Date 23/5/24
Page

LAB-4.

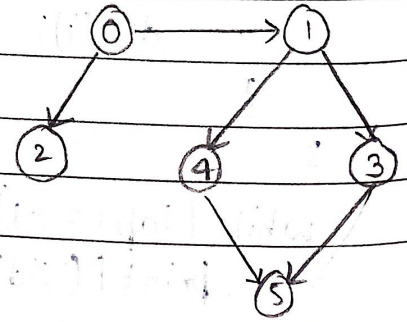
① Topological sorting using DFS

```
#define MAX 100
int adj[MAX][MAX];
int visited[MAX];
int stack[MAX], top = -1, n;
void push(int v){
    stack[++top] = v;
}
int pop(){
    return stack[top--];
}
void dfs(int v){
    visited[v]=1;
    for(int i=0; i<n;i++){
        if(adj[v][i] && !visited[i]){
            dfs(i);
        }
    }
    push(v);
}
void topologicalSort(){
    for(int i=0;i<n;i++){
        visited[i]=0;
    }
    for(int i=0; i<n; i++){
        if(!visited[i]){
            dfs(i);
        }
    }
    while(top!=-1){
        printf("%d",pop());
    }
```

```
    printf("\n");
}
int main(){
    int edges, start, end;
    printf("Enter the no. of vertices:");
    scanf("%d", &n);
    printf("Enter the no. of edges:");
    scanf("%d", &edges);
    for(int i=0; i<edges; i++){
        printf("Enter start and end vertex of edge
                %d:", i+1);
        scanf("%d %d", &start, &end);
        adj[start][end] = 1;
    }
    printf("Topological sort:");
    topologicalSort();
    return 0;
}
```

O/P ⇒

```
Enter no. of vertices : 5
Enter no. of edges: 6
Enter start & end vertex of edge 1 : 0 1
Enter start & end vertex of edge 2 : 0 2
Enter start & end vertex of edge 3 : 1 3
Enter start & end vertex of edge 4 : 1 4
Enter start & end vertex of edge 5 : 3 5
Enter start & end vertex of edge 6 : 4 5
Topological sort: 0 2 1 4 3 5
```



Date
Page

② Topological Order using Source Removal.

```
int st[100], top = -1;
void degree(int adj[][20], int n){
    int indegree[20];
    int sum=0;
    for(int j=0;j<n;j++){
        sum=0;
        for(int i=0; i<n;i++){
            sum =sum + adj[i][j];
        }
        indegree[j] =sum;
    }
    for(int i=0 ; i<n; i++){
        if (indegree[i]==0){
            top++;
            st[top]=i;
        }
    }
    while (top != -1){
        int u =st[top];
        top--;
        printf("%d", u);
        for(int v=0; v<n; v++){
            if(adj[u][v]==1){
                indegree[v]--;
                if (indegree[v]==0){
                    top++;
                    st[top] =v;
                }
            }
        }
    }
}
```

O/P:-

Ent no. of nodes: 4

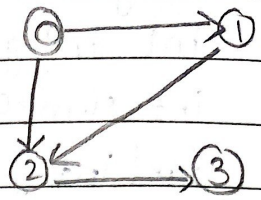
Enter the adjacency matrix

0 1 1 0

0 0 1 0

0 0 0 1

0 0 0 0



Topological order of nodes: 0 1 2 3

23/5/24