**PROGRAM 4 - TOPOLOGICAL SORT**

**Vertex Deletion Method**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int n;

int a[10][10];

int i,j,k,node;

int in[10]={0};

int v[10]={0};

printf("Enter n: ");

scanf("%d",&n);// read the number of nodes

printf("Enter Adj matrix: \n");

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

scanf("%d", &a[i][j]);

if(a[i][j] == 1)//check if vertex has indegree

in[j]++;

}

}

printf("\nTopological order: ");

for(k=1;k<=n;k++)

{

for(i=1;i<=n;i++)

{

if(in[i] == 0 && v[i] == 0)

{

node = i;

printf("%5d",node);/\*print the node in the topological order\*/

v[node] = 1;

break;

}

}

for(i=1;i<=n;i++)

if(a[node][i] == 1)

in[i]--;

}

printf("\n\n");

}

**Output**

Enter n:7

Enter Adjacency matrix:

0 1 1 0 0 0 0

0 0 0 0 1 0 1

0 0 0 0 0 0 0

1 1 1 0 0 1 1

0 0 0 0 0 0 0

0 0 0 0 0 0 0

0 0 0 0 1 1 0

Topological order:4 1 2 3 7 5 6

**DFS METHOD**

#include<stdio.h>

#include<stdlib.h>

int j=0,pop[10],v[10];

void dfs(int source,int n,int a[10][10])

{

int i,k,top=-1,stack[10];

v[source]=1;

stack[++top]= source+1;

while(top!=-1)//check if stack is not empty

{

for(k=0;k<n;k++)

{

if( a[source][k] == 1 && v[k] == 1 )

{

for(i=top;i>=0;i--)

if(stack[i] == k+1 )

{

printf("\n Topological order not possible");

exit(0);

}

}

else

{

if( a[source][k] == 1 && v[k] == 0)

{

v[k]=1;

stack[++top]= k+1;

source = k;

k=0;

}

}

}

pop[j++]=source+1;

top--;

source=stack[top]-1;

}

}

void topo(int n , int a[10][10])

{

int i,k;

for(i=0;i<n;i++)

v[i]=0;

for(k=0;k<n;k++)

if(v[k]== 0)

dfs(k,n,a);//dfs function call

}

int main()

{

int n,i,j,a[10][10];

printf("\n Enter the no of Vertices : ");

scanf("%d",&n);

printf("\n Enter the Adjacency matrix\n");

for(i=0;i<n;i++)

for(j=0;j<n;j++)

scanf("%d",&a[i][j]);

topo(n,a);

printf("\n The topological ordering is\n");

for(i=n-1;i>=0;i--)

printf("%d\t",pop[i]);

}