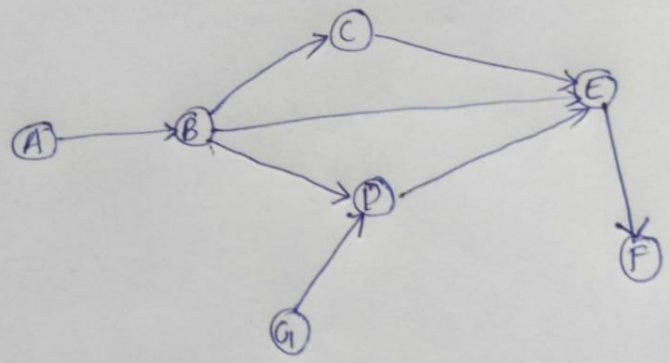
SIJI JOSE
20 MCA 237

REPO LINK:

https://github.com/siji1999/Data-Structures.git

QUESTION-1

1. consider a directed acyclic graph G given in following figure.



Develop a program to implement topological sorting.

ALGORITHM

Algorithm

Topological sort (G)

1 For each vertex $u \in V$

2. do indegree $[u] \leftarrow 0$

3. for each vertex $u \in V$

4. do for each $v \in adj[u]$

5. do indegree $[v] \leftarrow$ indegree $[v] + 1$

6. $Q \leftarrow \phi$

7. For each vertex $u \in V$

8. do if indegree $[u] = 0$

9. then enqueue (Q, u)

10. while $Q = \phi$

11. do $u \leftarrow$ Dequeue (Q)

12. Output u

13. ~~do u ← Deque~~ for each $v \in adj[u]$

14. do in-degree $[v] \leftarrow$ indegree $[v] - 1$

15. if in degree $[v] = 0$

16. the enqueue $[Q, v]$

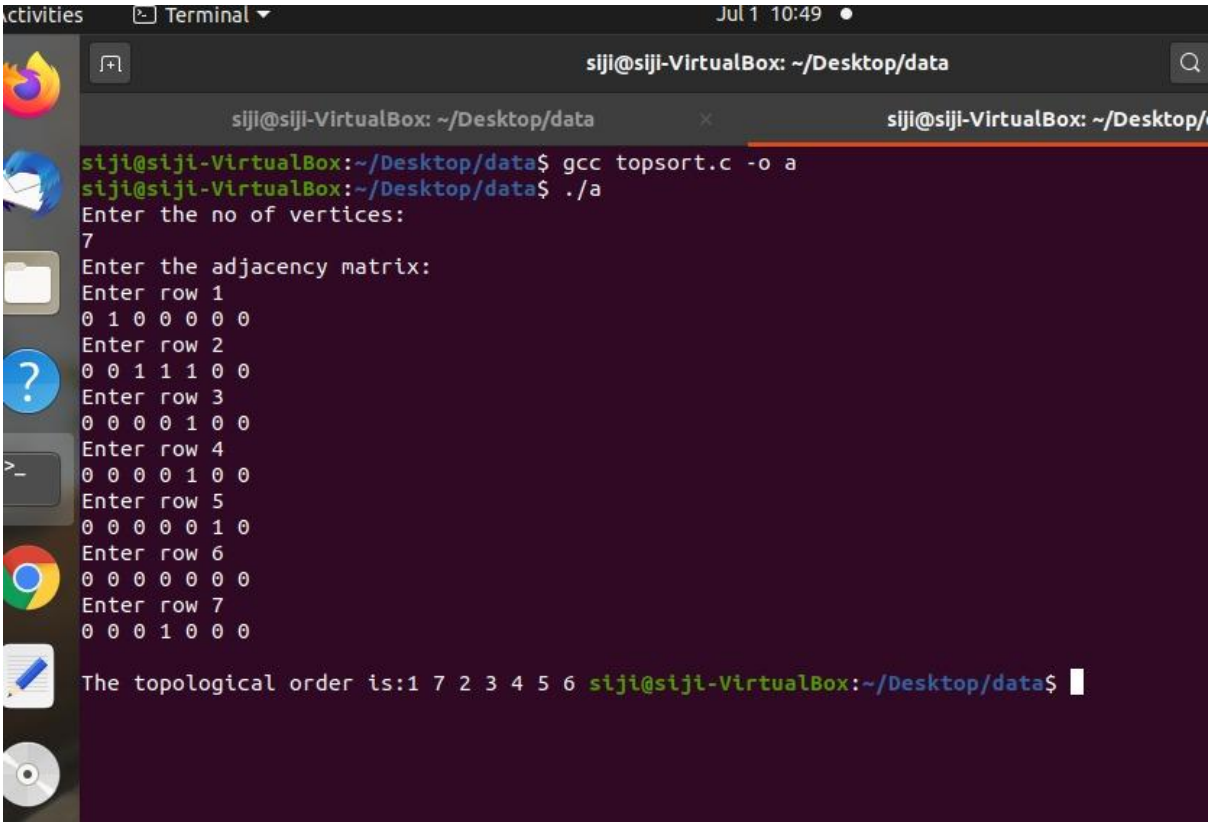
17. do if indegree $[u] \neq 0$

18. Report there is a cycle.

PROGRAM CODE

```
#include <stdio.h>
int main(){
int i,j,k,n,a[10][10],indeg[10],flag[10],count=0;

printf("Enter the no of vertices:\n");
scanf("%d",&n);

printf("Enter the adjacency matrix:\n");
for(i=0;i<n;i++){
printf("Enter row %d\n",i+1);
for(j=0;j<n;j++)
scanf("%d",&a[i][j]);
}

for(i=0;i<n;i++){
        indeg[i]=0;
        flag[i]=0;
    }

    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            indeg[i]=indeg[i]+a[j][i];

    printf("\nThe topological order is:");

    while(count<n){
        for(k=0;k<n;k++){
            if((indeg[k]==0) && (flag[k]==0)){
                printf("%d ",(k+1));
                flag [k]=1;
            }

            for(i=0;i<n;i++){
                if(a[i][k]==1)
                    indeg[k]--;
            }
        }

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

OUTPUT


Activities
Terminal
Jul 1 10:49
siji@siji-VirtualBox: ~/Desktop/data
siji@siji-VirtualBox: ~/Desktop/data
siji@siji-VirtualBox: ~/Desktop/
siji@siji-VirtualBox:~/Desktop/data$ gcc topsort.c -o a
siji@siji-VirtualBox:~/Desktop/data$ ./a
Enter the no of vertices:
7
Enter the adjacency matrix:
Enter row 1
0 1 0 0 0 0 0
Enter row 2
0 0 1 1 1 0 0
Enter row 3
0 0 0 0 1 0 0
Enter row 4
0 0 0 0 1 0 0
Enter row 5
0 0 0 0 0 1 0
Enter row 6
0 0 0 0 0 0 0
Enter row 7
0 0 0 1 0 0 0
The topological order is:1 7 2 3 4 5 6 siji@siji-VirtualBox:~/Desktop/data$

QUESTION-2

2. Write a program for creating doubly linked list and perform the following operation.

A) Insert an element at a particular position

B) Search an element

C) Delete an element at the end of the list

ALGORITHM

Algorithm

1. Define a node class which represents a node in the list. It will have three properties, data, previous which will point to the previous node and next which will point to the next node.

2. Define another class for creating a doubly linked list, and it has two nodes: head and tail. Initially, head and tail will point to null.

3. add node() will add node to the list:

a) It it will insert the node as the head by checking whether the head is null.

b) Both head and tail will point to a newly added node

c) Head's previous pointer will point to null and tail's next pointer will point to null.

d) If the head is not ~~full~~ null, the new node will be inserted at the end of the list such that new node's previous pointer will point to tail.

e) The new node will become the new tail. Tail's next

4. display () will show all the nodes present in the list.

a) Define a new node 'current' that will point to the head

b) print current data till current points to null

c) current will point to the next node in the list in each iteration.

PROGRAM CODE

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
   struct node *prev;
   int n;
   struct node *next;
}*h,*temp,*temp1,*temp2,*temp4;

void insert1();
void insert2();
void insert3();
void traverse beeg();
void traverse end(int);
void sort();
void search();
void update();
void delete();

int count = 0;

void main()
{
   int ch;

   h = NULL;
   temp = temp1 = NULL;


   printf("\n 1 - Insert at position i");
   printf("\n 2 - Delete at i");
   printf("\n 3 - Search for element");
   printf("\n 4 - Exit");

   while (1)
   {
      printf("\n Enter choice : ");
      scanf("%d", &ch);
      switch (ch)
```

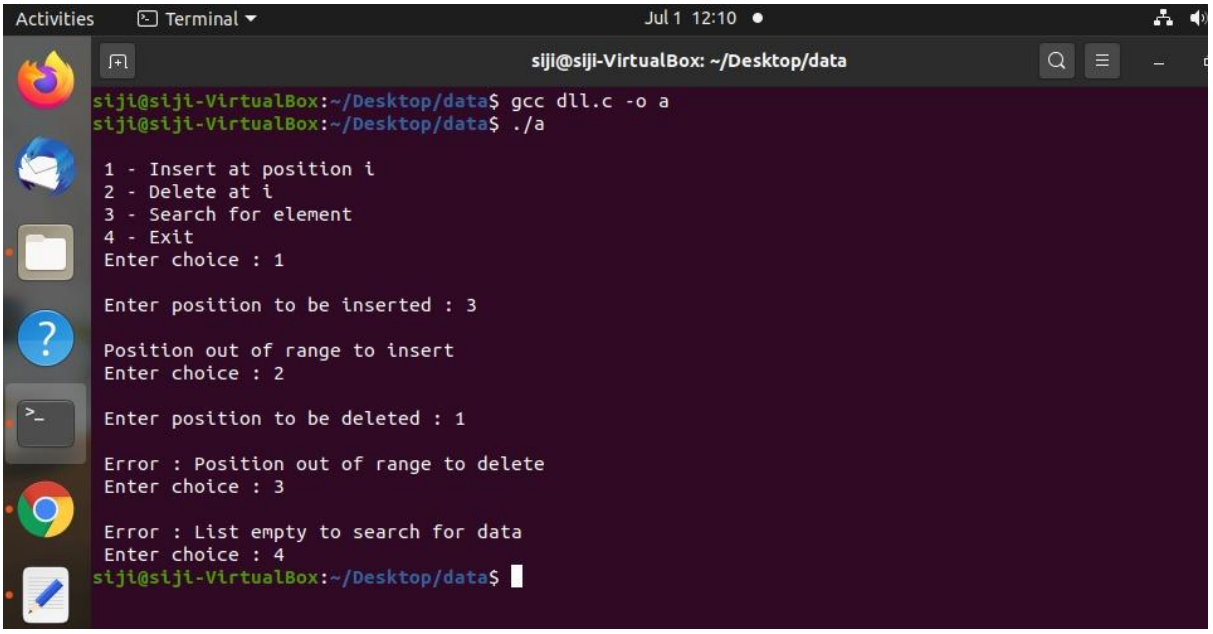
```
        {

        case 1:
            insert3();
            break;
        case 2:
            delete();
            break;
        case 3:
            search();
            break;
        case 4:
            exit(0);
        default:
            printf("\n Wrong choice menu");
        }
    }
}

/* TO create an empty node */
void create()
{
    int data;

    temp =(struct node *)malloc(1*sizeof(struct node));
    temp->prev = NULL;
    temp->next = NULL;
    printf("\n Enter value to node : ");
    scanf("%d", &data);
    temp->n = data;
    count++;
}

/* To insert at any position */
void insert3()
{
    int pos, i = 2;

    printf("\n Enter position to be inserted : ");
    scanf("%d", &pos);
    temp2 = h;

    if ((pos < 1) || (pos >= count + 1))
    {
```

```
        printf("\n Position out of range to insert");
        return;
    }
    if ((h == NULL) && (pos != 1))
    {
        printf("\n Empty list cannot insert other than 1st position");
        return;
    }
    if ((h == NULL) && (pos == 1))
    {
        create();
        h = temp;
        temp1 = h;
        return;
    }
    else
    {
        while (i < pos)
        {
            temp2 = temp2->next;
            i++;
        }
        create();
        temp->prev = temp2;
        temp->next = temp2->next;
        temp2->next->prev = temp;
        temp2->next = temp;
    }
}

/* To delete an element */
void delete()
{
    int i = 1, pos;

    printf("\n Enter position to be deleted : ");
    scanf("%d", &pos);
    temp2 = h;

    if ((pos < 1) || (pos >= count + 1))
    {
        printf("\n Error : Position out of range to delete");
        return;
    }
```

```
    if (h == NULL)
    {
        printf("\n Error : Empty list no elements to delete");
        return;
    }
    else
    {
        while (i < pos)
        {
            temp2 = temp2->next;
            i++;
        }
        if (i == 1)
        {
            if (temp2->next == NULL)
            {
                printf("Node deleted from list");
                free(temp2);
                temp2 = h = NULL;
                return;
            }
        }
        if (temp2->next == NULL)
        {
            temp2->prev->next = NULL;
            free(temp2);
            printf("Node deleted from list");
            return;
        }
        temp2->next->prev = temp2->prev;
        if (i != 1)
            temp2->prev->next = temp2->next;    /* Might not need this statement if i == 1
check */
        if (i == 1)
            h = temp2->next;
        printf("\n Node deleted");
        free(temp2);
    }
    count--;
}


/* To search for an element in the list */
void search()
```

```
{
    int data, count = 0;
    temp2 = h;

    if (temp2 == NULL)
    {
        printf("\n Error : List empty to search for data");
        return;
    }
    printf("\n Enter value to search : ");
    scanf("%d", &data);
    while (temp2 != NULL)
    {
        if (temp2->n == data)
        {
            printf("\n Data found in %d position",count + 1);
            return;
        }
        else
             temp2 = temp2->next;
            count++;
    }
    printf("\n Error : %d not found in list", data);
}
```

OUTPUT


Activities
Terminal
Jul 1 12:10
siji@siji-VirtualBox: ~/Desktop/data
siji@siji-VirtualBox:~/Desktop/data$ gcc dll.c -o a
siji@siji-VirtualBox:~/Desktop/data$ ./a
1 - Insert at position i
2 - Delete at i
3 - Search for element
4 - Exit
Enter choice : 1
Enter position to be inserted : 3
Position out of range to insert
Enter choice : 2
Enter position to be deleted : 1
Error : Position out of range to delete
Enter choice : 3
Error : List empty to search for data
Enter choice : 4
siji@siji-VirtualBox:~/Desktop/data$