**SSN College of Engineering, Kalavakkam**

**Department of Computer Science and Engineering**

**III Semester**

**UCS 1312 Data Structures Lab Laboratory**

**Academic Year: 2019-2020 Batch: 2018-2022**

**Name: S. Prathyush**

**Class: CSE Sec: B**

**Roll No:- 185001112**

**ASSIGNMENT – 11: GRAPH TRAVERSAL**

#include<stdio.h>  
#define MAX 25  
#define QUEUE\_SIZE 25  
int queue[QUEUE\_SIZE];  
int queue\_front, queue\_end;  
void dfs(int adj[][MAX],int visited[],int start,int n,int sv)  
{  
    int stack[MAX];  
int top=-1,i;  
    if(sv==1)  
{  
       printf("%c ",start+65);  
}   else{  
         printf("%d ",start);}  
    visited[start]=1;  
    stack[++top]=start;  
    while(top!=-1)  
    {  
        start=stack[top];  
        for(i=0;i<n;i++)  
        {  
            if(adj[start][i] && visited[i]==0)  
            {  
                stack[++top]=i;  
if(sv==1){  
                printf("%c ",i+65);}  
else{  
printf("%d ",i);}  
                visited[i]=1;  
                break;  
            }  
        }  
        if(i==n)  
            top--;  
    }  
}  
void enqueue(int v) {  
  queue[queue\_end] = v;  
  queue\_end++;  
}  
  
int dequeue() {  
  int index = queue\_front;  
  queue\_front++;  
  return queue[index];  
}  
  
  
void bfs(int Adj[][MAX], int n, int source,int sv) {  
  int i, j;  
   
  //visited array to flag the vertex that  
  //were visited  
  int visited[MAX];  
  
  queue\_front = 0;  
  queue\_end = 0;  
   
  //set visited for all vertex to 0 (means unvisited)  
  for(i = 0; i < MAX; i++) {  
    visited[i] = 0;  
  }  
   
  //mark the visited source  
  visited[source] = 1;  
   
  //enqueue visited vertex  
  enqueue(source);  
   
  //print the vertex as result  
 if(sv==1)  
{  
       printf("%c ",source+65);  
}   else{  
         printf("%d ",source);}  
  
   
   
  //continue till queue is not empty  
  while(queue\_front <= queue\_end) {  
    //dequeue first element from the queue  
    i = dequeue();  
     
    for(j = 0; j < n; j++) {  
      if(visited[j] == 0 && Adj[i][j] == 1) {  
        //mark vertex as visited  
        visited[j] = 1;  
         
        //push vertex into queue  
        enqueue(j);  
         
        //print the vertex as result  
if(sv==1)  
{  
       printf("%c ",j+65);  
}   else{  
         printf("%d ",j);}  
      }  
    }  
  }  
  printf("\n");  
}  
  
int main()  
{  
    int adj[MAX][MAX];  
    int visited[MAX]={0},i,j;  
    int n,sv,c=1;  
    while(c!=0)  
    {  
    printf("Enter the number of vertices:");  
        scanf("%d",&n);  
for(i=0;i<n;i++)  
    {  
for(j=0;j<n;j++)  
{  
    printf("ENTER 1 IF %d has a node with %d ELSE 0 ",i,j);  
             scanf("%d",&adj[i][j]);  
}  
}  
printf("THE ADJACENCY MATRIX IS\n");  
for(i=0;i<n;i++)  
{  
for(j=0;j<n;j++)  
{  
printf(" %d",adj[i][j]);  
}  
printf("\n");  
}  
printf("1-alphabet graph\n2-number graph\n");  
        printf("Enter choice:");  
        scanf("%d",&sv);  
        printf("DFS Traversal is:");  
        dfs(adj,visited,0,n,sv);  
        printf("\n");  
        printf("BFS Traversal is:");  
        bfs(adj, n,0,sv );  
        printf("Do you want to continue:1-YES; 0-NO:");  
        scanf(" %d",&c);  
    }        
    return 0;  
}

**OUTPUT**

Enter the number of vertices:5  
ENTER 1 IF 0 has a node with 0 ELSE 0 0  
ENTER 1 IF 0 has a node with 1 ELSE 0 1  
ENTER 1 IF 0 has a node with 2 ELSE 0 1  
ENTER 1 IF 0 has a node with 3 ELSE 0 0  
ENTER 1 IF 0 has a node with 4 ELSE 0 1  
ENTER 1 IF 1 has a node with 0 ELSE 0 1  
ENTER 1 IF 1 has a node with 1 ELSE 0 0  
ENTER 1 IF 1 has a node with 2 ELSE 0 0  
ENTER 1 IF 1 has a node with 3 ELSE 0 1  
ENTER 1 IF 1 has a node with 4 ELSE 0 1  
ENTER 1 IF 2 has a node with 0 ELSE 0 1  
ENTER 1 IF 2 has a node with 1 ELSE 0 0  
ENTER 1 IF 2 has a node with 2 ELSE 0 0  
ENTER 1 IF 2 has a node with 3 ELSE 0 0  
ENTER 1 IF 2 has a node with 4 ELSE 0 0  
ENTER 1 IF 3 has a node with 0 ELSE 0 0  
ENTER 1 IF 3 has a node with 1 ELSE 0 1  
ENTER 1 IF 3 has a node with 2 ELSE 0 0  
ENTER 1 IF 3 has a node with 3 ELSE 0 0  
ENTER 1 IF 3 has a node with 4 ELSE 0 0  
ENTER 1 IF 4 has a node with 0 ELSE 0 1  
ENTER 1 IF 4 has a node with 1 ELSE 0 1  
ENTER 1 IF 4 has a node with 2 ELSE 0 0  
ENTER 1 IF 4 has a node with 3 ELSE 0 0  
ENTER 1 IF 4 has a node with 4 ELSE 0 0  
THE ADJACENCY MATRIX IS  
 0 1 1 0 1  
 1 0 0 1 1  
 1 0 0 0 0  
 0 1 0 0 0  
 1 1 0 0 0  
1-alphabet graph  
2-number graph  
Enter choice:1      
DFS Traversal is:A B D E C  
BFS Traversal is:A B C E D  
Do you want to continue:1-YES; 0-NO:1  
Enter the number of vertices:5  
ENTER 1 IF 0 has a node with 0 ELSE 0 0  
ENTER 1 IF 0 has a node with 1 ELSE 0 1  
ENTER 1 IF 0 has a node with 2 ELSE 0 0  
ENTER 1 IF 0 has a node with 3 ELSE 0 0  
ENTER 1 IF 0 has a node with 4 ELSE 0 0  
ENTER 1 IF 1 has a node with 0 ELSE 0 0  
ENTER 1 IF 1 has a node with 1 ELSE 0 0  
ENTER 1 IF 1 has a node with 2 ELSE 0 1  
ENTER 1 IF 1 has a node with 3 ELSE 0 0  
ENTER 1 IF 1 has a node with 4 ELSE 0 0  
ENTER 1 IF 2 has a node with 0 ELSE 0 0  
ENTER 1 IF 2 has a node with 1 ELSE 0 0  
ENTER 1 IF 2 has a node with 2 ELSE 0 0  
ENTER 1 IF 2 has a node with 3 ELSE 0 1  
ENTER 1 IF 2 has a node with 4 ELSE 0 1  
ENTER 1 IF 3 has a node with 0 ELSE 0 1  
ENTER 1 IF 3 has a node with 1 ELSE 0 0  
ENTER 1 IF 3 has a node with 2 ELSE 0 0  
ENTER 1 IF 3 has a node with 3 ELSE 0 0  
ENTER 1 IF 3 has a node with 4 ELSE 0 0  
ENTER 1 IF 4 has a node with 0 ELSE 0 0  
ENTER 1 IF 4 has a node with 1 ELSE 0 0  
ENTER 1 IF 4 has a node with 2 ELSE 0 1  
ENTER 1 IF 4 has a node with 3 ELSE 0 0  
ENTER 1 IF 4 has a node with 4 ELSE 0 0  
THE ADJACENCY MATRIX IS  
 0 1 0 0 0  
 0 0 1 0 0  
 0 0 0 1 1  
 1 0 0 0 0  
 0 0 1 0 0  
1-alphabet graph  
2-number graph  
Enter choice:2  
DFS Traversal is:0  
BFS Traversal is:0 1 2 3 4  
Do you want to continue:1-YES; 0-NO:0