**Practical No. 10**

AIM: To write a menu-driven program for DFS and BFS.

PROGRAM:

#include<stdio.h>

#include<stdlib.h>

#define MAX 20

typedef struct Q

{

int data[MAX];

int R,F;

}Q;

typedef struct node

{

struct node \*next;

int vertex;

}node;

void enqueue(Q \*,int);

int dequeue(Q\*);

int empty(Q\*);

int full(Q\*);

void BFS(int);

void readgraph();

void insert(int vi, int vj);

void DFS(int i);

int visited[MAX];

node \*G[20];

int n;

void main()

{

int i,op;

do

{

printf("\n\n1.Create\n2.BFS\n3.DFS\n4.Quit");

printf("\nEnter Your Choice:");

scanf("%d",&op);

switch(op)

{

case 1:

readgraph();

break;

case 2:

printf("\nStarting Node No. :");

scanf("%d",&i);

BFS(i);

break;

case 3:

for(i=0;i<n;i++)

visited[i]=0;

printf("\nStarting Node No. :");

scanf("%d",&i);

DFS(i);

break;

}

}while(op!=4);

}

void BFS(int v)

{

int w,i,visited[MAX];

Q q;

node \*p;

q.R=q.F=-1;

for(i=0;i<n;i++)

visited[i]=0;

enqueue(&q,v);

printf("\nVisit\t%d",v);

visited[v]=1;

while(!empty(&q))

{

v=dequeue(&q);

for(p=G[v];p!=NULL;p=p->next)

{

w=p->vertex;

if(visited[w]==0)

{

enqueue(&q,w);

visited[w]=1;

printf("\nvisit\t%d",w);

}

}

}

}

void DFS(int i)

{

node \*p;

printf("\n%d",i);

p=G[i];

visited[i]=1;

while(p!=NULL)

{

i=p->vertex;

if(!visited[i])

DFS(i);

p=p->next;

}

}

int empty(Q \*P)

{

if(P->R==-1)

return(1);

return(0);

}

int full(Q \*P)

{

if(P->R==MAX-1)

return(1);

return(0);

}

void enqueue(Q \*P,int x)

{

if(P->R==-1)

{

P->R=P->F=0;

P->data[P->R]= x;

}

else

{

P->R=P->R+1;

P->data[P->R]=x;

}

}

int dequeue(Q \*P)

{

int x;

x=P->data[P->F];

if(P->R==P->F)

{

P->R=-1;

P->F=-1;

}

else

P->F=P->F+1;

return(x);

}

void readgraph()

{

int i,vi,vj,no\_of\_edges;

printf("\nEnter no. of Vertices:");

scanf("%d",&n);

for(i=0;i<n;i++)

G[i]=NULL;

printf("\nEnter no of edges:");

scanf("%d",&no\_of\_edges);

for(i=0;i<no\_of\_edges;i++)

{

printf("\nEnter an edge (u,v):");

scanf("%d%d",&vi,&vj);

insert(vi,vj);

insert(vj,vi);

}

}

void insert(int vi,int vj)

{

node \*p,\*q;

q=(node \*)malloc(sizeof(node));

q->vertex=vj;

q->next=NULL;

if(G[vi]==NULL)

G[vi]=q;

else

{

p=G[vi];

while(p->next!=NULL)

p=p->next;

p->next=q;

}

}

OUTPUT:

1.Create

2.BFS

3.DFS

4.Quit

Enter Your Choice:1

Enter no. of Vertices:4

Enter no of edges:3

Enter an edge (u,v):1 2

Enter an edge (u,v):1 3

Enter an edge (u,v):3 4

1.Create

2.BFS

3.DFS

4.Quit

Enter Your Choice:2

Starting Node No. :1

Visit   1

visit   2

visit   3

visit   4

1.Create

2.BFS

3.DFS

4.Quit

Enter Your Choice:3

Starting Node No. :1

1

2

3

4

1.Create

2.BFS

3.DFS

4.Quit

Enter Your Choice: