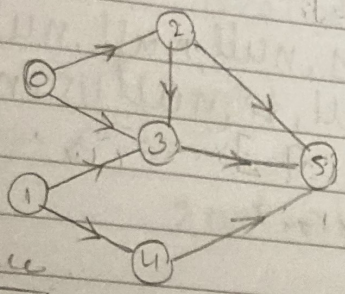
23/05/24

LAB-04:-

// Topological sorting using source removal method and DFS



Adjacency matrix:-

	0	1	2	3	4	5
0	0	0	1	1	0	0
1	0	0	0	1	1	0
2	0	0	0	1	0	1
3	0	0	0	0	0	1
4	0	0	0	0	0	1
5	0	0	0	0	0	0

using source removal:- → 0, 2, 1, 3, 4, 5 (or)
→ 1, 4, 0, 2, 3, 5

using DFS:-

0 (1), (4)
2 (2), (3)
3 (3), (2)
5 (4), (1)
1 (5), (6)
4 (6), (5)

popping sequence
5 3 2 0 4 1
Topological sort sequence
1 4 0 2 3 5

```
#include<stdio.h>
void topologicalSortSourceRemoval(int a[][100], int n);
void DFS(int u, int n, int a[][100], int *s, int *j, int *res,
    int *poppingSeq);
void topologicalSortDFS(int n, int a[100], int *poppingSeq);
void topologicalOrder(int n, int a[][100]);

int main(){
    int n;
    printf("Enter the number of vertices in the graph:");
    scanf("%d", &n);

    int a[100][100];
```

```
printf(" Enter the adjacency matrix: \n");
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        scanf("%d", &a[i][j]);
    }
}
printf("Topological sort using source removal method: \n");
topologicalSortSourceRemoval(a, n);

printf("\nTopological Sort using DFS: \n");
topologicalSortDFS(n, a, NULL);

printf("\nTopological Order: \n");
topologicalOrder(n, a);

return 0;
}

void topologicalSortSourceRemoval(int a[][100], int n){
    int indegree[100]={0};
    int s[100]={0};
    int top = -1;

    for (int j=0; j<n; j++){
        int sum = 0;
        for(int i=0; i<n; i++){
            sum += a[i][j];
        }
        indegree[j] = sum;
    }
```

- pro

```
for(int i=0; i<n; i++){
    if(indegree[i]==0){
        top++;
        s[top]=i;
    }
}
while(top!=-1){
    int u=s[top];
    top--;
    printf("%d ",u);
    for(int v=0; v<n; v++){
        if(a[u][v]==1){
            indegree[v]--;
            if(indegree[v]==0){
                top++;
                s[top]=v;
            }
        }
    }
}
}

void DFS(int u, int n, int a[][100], int *s, int *j,
    int *res, int *poppingSeq){
    s[u]=1;
    for(int v=0; v<n; v++){
        if(a[u][v]==1 && s[v]==0){
            DFS(v,n,a,s,j,res,poppingSeq);
        }
    }
    (*j)++;
    res[*j]=u;
```

```
    if (poppingSeq != NULL){
        (*poppingSeq)++;
        printf("popped: %d\n",u);
    }
}
void topologicalSortDFS(int n, int a[][100], int *poppingSeq){
    int s[100]={0};
    int j=-1;
    int res[100]={0};

    for(int u=0;u<n;u++){
        if(s[u]==0){
            DFS(u,n,a,s,&j,res,poppingSeq);
        }
    }
    for(int i=n-1;i>=0;i--){
        printf("%d",res[i]);
    }
}
void topologicalOrder(int n, int a[][100]){
    int s[100]={0};
    int j=0;
    int res[100]={0};
    int poppingSeq=0;

    for(int u=0;u<n;u++){
        if(s[u]==0){
            DFS(u,n,a,s,&j,res,&poppingSeq);
        }
    }
    for(int i=n-1;i>0;i--){
        printf("%d ",res[i]);
```

output:-

```
Enter the number of vertices in the graph: 6
Enter the adjacency matrix:
0 0 1 1 0 0
0 0 0 1 1 0
0 0 0 1 0 1
0 0 0 0 0 1
0 0 0 0 0 1
0 0 0 0 0 0

Topological Sort using Source Removal Method:
1 4 0 2 3 5

Topological Sort using DFS:
1 4 0 2 3 5

Topological order:
Popped : 5
Popped : 3
Popped : 2
Popped : 0
Popped : 4
Popped : 1

4 0 2 3 5 0.
```