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:

```c
#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 o
rder 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:

```
D:\CSE\CSE Labs\DAA Lab\Week 5>gcc topologicalsortdfs.c -o topodfs

D:\CSE\CSE Labs\DAA Lab\Week 5>topodfs
Enter edges to be joined : 0 2
Enter edges to be joined : 1 2
Enter edges to be joined : 2 3
Enter edges to be joined : 3 4
Enter edges to be joined : 2 4
Enter edges to be joined : -1 -1

The Graph is a Directed Acyclic Graph (DAG), Topoplogical Sort order is : 0 1 2 3 4 5

D:\CSE\CSE Labs\DAA Lab\Week 5>topodfs
Enter edges to be joined : 0 2
Enter edges to be joined : 2 1
Enter edges to be joined : 1 0
Enter edges to be joined : 2 3
Enter edges to be joined : 2 4
Enter edges to be joined : 3 4
Enter edges to be joined : -1 -1

The Graph is a Cyclic Graph.

D:\CSE\CSE Labs\DAA Lab\Week 5>
```

(B) Source removal technique

CODE:

```c
#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 ord
er 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:

```
D:\CSE\CSE Labs\DAA Lab\Week 5>gcc topologicalsortsrm.c -o toposrm

D:\CSE\CSE Labs\DAA Lab\Week 5>toposrm
Enter edges to be joined : 0 2
Enter edges to be joined : 1 2
Enter edges to be joined : 2 3
Enter edges to be joined : 3 4
Enter edges to be joined : 2 4
Enter edges to be joined : -1 -1
The Graph is a Directed Acyclic Graph (DAG), Topoplogical Sort order is : 0 1 2 3 4 5

D:\CSE\CSE Labs\DAA Lab\Week 5>toposrm
Enter edges to be joined : 0 2
Enter edges to be joined : 2 1
Enter edges to be joined : 1 0
Enter edges to be joined : 2 3
Enter edges to be joined : 2 4
Enter edges to be joined : 3 4
Enter edges to be joined : -1 -1
The Graph is a Cyclic Graph.

D:\CSE\CSE Labs\DAA Lab\Week 5>
```

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:

```
D:\CSE\CSE Labs\DAA Lab\Week 5>gcc diameter.c -o diameter

D:\CSE\CSE Labs\DAA Lab\Week 5>diameter
Enter element : 1
Insert Left of 1 : (Yes : 1, No : 0) : 1
Enter element : 2
Insert Left of 2 : (Yes : 1, No : 0) : 1
Enter element : 3
Insert Left of 3 : (Yes : 1, No : 0) : 0
Insert Right of 3 : (Yes : 1, No : 0) : 0
Insert Right of 2 : (Yes : 1, No : 0) : 1
Enter element : 4
Insert Left of 4 : (Yes : 1, No : 0) : 0
Insert Right of 4 : (Yes : 1, No : 0) : 1
Enter element : 5
Insert Left of 5 : (Yes : 1, No : 0) :
0
Insert Right of 5 : (Yes : 1, No : 0) : 0
Insert Right of 1 : (Yes : 1, No : 0) : 1
Enter element : 6
Insert Left of 6 : (Yes : 1, No : 0) : 1
Enter element : 7
Insert Left of 7 : (Yes : 1, No : 0) : 0
Insert Right of 7 : (Yes : 1, No : 0) : 0
Insert Right of 6 : (Yes : 1, No : 0) : 0
The InOrder Traversal is : 3 2 4 5 1 7 6
The Diameter of the Binary Tree is : 6

D:\CSE\CSE Labs\DAA Lab\Week 5>
```

THE END