

**Assignment-9**

**Name: - Mann Kothari**

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

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

**Course Code: - CSE3023**

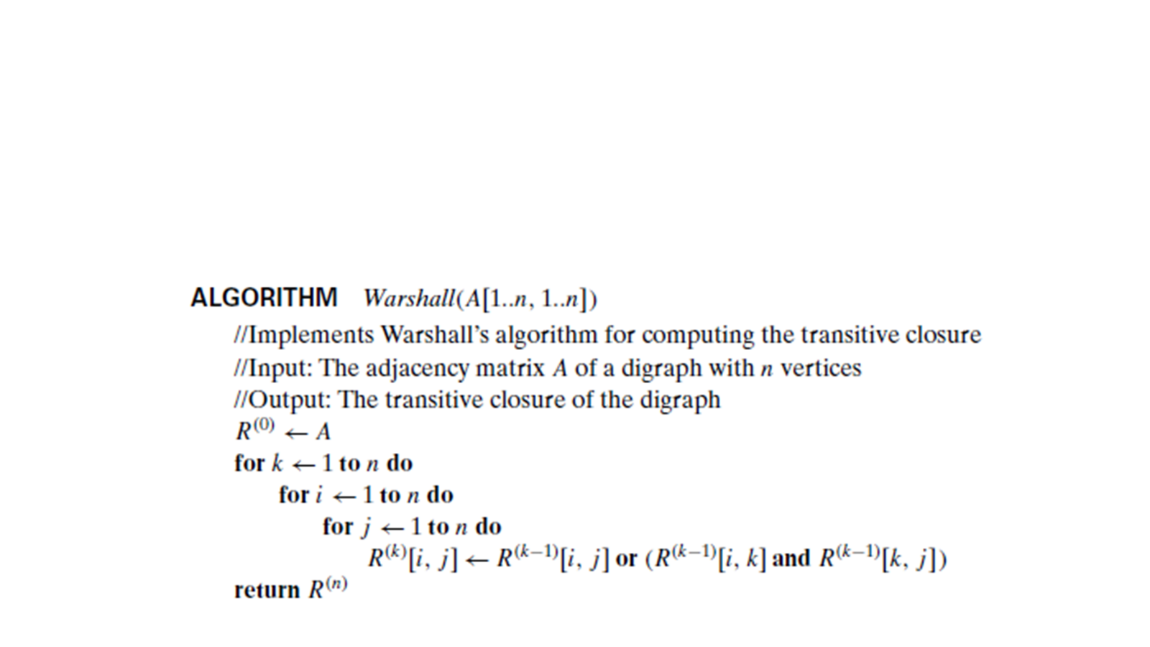
**Slot: - L21+L22**

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

* **INDEX**
* Warshall’s Algorithm
* Algorithm
* Code
* Output
* Complexity
* Optimal Binary Search Tree
* Algorithm
* Code
* Output
* Complexity
* Complexity Chart

1. Write a C/C++ program to Implement Warshal's Algorithm.

Algorithm: -



Time Complexity: -

O(n3)

Code: -

#include <iostream>

using namespace std;

#define V 4 // Number of vertices in the graph

void printMatrix(int reach[][V]) {

    for (int i = 0; i < V; i++) {

        for (int j = 0; j < V; j++) {

            cout << reach[i][j] << " ";

        }

        cout << endl;

    }

}

void transitiveClosure(int graph[][V]) {

    int reach[V][V];

    for (int i = 0; i < V; i++)

        for (int j = 0; j < V; j++)

            reach[i][j] = graph[i][j];

    for (int k = 0; k < V; k++) {

        for (int i = 0; i < V; i++) {

            for (int j = 0; j < V; j++) {

                reach[i][j] = reach[i][j] || (reach[i][k] && reach[k][j]);

            }

        }

    }

    printMatrix(reach);

}

int main() {

    int graph[V][V] = { {0, 1, 0, 0},

                        {0, 0, 0, 1},

                        {0, 0, 0, 0},

                        {1, 0, 1, 0} };

    transitiveClosure(graph);

    return 0;

}

Output: -

1. Write a C/C++ program that uses a dynamic programming algorithm to solve the optimal binary search tree problem

Algorithm: -

Algorithm optCost(freq, i, j){

    if (j < i)then

        return 0;

    if (j == i) then

        return freq[i];

    int fsum:= sum(freq, i, j);

    int min:= INT\_MAX;

    for r:= i to j do

    {

        int cost:= optCost(freq, i, r-1) + optCost(freq, r+1, j);

        if (cost < min)

            min:= cost;

    }

    return min + fsum;

}

Time Complexity: -

O(n3)

Code: -

#include <stdio.h>

#include <limits.>

#define INT\_MAX 100

int sum(int freq[], int i, int j)

{

    int s = 0;

    int k;

    for (k = i; k <=j; k++)

    s += freq[k];

    return s;

}

int optCost(int freq[], int i, int j){

    if (j < i)

        return 0;

    if (j == i)

        return freq[i];

    int fsum = sum(freq, i, j);

    int min = INT\_MAX;

    int r;

    for (r = i; r <= j; ++r)

    {

        int cost = optCost(freq, i, r-1) +

                    optCost(freq, r+1, j);

        if (cost < min)

            min = cost;

    }

    return min + fsum;

}

int optimalSearchTree(int keys[], int freq[], int n)

{

    return optCost(freq, 0, n-1);

}

int main()

{

    int keys[] = {10, 12, 20};

    int freq[] = {34, 8, 50};

    int n = sizeof(keys)/sizeof(keys[0]);

    printf("Cost of Optimal BST is %d ",

            optimalSearchTree(keys, freq, n));

    return 0;

}

Output:

|  |  |  |
| --- | --- | --- |
| Algorithm | Time Complexity | Space Complexity |
| Warshall’s Algorithm | O(n3) | O(n2) |
| Optimal Binary search Tree | O(n3) | O(n2) |