**Ayush Goyal**

**190905522 CSE D 62**

**DAA Lab 5 (Week 5) – Decrease and Conquer**

**1) Write a program to determine the Topological sort of a given graph using**

**(A) Depth-First technique**

**CODE:**

#include <stdio.h>

#include <stdlib.h>

void insertEdge(int \*\*mat, int first, int second)

{

    mat[first][second] = 1;

}

int check(int \*\*mat, int num, int node)

{

    int result = 1;

    for (int i = 0; i < num; i++)

    {

        if (mat[i][node] == 1)

        {

            result = 0;

        }

    }

    return result;

}

void deleteEdges(int \*\*mat, int num, int node)

{

    for (int i = 0; i < num; i++)

    {

        mat[node][i] = 0;

    }

}

void topologicalsort(int \*\*mat, int num)

{

    int \*removed = (int \*)calloc(num, sizeof(int));

    for (int i = 0; i < num; i++)

    {

        removed[i] = 0;

    }

    int popped[num];

    int popIndex = 0;

    for (int i = 0; i < num; i++)

    {

        if (removed[i] == 0 && check(mat, num, i))

        {

            removed[i] = 1;

            popped[popIndex++] = i;

            deleteEdges(mat, num, i);

            i = -1;

        }

    }

    for (int i = 0; i < num; i++)

    {

        if (removed[i] == 0)

        {

            printf("\nThe Graph is a Cyclic Graph.\n");

            return;

        }

    }

    printf("\nThe Graph is a Directed Acyclic Graph (DAG), Topoplogical Sort order is : ");

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

    {

        printf("%d ", popped[i]);

    }

    printf("\n");

}

int main()

{

    int num = 6;

    int \*\*mat = (int \*\*)calloc(num, sizeof(int \*));

    for (int i = 0; i < num; i++)

    {

        mat[i] = (int \*)calloc(num, sizeof(int));

        for (int j = 0; j < num; j++)

        {

            mat[i][j] = 0;

        }

    }

    int m,n;

    do{

        printf("Enter edges to be joined : ");

        scanf("%d %d",&m,&n);

        if(m!=-1 && n!=-1)

            insertEdge(mat, m, n);

    }while(m!=-1);

    topologicalsort(mat, num);

    return 0;

}

**OUTPUT:**

**Text

Description automatically generated**

**(B) Source removal technique**

**CODE:**

#include <stdio.h>

#include <stdlib.h>

void insertEdge(int \*\*mat, int first, int second)

{

    mat[first][second] = 1;

}

int check(int \*\*mat, int num, int node)

{

    int result = 1;

    for (int i = 0; i < num; i++)

    {

        if (mat[i][node] == 1)

        {

            result = 0;

        }

    }

    return result;

}

void deleteEdge(int \*\*mat, int num, int node)

{

    for (int i = 0; i < num; i++)

    {

        mat[node][i] = 0;

    }

}

void topologicalsort(int \*\*mat, int num)

{

    int \*removed = (int \*)calloc(num, sizeof(int));

    for (int i = 0; i < num; i++)

    {

        removed[i] = 0;

    }

    int popped[num];

    int popIndex = 0;

    for (int i = 0; i < num; i++)

    {

        if (removed[i] == 0 && check(mat, num, i))

        {

            removed[i] = 1;

            popped[popIndex++] = i;

            deleteEdge(mat, num, i);

            i = -1;

        }

    }

    for (int i = 0; i < num; i++)

    {

        if (removed[i] == 0)

        {

            printf("The Graph is a Cyclic Graph.\n");

            return;

        }

    }

    printf("The Graph is a Directed Acyclic Graph (DAG), Topoplogical Sort order is : ");

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

    {

        printf("%d ", popped[i]);

    }

    printf("\n");

}

int main()

{

    int num = 6;

    int \*\*mat = (int \*\*)calloc(num, sizeof(int \*));

    for (int i = 0; i < num; i++)

    {

        mat[i] = (int \*)calloc(num, sizeof(int));

        for (int j = 0; j < num; j++)

        {

            mat[i][j] = 0;

        }

    }

    int m,n;

    do{

        printf("Enter edges to be joined : ");

        scanf("%d %d",&m,&n);

        if(m!=-1 && n!=-1)

            insertEdge(mat, m, n);

    }while(m!=-1);

    topologicalsort(mat, num);

    return 0;

}

**OUTPUT:**

**Text

Description automatically generated**

**2). Write a program to find diameter of a binary tree. Diameter of a binary tree is**

**the longest path between any two nodes.**

**CODE:**

#include <stdio.h>

#include <stdlib.h>

typedef struct node

{

    int val;

    struct node \*left;

    struct node \*right;

} \* NODE;

void inorder(NODE n)

{

    if (n)

    {

        inorder(n->left);

        printf("%d ", n->val);

        inorder(n->right);

    }

}

NODE insertNode(){

    int val;

    int check;

    printf("Enter element : ");

    scanf("%d", &val);

    NODE n = (NODE)malloc(sizeof(struct node));

    n->val = val;

    n->left = NULL;

    n->right = NULL;

    printf("Insert Left of %d : (Yes : 1, No : 0) : ", val);

    scanf("%d", &check);

    if (check)

        n->left = insertNode();

    printf("Insert Right of %d : (Yes : 1, No : 0) : ", val);

    scanf("%d", &check);

    if (check)

        n->right = insertNode();

    return n;

}

int max(int a, int b)

{

    return a > b ? a : b;

}

int height(NODE head)

{

    if (head == NULL)

    {

        return 0;

    }

    return max(height(head->left), height(head->right)) + 1;

}

void diameter(NODE cur, int \*max)

{

    if (cur)

    {

        int currentDiameter = height(cur->left) + height(cur->right) + 1;

        if (currentDiameter > \*max)

        {

            \*max = currentDiameter;

        }

        diameter(cur->left, max);

        diameter(cur->right, max);

    }

}

int main()

{

    NODE head = insertNode();

    printf("The InOrder Traversal is : ");

    inorder(head);

    int di = -1;

    diameter(head, &di);

    printf("\nThe Diameter of the Binary Tree is : %d\n", di);

    return 0;

}

**OUTPUT:**

**A picture containing text

Description automatically generated**

**THE END**