

**Assignment-8**

**Name: - Mann Kothari**

**Reg. No.: - 22BCB7064**

**Course Title: - Design and Analysis of Algorithm (Embedded Lab)**

**Course Code: - CSE3023**

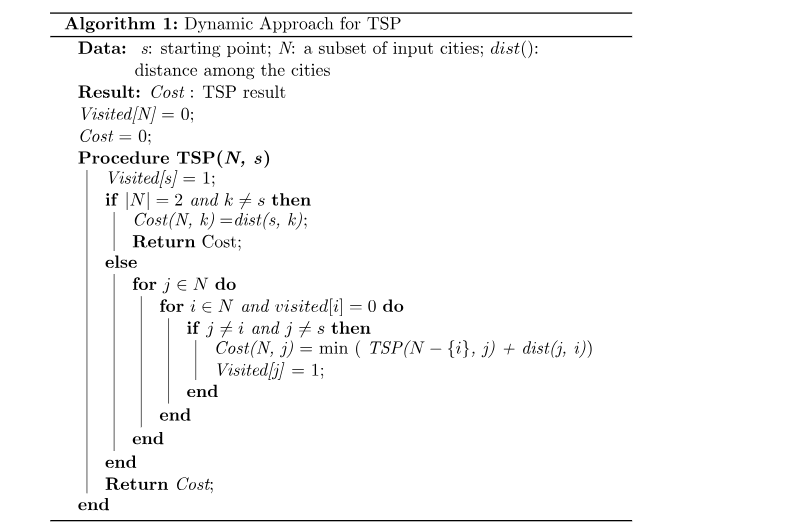
**Slot: - L21+L22**

**Submitted to: - Prof. Tanikella Divya Naga Pavani**

* **INDEX**
* Travelling SalesPerson(TSP) Problem using Dynamic Programming.
* Algorithm
* Code
* Output
* Complexity
* All Pairs Shortest Paths( Floyd's- Warshall Algorithm) Problem using Dynamic Programming.
* Algorithm
* Code
* Output
* Complexity
* Complexity Chart

1. Write a C/C++ program to Implement the Travelling SalesPerson(TSP) Problem using Dynamic Programming.

Algorithm: -



Time Complexity: -

O(n2\*2n)

Code: -

#include <stdio.h>

#define n 4

#define MAX 10000

int dist[n + 1][n + 1] = {

    {0, 0, 0, 0, 0},

    {0, 0, 10, 15, 20},

    {0, 10, 0, 25, 25},

    {0, 15, 25, 0, 30},

    {0, 20, 25, 30, 0},

};

int memo[n + 1][1 << (n + 1)];

int min(int a, int b) { return a < b ? a : b; }

int fun(int i, int mask)

{

    if (mask == ((1 << i) | 3))

        return dist[1][i];

    if (memo[i][mask] != 0)

        return memo[i][mask];

    int res = MAX;

    for (int j = 1; j <= n; j++)

        if ((mask & (1 << j)) && j != i && j != 1)

            res = min(res, fun(j, mask & (~(1 << i))) + dist[j][i]);

    return memo[i][mask] = res;

}

int main()

{

    int ans = MAX;

    for (int i = 1; i <= n; i++)

        ans = min(ans, fun(i, (1 << (n + 1)) - 1) + dist[i][1]);

    printf("The cost of most efficient tour = %d", ans);

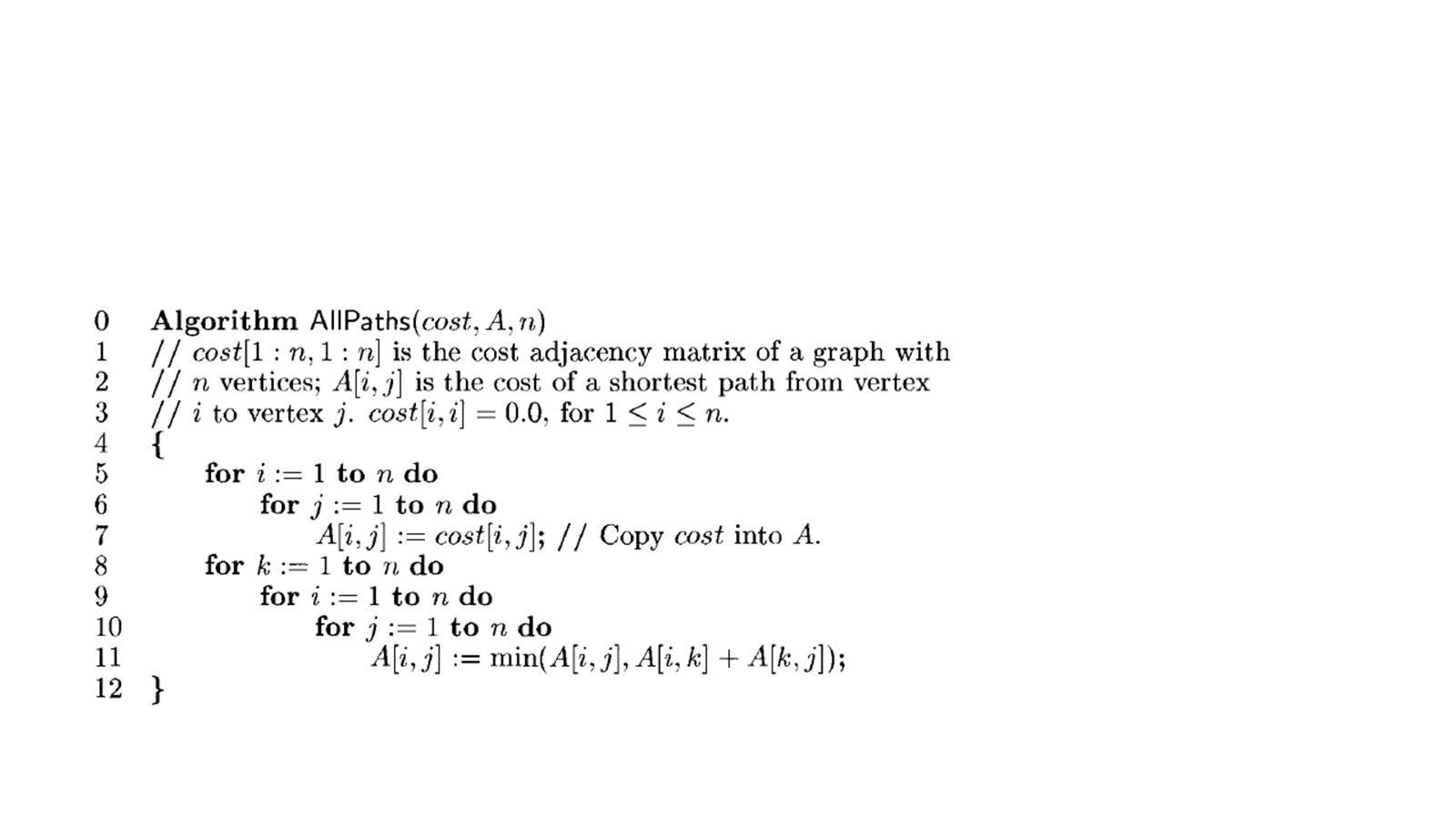
    return 0;

}

Output: -

1. Write a C/C++ program to Implement the  All Pairs Shortest Paths( Floyd's- Warshall Algorithm) Problem using Dynamic Programming.

Algorithm: -



Time Complexity: -

O(n3)

Code: -

#include <stdio.h>

#define V 4

#define INF 99999

void printSolution(int dist[][V])

{

    printf("The following matrix shows the shortest distances"

           " between every pair of vertices \n");

    for (int i = 0; i < V; i++)

    {

        for (int j = 0; j < V; j++)

        {

            if (dist[i][j] == INF)

                printf("%7s", "INF");

            else

                printf("%7d", dist[i][j]);

        }

        printf("\n");

    }

}

void floydWarshall(int dist[][V])

{

    int i, j, k;

    for (k = 0; k < V; k++)

    {

        for (i = 0; i < V; i++)

        {

            for (j = 0; j < V; j++)

            {

                if (dist[i][k] + dist[k][j] < dist[i][j])

                    dist[i][j] = dist[i][k] + dist[k][j];

            }

        }

    }

    printSolution(dist);

}

int main()

{

    /\* Let us create the following weighted graph

         10

    (0)------->(3)

     |         /|\

    5|          |

     |          | 1

    \|/         |

    (1)------->(2)

          3      \*/

    int graph[V][V] = {{0, 5, INF, 10},

                       {INF, 0, 3, INF},

                       {INF, INF, 0, 1},

                       {INF, INF, INF, 0}};

    floydWarshall(graph);

    return 0;

}

Output: -

|  |  |  |
| --- | --- | --- |
| Algorithm | Time Complexity | Space Complexity |
| Travelling SalesPerson(TSP) Problem using Dynamic Programming. | O(n2\*2n) | O(n) |
| All Pairs Shortest Paths( Floyd's- Warshall Algorithm) Problem using Dynamic Programming. | O(n3) | O(n2) |