX, second = INT\_MAX;

for (int j=0; j<N; j++)

{

if (i == j)

continue;

if (adj[i][j] <= first)

{

second = first;

first = adj[i][j];

}

else if (adj[i][j] <= second &&

adj[i][j] != first)

second = adj[i][j];

}

return second;

}

void TSPRec(int adj[N][N], int curr\_bound, int curr\_weight,

int level, int curr\_path[])

{

if (level==N)

{

if (adj[curr\_path[level-1]][curr\_path[0]] != 0)

{

int curr\_res = curr\_weight +

adj[curr\_path[level-1]][curr\_path[0]];

if (curr\_res < final\_res)

{

copyToFinal(curr\_path);

final\_res = curr\_res;

}

}

return;

}

for (int i=0; i<N; i++)

{

if (adj[curr\_path[level-1]][i] != 0 &&

visited[i] == false)

{

int temp = curr\_bound;

curr\_weight += adj[curr\_path[level-1]][i];

if (level==1)

curr\_bound -= ((firstMin(adj, curr\_path[level-1]) +

firstMin(adj, i))/2);

else

curr\_bound -= ((secondMin(adj, curr\_path[level-1]) +

firstMin(adj, i))/2);

if (curr\_bound + curr\_weight < final\_res)

{

curr\_path[level] = i;

visited[i] = true;

TSPRec(adj, curr\_bound, curr\_weight, level+1,

curr\_path);

}

curr\_weight -= adj[curr\_path[level-1]][i];

curr\_bound = temp;

memset(visited, false, sizeof(visited));

for (int j=0; j<=level-1; j++)

visited[curr\_path[j]] = true;

}

}

}

void TSP(int adj[N][N])

{

int curr\_path[N+1];

int curr\_bound = 0;

memset(curr\_path, -1, sizeof(curr\_path));

memset(visited, 0, sizeof(curr\_path));

for (int i=0; i<N; i++)

curr\_bound += (firstMin(adj, i) +

secondMin(adj, i));

curr\_bound = (curr\_bound&1)? curr\_bound/2 + 1 :

curr\_bound/2;

visited[0] = true;

curr\_path[0] = 0;

TSPRec(adj, curr\_bound, 0, 1, curr\_path);

}

int main()

{

//Adjacency matrix for the given graph

int adj[N][N] =

{ {0, 10, 15, 20},

{10, 0, 35, 25},

{15, 35, 0, 30},

{20, 25, 30, 0}

};

TSP(adj);

printf("Minimum cost : %d\n", final\_res);

printf("Path Taken : ");

for (int i=0; i<=N; i++)

printf("%d ", final\_path[i]);

return 0;}