Dr. D.Y.Patil Arts, Commerce & Science College,

Pimpri, Pune-18

S.Y.B.Sc(Comp. Sci) sem-IV 2022-23

Data Structures and Algorithms – II

Assignment 4: Graph as Adjacency List

Date: 7/4/23

Set A

a) Write a C program that accepts the vertices and edges of a graph. Create adjacency list and display the adjacency list.

```
#include<stdio.h>
#include<stdlib.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list[10];
int nov;
node *getnodenum(int vno)
{
   node *temp;
   temp=(node*) malloc(sizeof(node));
   temp->data=vno;
   temp->next=NULL;
   return temp;
}
void display(node *list[10])
{
  int i;
  node *ptr;
  for(i=1;i<=nov;i++)
  {
        printf("V%d  ",i);
        for(ptr=list[i];ptr!=NULL;ptr=ptr->next)
             printf("%d->",ptr->data);
        printf("NULL");
        printf("\n");
  }

}
void creatadjacencylist()
{
      int i,j;
      char ch;
      node *temp,*last;
      for(i=1;i<=nov;i++)
            list[i]=NULL;
      for(i=1;i<=nov;i++)
      {
            for(j=1;j<=nov;j++)
            {
```

```
                printf("\nIs there edge between V[%d] and V[%d] (Choose
(y/n): ",i,j);
                scanf(" %c",&ch);
                if(ch=='Y' || ch=='y')
                {
                      temp=getnodenum(j);
                      if(list[i]==NULL)
                            list[i]=temp;
                      else
                      {
                            for(last=list[i];last->next!=NULL;last=last-
>next);
                            last->next=temp;
                      }
                }
           }
      }

      //display(list);
}

main()
{
  int i,j,a[20][20];

  printf("\nEnter no. of vertices: ");
  scanf("%d",&nov);

  creatadjacencylist();
  printf("\n******Adjacency List*******\n");
  display(list);
}
/*
[root@localhost ass4]# cc ass4setaq1.c
[root@localhost ass4]# ./a.out

Enter no. of vertices: 5

Is there edge between V[1] and V[1] (Choose (y/n): n

Is there edge between V[1] and V[2] (Choose (y/n): y

Is there edge between V[1] and V[3] (Choose (y/n): y

Is there edge between V[1] and V[4] (Choose (y/n): y

Is there edge between V[1] and V[5] (Choose (y/n): n

Is there edge between V[2] and V[1] (Choose (y/n): n

Is there edge between V[2] and V[2] (Choose (y/n): n

Is there edge between V[2] and V[3] (Choose (y/n): y

Is there edge between V[2] and V[4] (Choose (y/n): n
```

```
Is there edge between V[2] and V[5] (Choose (y/n): n

Is there edge between V[3] and V[1] (Choose (y/n): n

Is there edge between V[3] and V[2] (Choose (y/n): n

Is there edge between V[3] and V[3] (Choose (y/n): n

Is there edge between V[3] and V[4] (Choose (y/n): y

Is there edge between V[3] and V[5] (Choose (y/n): y

Is there edge between V[4] and V[1] (Choose (y/n): n

Is there edge between V[4] and V[2] (Choose (y/n): n

Is there edge between V[4] and V[3] (Choose (y/n): n

Is there edge between V[4] and V[4] (Choose (y/n): n

Is there edge between V[4] and V[5] (Choose (y/n): y

Is there edge between V[5] and V[1] (Choose (y/n): n

Is there edge between V[5] and V[2] (Choose (y/n): n

Is there edge between V[5] and V[3] (Choose (y/n): n

Is there edge between V[5] and V[4] (Choose (y/n): n

Is there edge between V[5] and V[5] (Choose (y/n): n

******Adjacency List*******
V1  2->3->4->NULL
V2  3->NULL
V3  4->5->NULL
V4  5->NULL
V5  NULL
*/
```

b) Write a C program that accepts the vertices and edges of a graph. Create adjacency list. Implement functions to print indegree, outdegree and total degree of all vertex of graph.

```
#include<stdio.h>
#include<stdlib.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list[10];
```

```
int nov;
node *getnodenum(int vno)
{
   node *temp;
   temp=(node*) malloc(sizeof(node));
   temp->data=vno;
   temp->next=NULL;
   return temp;
}
void display(node *list[10])
{
  int i;
  node *ptr;
  for(i=1;i<=nov;i++)
  {
        printf("V%d  ",i);
        for(ptr=list[i];ptr!=NULL;ptr=ptr->next)
             printf("%d->",ptr->data);
        printf("NULL");
        printf("\n");
  }

}
void creatadjacencylist()
{
      int i,j;
      char ch;
      node *temp,*last;
      for(i=1;i<=nov;i++)
           list[i]=NULL;
      for(i=1;i<=nov;i++)
      {
            for(j=1;j<=nov;j++)
            {
                  printf("\nIs there edge between V[%d] and V[%d] (choose :
y/n): ",i,j);
                  scanf(" %c",&ch);
                  if(ch=='Y' || ch=='y')
                  {
                        temp=getnodenum(j);
                        if(list[i]==NULL)
                              list[i]=temp;
                        else
                        {
                              for(last=list[i];last->next!=NULL;last=last-
>next);

                              last->next=temp;
                        }
                  }
            }
      }


}
void degree()
{
```

```
    int i,cnt,outcnt[10],incnt[10]={0};
    node *ptr;
    for(i=1;i<=nov;i++)
      {
        for(ptr=list[i],cnt=0;ptr!=NULL;ptr=ptr->next,cnt++)
          incnt[ptr->data]+=1;
        outcnt[i]=cnt;
      }
    printf("\nVertex\t   Indegree\t   Outdegree\t  Totaldegree");
    for(i=1;i<=nov;i++)
      printf("\nV%d \t\t %d \t\t %d \t\t
%d",i,incnt[i],outcnt[i],incnt[i]+outcnt[i]);
}
main()
{
  int i,j,a[20][20];

  printf("\nEnter no. of vertices: ");
  scanf("%d",&nov);

  creatadjacencylist();
  printf("\n******Adjacency List*******\n");
  display(list);
 degree();
}
/*
[root@localhost ass4]# cc ass4setaq2.c
[root@localhost ass4]# ./a.out

Enter no. of vertices: 5

Is there edge between V[1] and V[1] (choose : y/n): n

Is there edge between V[1] and V[2] (choose : y/n): y

Is there edge between V[1] and V[3] (choose : y/n): y

Is there edge between V[1] and V[4] (choose : y/n): y

Is there edge between V[1] and V[5] (choose : y/n): n

Is there edge between V[2] and V[1] (choose : y/n): n

Is there edge between V[2] and V[2] (choose : y/n): n

Is there edge between V[2] and V[3] (choose : y/n): y

Is there edge between V[2] and V[4] (choose : y/n): n

Is there edge between V[2] and V[5] (choose : y/n): n

Is there edge between V[3] and V[1] (choose : y/n): n

Is there edge between V[3] and V[2] (choose : y/n): n

Is there edge between V[3] and V[3] (choose : y/n): n
```

```
Is there edge between V[3] and V[4] (choose : y/n): y

Is there edge between V[3] and V[5] (choose : y/n): y

Is there edge between V[4] and V[1] (choose : y/n): n

Is there edge between V[4] and V[2] (choose : y/n): n

Is there edge between V[4] and V[3] (choose : y/n): n

Is there edge between V[4] and V[4] (choose : y/n): n

Is there edge between V[4] and V[5] (choose : y/n): y

Is there edge between V[5] and V[1] (choose : y/n): n

Is there edge between V[5] and V[2] (choose : y/n): n

Is there edge between V[5] and V[3] (choose : y/n): n

Is there edge between V[5] and V[4] (choose : y/n): n

Is there edge between V[5] and V[5] (choose : y/n): n

******Adjacency List*******
V1  2->3->4->NULL
V2  3->NULL
V3  4->5->NULL
V4  5->NULL
V5  NULL

Vertex       Indegree        Outdegree       Totaldegree
V1               0               3               3
V2               1               1               2
V3               2               2               4
V4               2               1               3
V5               2               0               2
*/
```

Set B

a) Write a C program that accepts the vertices and edges of a graph and store it as an adjacency list. Implement function to traverse the graph using Breadth First Search (BFS) traversal.

```
#include<stdio.h>
#include<stdlib.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
struct queue
{
```

```
        int front,rear;
        int Q[20];
};
typedef struct queue QUEUE;

int nov,a[20][20];
int visited[20];

void initqueue(QUEUE *q)
{
        int i;
        for(i=0;i<20;i++)
                q->Q[i]=0;
        q->rear=-1;
        q->front=-1;
        printf("\nQueue created");
}
void add(QUEUE *q,int data)
{

        q->Q[++q->rear]=data;
}
int delet(QUEUE *q)
{

        return(q->Q[++q->front]);
}
int isempty(QUEUE *q)
{
        if (q->rear==q->front)
                return 1;
        else
                return 0;
}
void bfs(node *list[10],int nov)

{
        int ver;
        node *ptr;
        QUEUE q;
        initqueue(&q);
        printf("\n \t BFS seq. :\n ");
        ver=1;
        add(&q,ver);
        visited[ver]=1;

        while(!isempty(&q))
        {
                 ver=delet(&q);
            for(ptr=list[ver];ptr!=NULL;ptr=ptr->next)

                {
                        if(visited[ptr->data]==0)
                        {
                                add(&q,ptr->data);
                                visited[ptr->data]=1;
```

```
                        }
                }
                printf("\t V%d ",ver);
        }
}

node *list[10];
int nov,visited[20];
node *getnodenum(int vno)
{
   node *temp;
   temp=(node*) malloc(sizeof(node));
   temp->data=vno;
   temp->next=NULL;
   return temp;
}
void display(node *list[10])
{
  int i;
  node *ptr;
  for(i=1;i<=nov;i++)
  {
      printf("V%d  ",i);
      for(ptr=list[i];ptr!=NULL;ptr=ptr->next)
           printf("%d->",ptr->data);
      printf("NULL");
      printf("\n");
  }

}
void creatadjacencylist()
{
    int i,j;
    char ch;
    node *temp,*last;
    for(i=1;i<=nov;i++)
         list[i]=NULL;
    for(i=1;i<=nov;i++)
    {
         for(j=1;j<=nov;j++)
         {
               printf("\nIs there edge between V[%d] and V[%d] (Choose
(y/n): ",i,j);
               scanf(" %c",&ch);
               if(ch=='Y' || ch=='y')
               {
                     temp=getnodenum(j);
                     if(list[i]==NULL)
                           list[i]=temp;
                     else
                     {
                           for(last=list[i];last->next!=NULL;last=last-
>next);
                           last->next=temp;
                     }
               }
```

```
          }
     }
}

main()
{
  int i,j,a[20][20];

  printf("\nEnter no. of vertices: ");
  scanf("%d",&nov);

  creatadjacencylist();
  printf("\n******Adjacency List*******\n");
  display(list);
  printf("\nDFS Sequence: ");
   bfs(list,nov);
}
/*
[root@localhost dsass4]# cc ass4setbq1.c
[root@localhost dsass4]# ./a.out

Enter no. of vertices: 5

Is there edge between V[1] and V[1] (Choose (y/n): n

Is there edge between V[1] and V[2] (Choose (y/n): y

Is there edge between V[1] and V[3] (Choose (y/n): n

Is there edge between V[1] and V[4] (Choose (y/n): n

Is there edge between V[1] and V[5] (Choose (y/n): y

Is there edge between V[2] and V[1] (Choose (y/n): n

Is there edge between V[2] and V[2] (Choose (y/n): n

Is there edge between V[2] and V[3] (Choose (y/n): y

Is there edge between V[2] and V[4] (Choose (y/n): y

Is there edge between V[2] and V[5] (Choose (y/n): n

Is there edge between V[3] and V[1] (Choose (y/n): n

Is there edge between V[3] and V[2] (Choose (y/n): n

Is there edge between V[3] and V[3] (Choose (y/n): n

Is there edge between V[3] and V[4] (Choose (y/n): n

Is there edge between V[3] and V[5] (Choose (y/n): y

Is there edge between V[4] and V[1] (Choose (y/n): n

Is there edge between V[4] and V[2] (Choose (y/n): n
```

```
 Is there edge between V[4] and V[3] (Choose (y/n): n

 Is there edge between V[4] and V[4] (Choose (y/n): n

 Is there edge between V[4] and V[5] (Choose (y/n): y

 Is there edge between V[5] and V[1] (Choose (y/n): n

 Is there edge between V[5] and V[2] (Choose (y/n): n

 Is there edge between V[5] and V[3] (Choose (y/n): n

 Is there edge between V[5] and V[4] (Choose (y/n): n

 Is there edge between V[5] and V[5] (Choose (y/n): n

 ******Adjacency List*******
 V1  2->5->NULL
 V2  3->4->NULL
 V3  5->NULL
 V4  5->NULL
 V5  NULL

 DFS Sequence:
 Queue created
          BFS seq. :
          V1      V2      V5      V3      V4
 */
```

b) Write a C program that accepts the vertices and edges of a graph and store it as an adjacency list. Implement function to traverse the graph using Depth First Search (DFS) traversal.

```
#include<stdio.h>
#include<stdlib.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list[10];
int nov,visited[20];
node *getnodenum(int vno)
{
   node *temp;
   temp=(node*) malloc(sizeof(node));
   temp->data=vno;
   temp->next=NULL;
   return temp;
}
void display(node *list[10])
{
  int i;
```

```
  node *ptr;
  for(i=1;i<=nov;i++)
  {
        printf("V%d  ",i);
        for(ptr=list[i];ptr!=NULL;ptr=ptr->next)
             printf("%d->",ptr->data);
        printf("NULL");
        printf("\n");
  }

}
void creatadjacencylist()
{
      int i,j;
      char ch;
      node *temp,*last;
      for(i=1;i<=nov;i++)
           list[i]=NULL;
      for(i=1;i<=nov;i++)
      {
           for(j=1;j<=nov;j++)
           {
                printf("\nIs there edge between V[%d] and V[%d] (Choose
(y/n): ",i,j);
                scanf(" %c",&ch);
                if(ch=='Y' || ch=='y')
                {
                     temp=getnodenum(j);
                     if(list[i]==NULL)
                          list[i]=temp;
                     else
                     {
                          for(last=list[i];last->next!=NULL;last=last-
>next);
                          last->next=temp;
                     }
                }
           }
      }
}
void recdfs(node *list[10],int nov,int ver)
{
        int i;
        node *ptr;
        visited[ver]=1;
        printf(" V%d",ver);
        //for(i=1;i<=nov;i++)
        for(ptr=list[ver];ptr!=NULL;ptr=ptr->next)
        {

                if((visited[ptr->data]==0))
                               recdfs(list,nov,ptr->data);
                }

}
```

```
main()
{
  int i,j,a[20][20];

  printf("\nEnter no. of vertices: ");
  scanf("%d",&nov);

  creatadjacencylist();
  printf("\n******Adjacency List*******\n");
  display(list);
  printf("\nDFS Sequence: ");
   recdfs(list,nov,1);
}
/*
[root@localhost dsass4]# cc ass4setbq2.c
[root@localhost dsass4]# ./a.out

Enter no. of vertices: 5

Is there edge between V[1] and V[1] (Choose (y/n): n

Is there edge between V[1] and V[2] (Choose (y/n): n

Is there edge between V[1] and V[3] (Choose (y/n): n

Is there edge between V[1] and V[4] (Choose (y/n): y

Is there edge between V[1] and V[5] (Choose (y/n): y

Is there edge between V[2] and V[1] (Choose (y/n): y

Is there edge between V[2] and V[2] (Choose (y/n): n

Is there edge between V[2] and V[3] (Choose (y/n): n

Is there edge between V[2] and V[4] (Choose (y/n): n

Is there edge between V[2] and V[5] (Choose (y/n): n

Is there edge between V[3] and V[1] (Choose (y/n): n

Is there edge between V[3] and V[2] (Choose (y/n): y

Is there edge between V[3] and V[3] (Choose (y/n): n

Is there edge between V[3] and V[4] (Choose (y/n): y

Is there edge between V[3] and V[5] (Choose (y/n): n

Is there edge between V[4] and V[1] (Choose (y/n): n

Is there edge between V[4] and V[2] (Choose (y/n): n

Is there edge between V[4] and V[3] (Choose (y/n): n
```

```
Is there edge between V[4] and V[4] (Choose (y/n): n

Is there edge between V[4] and V[5] (Choose (y/n): n

Is there edge between V[5] and V[1] (Choose (y/n): n

Is there edge between V[5] and V[2] (Choose (y/n): n

Is there edge between V[5] and V[3] (Choose (y/n): y

Is there edge between V[5] and V[4] (Choose (y/n): n

Is there edge between V[5] and V[5] (Choose (y/n): n

******Adjacency List*******
V1  4->5->NULL
V2  1->NULL
V3  2->4->NULL
V4  NULL
V5  3->NULL

DFS Sequence:  V1 V4 V5 V3 V2
*/
```

Set C

a) Which data structure is used to implement Depth First Search?
b) Where the new node is appended in Breadth-first search of OPEN list? c) What is complete graph?