# Advanced Datastructure LAB EXAM

**Rasika V V**

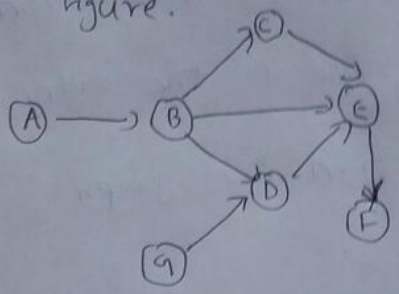
**Regno:TKM20MCA-2029**

**Roll no:MCA229**

**MCA s1**

Question

1) Consider a directed acyclic graph G given the following figure.




Develop a program to implement topological sorting.

2) write a program for creating doubly linked list and perform following operation.

A) Insert an element at a particular position.

B) Search an element.

C) Delete an element at the end of the list.

## Topological Sorting

Algorithm

1. Identify a node with no incoming edges.
2. Add that node to the ordering.
3. Remove it from the graph.
4. Repeat
5. Here the given graph adjacency matrix is.

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| B | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| C | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| D | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| E | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

**Program code**

```
#include <stdio.h>

int main(){
        int v,i,j,k,ar[10][10],indegree[10],flag[10];
   int count=0;

        printf("ENTER THE NUMBER OF VERTICES:\n");
        scanf("%d",&v);

        printf("ENTER THE ADJACENCY MATRIX:\n");
        for(i=0;i<v;i++){
                for(j=0;j<v;j++)
                        scanf("%d",&ar[i][j]);
        }

        for(i=0;i<v;i++){
      indegree[i]=0;
      flag[i]=0;
   }

   for(i=0;i<v;i++)
      for(j=0;j<v;j++)
         indegree[i]=indegree[i]+ar[j][i];

   printf("\nTHE TOPOLOGICAL SORT ORDER IS:");

   while(count<v){
      for(k=0;k<v;k++){
```

```
        if((indegree[k]==0) && (flag[k]==0)){
            printf("%d ",(k+1));
            flag [k]=1;
        }

        for(i=0;i<v;i++){
            if(ar[i][k]==1)
                indegree[k]--;
        }
      }

      count++;
   }
   return 0;
}
```

**Output**

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
Microsoft Windows [Version 10.0.19041.329]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\USER\Desktop\Datastructure_LAB\LAB_CYCLE-4>gcc -o Topological_sort Topological_sort.c

C:\Users\USER\Desktop\Datastructure_LAB\LAB_CYCLE-4>Topological_sort
ENTER THE NUMBER OF VERTICES:
7
ENTER THE ADJACENCY MATRIX:
0 1 0 0 0 0 0
0 0 1 1 1 0 0
0 0 0 0 1 0 0
0 0 0 0 1 0 0
0 0 0 0 0 1 0
0 0 0 0 0 0 0
0 0 0 1 0 0 0

THE TOPOLOGICAL SORT ORDER IS:1 7 2 3 4 5 6
C:\Users\USER\Desktop\Datastructure_LAB\LAB_CYCLE-4>
```

**Question 2**

1 Doubly linked list

Algorithm

2- Represent a node in doubly linked list

```
struct dnode
{
    int data;
    struct node *prev;
    struct node *next;
};
```

2. *prev → address of previous node.
   *next → address of next node.

- t = new node
- Enter "the node to be inserted"
- Read n
- t->info = n
- t->next = NULL
- t->prev = NULL.

Insertion

Begin

1. If start = NULL

   start = t

   else t->next = NULL

   t->next->prev = t

Start = t

Return

middle

1. Print the position of a node you want to insert.
2. Read x.
3. P = start
4. Repeat while P <> NULL

if (p->info = n)

t->next = p->next

p->next = t

t->prev = p

p->next->prev = t

Return

else

P = P->next

5. Print x not found

t->next = NULL

p->next = t.

Deletion

Last

2. P = start
2. Repeat while P <> NULL

if (p->next = NULL)

Delnode (P)

3. Return.

Display

1. p = start
2. Repeat while p <> NULL
   print p->info

Search

search the element in that inserted list using flag.

**Program code**

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   struct node *prev;
   struct node *next;
   int data;
};
struct node *head;
void insertion_beginning();
void insertion_specified();
void deletion_last();
void display();
void search();
void main ()
{
int choice =0;
   while(choice != 6)
   {
      printf("\nChoose one option from the following list ...\n");
      printf("\n---------------------------------------------------\n");
      printf("\n1.Insert in begining\n2.Insert at any random location\n3.Delete
from last\n4.Search\n5.Show\n6.Exit\n");
      printf("\nEnter your choice?\n");
      scanf("\n%d",&choice);
      switch(choice)
```

```
{
   case 1:
   insertion_beginning();
   break;

   case 2:
   insertion_specified();
   break;

   case 3:
   deletion_last();
   break;

   case 4:
   search();
   break;

   case 5:
   display();
   break;

   case 6:
   exit(0);
   break;

   default:
```

```
            printf("Please Enter Valid choice..");
        }
    }
}
void insertion_beginning()
{
   struct node *ptr;
   int item;
   ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
      printf("\nOVERFLOW");
   }
   else
   {
    printf("\nEnter Item value");
    scanf("%d",&item);

   if(head==NULL)
   {
      ptr->next = NULL;
      ptr->prev=NULL;
      ptr->data=item;
      head=ptr;
   }
   else
```

```
  {
     ptr->data=item;
     ptr->prev=NULL;
     ptr->next = head;
     head->prev=ptr;
     head=ptr;
  }
  printf("\nNode inserted\n");
}

}

void insertion_specified()
{
  struct node *ptr,*temp;
       int item,loc,i;
  ptr = (struct node *)malloc(sizeof(struct node));
  if(ptr == NULL)
  {
     printf("\n OVERFLOW");
  }
  else
  {
     temp=head;
     printf("Enter the location");
     scanf("%d",&loc);
```

```
      for(i=0;i<loc;i++)
      {
         temp = temp->next;
         if(temp == NULL)
         {
            printf("\n There are less than %d elements", loc);
            return;
         }
      }
      printf("Enter value");
      scanf("%d",&item);
      ptr->data = item;
      ptr->next = temp->next;
      ptr -> prev = temp;
      temp->next = ptr;
      temp->next->prev=ptr;
      printf("\nnode inserted\n");
   }
}

   void deletion_last()
   {
   struct node *ptr;
   if(head == NULL)
   {
      printf("\n UNDERFLOW");
```

```
    }
    else if(head->next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted\n");
    }
    else
    {
        ptr = head;
        if(ptr->next != NULL)
        {
            ptr = ptr -> next;
        }
        ptr -> prev -> next = NULL;
        free(ptr);
        printf("\nnode deleted\n");
    }
}

void display()
{
    struct node *ptr;
    printf("\n printing values...\n");
    ptr = head;
    while(ptr != NULL)
```

```
    {
        printf("%d\n",ptr->data);
        ptr=ptr->next;
    }
}
void search()
{
    struct node *ptr;
    int item,i=0,flag;
    ptr = head;
    if(ptr == NULL)
    {
        printf("\nEmpty List\n");
    }
    else
    {
        printf("\nEnter item which you want to search?\n");
        scanf("%d",&item);
        while (ptr!=NULL)
        {
            if(ptr->data == item)
            {
                printf("\nitem found at location %d ",i+1);
                flag=0;
                break;
            }
```

```
            else
            {
               flag=1;
            }
            i++;
            ptr = ptr -> next;
        }
        if(flag==1)
        {
            printf("\nItem not found\n");
        }
    }

}
```

Output

```
C:\Users\USER\Desktop\Datastructure_LAB\LAB_CYCLE-4>Doubly_linkedlist

Choose one option from the following list ...

----------------------------------------------------

1.Insert in begining
2.Insert at any random location
3.Delete from last
4.Search
5.Show
6.Exit

Enter your choice?
1

Enter Item value20

Node inserted

Choose one option from the following list ...

----------------------------------------------------

1.Insert in begining
2.Insert at any random location
3.Delete from last
4.Search
5.Show
6.Exit

Enter your choice?
1

Enter Item value10

Node inserted

Choose one option from the following list ...

----------------------------------------------------

1.Insert in begining
2.Insert at any random location
3.Delete from last
4.Search
5.Show
6.Exit

Enter your choice?
5

 printing values...
10
20
```

```
Choose one option from the following list ...

----------------------------------------------------

1.Insert in begining
2.Insert at any random location
3.Delete from last
4.Search
5.Show
6.Exit

Enter your choice?
3

node deleted

Choose one option from the following list ...

----------------------------------------------------

1.Insert in begining
2.Insert at any random location
3.Delete from last
4.Search
5.Show
6.Exit

Enter your choice?
5

 printing values...
10
```

```
Choose one option from the following list ...

----------------------------------------------------------

1.Insert in begining
2.Insert at any random location
3.Delete from last
4.Search
5.Show
6.Exit

Enter your choice?
2
Enter the location0
Enter value40

node inserted

Choose one option from the following list ...

----------------------------------------------------------

1.Insert in begining
2.Insert at any random location
3.Delete from last
4.Search
5.Show
6.Exit

Enter your choice?
5

 printing values...
10
40

Choose one option from the following list ...

----------------------------------------------------------

1.Insert in begining
2.Insert at any random location
3.Delete from last
4.Search
5.Show
6.Exit

Enter your choice?
4

Enter item which you want to search?
10

item found at location 1
Choose one option from the following list ...

----------------------------------------------------------

1.Insert in begining
2.Insert at any random location
```

Git repo link: [https://github.com/rasikavv/Datastructure_LAB_EXAM](https://github.com/rasikavv/Datastructure_LAB_EXAM)