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 3: Graph as Adjacency Matrix

Date: 5/4/23

Set A

a) Write a C program that accepts the vertices and edges of a graph and stores it as an adjacency matrix. Display the adjacency matrix.

```
#include<stdio.h>
int nov,a[20][20];
void creatematrix()
{
     int i,j;
     printf("\nEnter no. of vertices: ");
     scanf("%d",&nov);

     //accept the matrix
     for(i=1;i<=nov;i++)
     {
          for(j=1;j<=nov;j++)
          {
               printf("\nIs there egde between V[%d] and V[%d]: ",i,j);
               scanf("%d",&a[i][j]);
          }
     }

}
void display(int a[20][20])  //print the matrix
{
     int i,j;
     for(i=1;i<=nov;i++)
     {
          for(j=1;j<=nov;j++)
               printf("\t%d",a[i][j]);
          printf("\n");
     }
}

main()
{
     int ch;
     creatematrix();
     printf("\n\t***Adjacency Matrix****\n");
     display(a);
}
/*
[root@localhost ass3]# cc setaq1.c
[root@localhost ass3]# ./a.out

Enter no. of vertices: 4

Is there egde between V[1] and V[1]: 0
```

```
Is there egde between V[1] and V[2]: 1

Is there egde between V[1] and V[3]: 0

Is there egde between V[1] and V[4]: 1

Is there egde between V[2] and V[1]: 0

Is there egde between V[2] and V[2]: 0

Is there egde between V[2] and V[3]: 0

Is there egde between V[2] and V[4]: 1

Is there egde between V[3] and V[1]: 0

Is there egde between V[3] and V[2]: 1

Is there egde between V[3] and V[3]: 0

Is there egde between V[3] and V[4]: 0

Is there egde between V[4] and V[1]: 0

Is there egde between V[4] and V[2]: 0

Is there egde between V[4] and V[3]: 1

Is there egde between V[4] and V[4]: 0

        ***Adjacency Matrix****
        0       1       0       1
        0       0       0       1
        0       1       0       0
        0       0       1       0

*/
```

---

b) Write a C program that accepts the vertices and edges of a graph and store it as an adjacency matrix. Implement functions to print indegree, outdegree and total degree of all vertices of graph.

```
#include<stdio.h>
int nov,a[20][20];
void creatematrix()
{
     int i,j;
     printf("\nEnter no. of vertices: ");
     scanf("%d",&nov);

     //accept the matrix
     for(i=1;i<=nov;i++)
```

```
     {
          for(j=1;j<=nov;j++)
          {
               printf("\nIs there egede between V[%d] and V[%d]: ",i,j);
               scanf("%d",&a[i][j]);
          }
     }

}
void display(int a[20][20])  //print the matrix
{
     int i,j;
     for(i=1;i<=nov;i++)
     {
          for(j=1;j<=nov;j++)
               printf("\t%d",a[i][j]);
          printf("\n");
     }
}

void degree()
{
     int i,j,indegree,outdegree;

     printf("\nVertex\t     Indegree\t     Outdegree\t      Totaldegree
\n");
     for(i=1;i<=nov;i++)
     {
          indegree=0;
          outdegree=0;
          for(j=1;j<=nov;j++)
          {
               if(a[i][j]==1)
                    outdegree+=1;
               if(a[j][i]==1)
                    indegree+=1;
          }

          printf("\nV%d \t\t %d \t\t %d \t\t
%d",i,indegree,outdegree,indegree+outdegree);
     }

}
main()
{
     int ch;
     creatematrix();
     printf("\n\t***Adjacency Matrix****\n");
     display(a);
     degree();
}
/*
[root@localhost ass3]# cc setaq2.c
[root@localhost ass3]# ./a.out

Enter no. of vertices: 4
```

```
Is there egede between V[1] and V[1]: 0

Is there egede between V[1] and V[2]: 1

Is there egede between V[1] and V[3]: 0

Is there egede between V[1] and V[4]: 1

Is there egede between V[2] and V[1]: 0

Is there egede between V[2] and V[2]: 0

Is there egede between V[2] and V[3]: 0

Is there egede between V[2] and V[4]: 1

Is there egede between V[3] and V[1]: 0

Is there egede between V[3] and V[2]: 1

Is there egede between V[3] and V[3]: 0

Is there egede between V[3] and V[4]: 0

Is there egede between V[4] and V[1]: 0

Is there egede between V[4] and V[2]: 0

Is there egede between V[4] and V[3]: 1

Is there egede between V[4] and V[4]: 0

        ***Adjacency Matrix****
        0       1       0       1
        0       0       0       1
        0       1       0       0
        0       0       1       0

Vertex       Indegree        Outdegree        Totaldegree

V1               0               2               2
V2               2               1               3
V3               1               1               2
V4               2               1               3
*/
```

Set B

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

```
//bfs
#include<stdio.h>
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(int a[20][20],int nov)
{
     int ver,j;
     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(j=1;j<=nov;j++)
          {
               if(a[ver][j]==1 && visited[j]==0)
               {
                    add(&q,j);
                    visited[j]=1;
               }
          }
          printf("\t V%d ",ver);
     }
}

void creatematrix()
{
     int i,j;
     printf("\nEnter no. of vertices: ");
     scanf("%d",&nov);

     //accept the matrix
     for(i=1;i<=nov;i++)
     {
          for(j=1;j<=nov;j++)
          {
               printf("\nIs there egde between V[%d] and V[%d]: ",i,j);
               scanf("%d",&a[i][j]);
          }
     }

}
void display(int a[20][20])  //print the matrix
{
     int i,j;
     for(i=1;i<=nov;i++)
     {
          for(j=1;j<=nov;j++)
               printf("\t%d",a[i][j]);
          printf("\n");
     }
}
main()
{
     int ch,i;
     creatematrix();
     printf("\n\t***Adjacency Matrix****\n");
     display(a);

     printf("\nThe Depth First search Traversal(DFS) is:");
     bfs(a,nov);
}
/*
   [root@localhost dsass3]# cc setbq1.c
[root@localhost dsass3]# ./a.out
```

```
Enter no. of vertices: 5

Is there egde between V[1] and V[1]: 0

Is there egde between V[1] and V[2]: 1

Is there egde between V[1] and V[3]: 0

Is there egde between V[1] and V[4]: 0

Is there egde between V[1] and V[5]: 1

Is there egde between V[2] and V[1]: 0

Is there egde between V[2] and V[2]: 0

Is there egde between V[2] and V[3]: 1

Is there egde between V[2] and V[4]: 1

Is there egde between V[2] and V[5]: 0

Is there egde between V[3] and V[1]: 0

Is there egde between V[3] and V[2]: 0

Is there egde between V[3] and V[3]: 0

Is there egde between V[3] and V[4]: 0

Is there egde between V[3] and V[5]: 1

Is there egde between V[4] and V[1]: 0

Is there egde between V[4] and V[2]: 0

Is there egde between V[4] and V[3]: 0

Is there egde between V[4] and V[4]: 0

Is there egde between V[4] and V[5]: 1

Is there egde between V[5] and V[1]: 0

Is there egde between V[5] and V[2]: 0

Is there egde between V[5] and V[3]: 0

Is there egde between V[5] and V[4]: 0

Is there egde between V[5] and V[5]: 0

        ***Adjacency Matrix****
        0       1       0       0       1
        0       0       1       1       0
```

```
         0       0       0       0       1
         0       0       0       0       1
         0       0       0       0       0

 The Depth First search Traversal(DFS) is:
 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 matrix. Implement function to traverse the graph using Depth First Search (DFS) traversal.

```
#include<stdio.h>
int nov,a[20][20];
int visited[20];
void creatematrix()
{
      int i,j;
      printf("\nEnter no. of vertices: ");
      scanf("%d",&nov);

      //accept the matrix
      for(i=1;i<=nov;i++)
      {
            for(j=1;j<=nov;j++)
            {
                  printf("\nIs there egde between V[%d] and V[%d]: ",i,j);
                  scanf("%d",&a[i][j]);
            }
      }

}
void display(int a[20][20])  //print the matrix
{
      int i,j;
      for(i=1;i<=nov;i++)
      {
            for(j=1;j<=nov;j++)
                  printf("\t%d",a[i][j]);
            printf("\n");
      }
}
void recdfs(int a[20][20],int nov,int ver)
{
      int i;
      visited[ver]=1;
      printf(" V%d",ver);
      for(i=1;i<=nov;i++)
      {

            if((a[ver][i]==1) && (visited[i]==0))
```

```
                        recdfs(a,nov,i);
            }

}
main()
{
      int ch,i;
      creatematrix();
      printf("\n\t***Adjacency Matrix****\n");
      display(a);

         printf("\nThe Depth First search Traversal(DFS) is:");
         recdfs(a,nov,1);
}
/*
[root@localhost ass3]# cc setbq2.c
[root@localhost ass3]# ./a.out

Enter no. of vertices: 5

Is there egde between V[1] and V[1]: 0

Is there egde between V[1] and V[2]: 0

Is there egde between V[1] and V[3]: 1

Is there egde between V[1] and V[4]: 1

Is there egde between V[1] and V[5]: 0

Is there egde between V[2] and V[1]: 0

Is there egde between V[2] and V[2]: 0

Is there egde between V[2] and V[3]: 1

Is there egde between V[2] and V[4]: 0

Is there egde between V[2] and V[5]: 1

Is there egde between V[3] and V[1]: 0

Is there egde between V[3] and V[2]: 1

Is there egde between V[3] and V[3]: 0

Is there egde between V[3] and V[4]: 0

Is there egde between V[3] and V[5]: 0

Is there egde between V[4] and V[1]: 0

Is there egde between V[4] and V[2]: 0

Is there egde between V[4] and V[3]: 0
```

```
Is there egde between V[4] and V[4]: 0

Is there egde between V[4] and V[5]: 1

Is there egde between V[5] and V[1]: 0

Is there egde between V[5] and V[2]: 0

Is there egde between V[5] and V[3]: 0

Is there egde between V[5] and V[4]: 0

Is there egde between V[5] and V[5]: 0

        ***Adjacency Matrix****
        0       0       1       1       0
        0       0       1       0       1
        0       1       0       0       0
        0       0       0       0       1
        0       0       0       0       0

The Depth First search Traversal(DFS) is: V1 V3 V2 V5 V4

*/
```

Set C

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