LAB 1

Q1. DFS and BFS using adjacency matrix

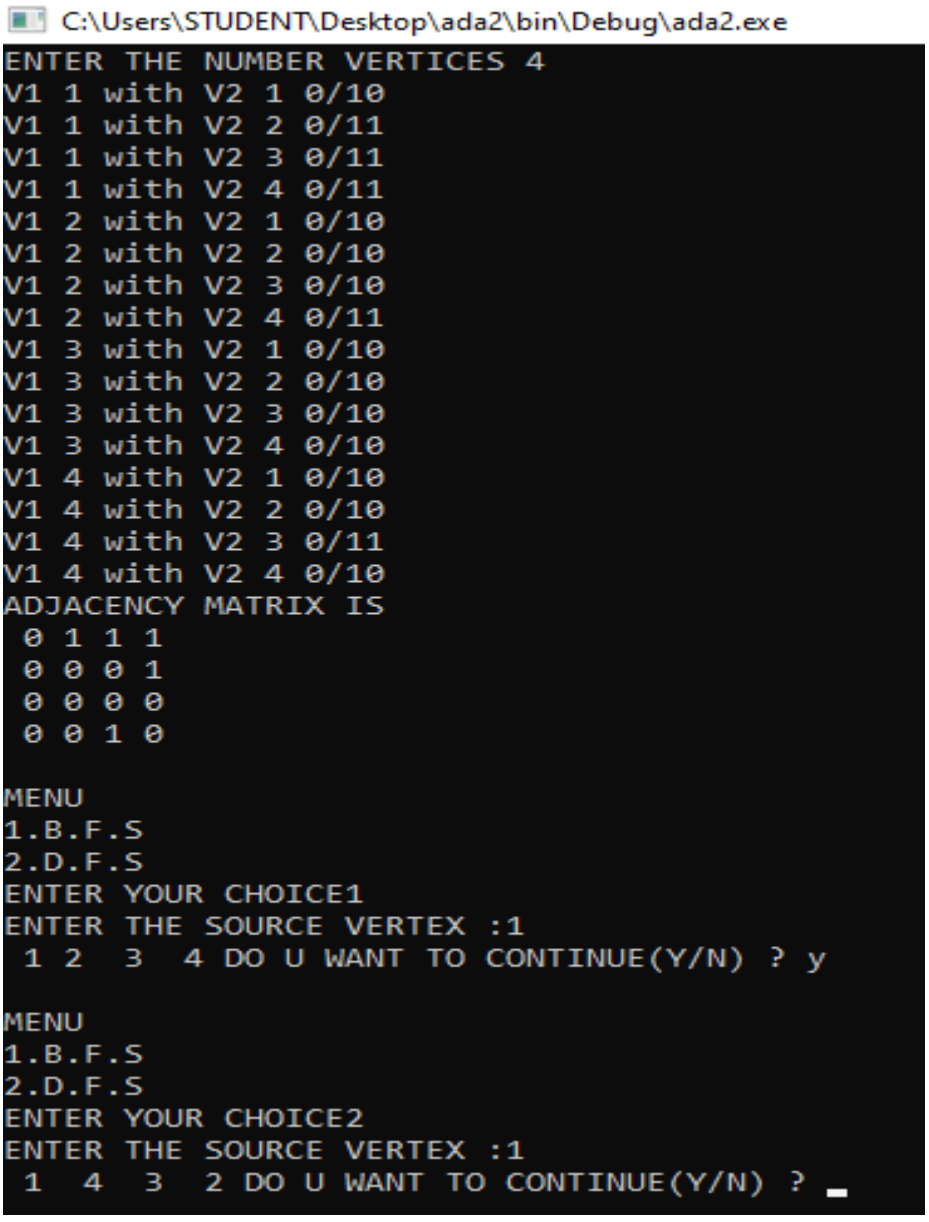
Aim: C program for DFS and BFS algorithm

```
#include<stdio.h>

int q[20],top=-1,front=-1,rear=-1,a[20][20],vis[20],stack[20];
int delete();
void add(int item);
void bfs(int s,int n);
void dfs(int s,int n);
void push(int item);
int pop();

void main()
{
int n,i,s,ch,j;
char c,dummy;
printf("ENTER THE NUMBER VERTICES ");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf("ENTER 1 IF %d HAS A NODE WITH %d ELSE 0 ",i,j);
scanf("%d",&a[i][j]);
}
}
printf("ADJACENCY MATRIX IS\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf(" %d",a[i][j]);
}
printf("\n");
}

do
{
for(i=1;i<=n;i++)
vis[i]=0;
printf("\nMENU");
printf("\n1.B.F.S");
printf("\n2.D.F.S");
printf("\nENTER YOUR CHOICE");
scanf("%d",&ch);
printf("ENTER THE SOURCE VERTEX :");
scanf("%d",&s);

switch(ch)
{
case 1:bfs(s,n);
break;
```

```
case 2:
dfs(s,n);
break;
}
printf("DO U WANT TO CONTINUE(Y/N) ? ");
scanf("%c",&dummy);
scanf("%c",&c);
}while((c=='y')||(c=='Y'));
}



void bfs(int s,int n)
{
int p,i;
add(s);
vis[s]=1;
p=delete();
if(p!=0)
printf(" %d",p);
while(p!=0)
{
for(i=1;i<=n;i++)
if((a[p][i]!=0)&&(vis[i]==0))
{
add(i);
vis[i]=1;
}
p=delete();
if(p!=0)
printf(" %d ",p);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
bfs(i,n);
}


void add(int item)
{
if(rear==19)
printf("QUEUE FULL");
else
{
if(rear==-1)
{
q[++rear]=item;
front++;
}
else
q[++rear]=item;
}
}
int delete()
{
```

```
int k;
if((front>rear)||(front==-1))
return(0);
else
{
k=q[front++];
return(k);
}
}



void dfs(int s,int n)
{
int i,k;
push(s);
vis[s]=1;
k=pop();
if(k!=0)
printf(" %d ",k);
while(k!=0)
{
for(i=1;i<=n;i++)
if((a[k][i]!=0)&&(vis[i]==0))
{
push(i);
vis[i]=1;
}
k=pop();
if(k!=0)
printf(" %d ",k);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
dfs(i,n);
}
void push(int item)
{
if(top==19)
printf("Stack overflow ");
else
stack[++top]=item;
}
int pop()
{
int k;
if(top==-1)
return(0);
else
{
k=stack[top--];
return(k);
}
}
```

RESULT:


C:\Users\STUDENT\Desktop\ada2\bin\Debug\ada2.exe
ENTER THE NUMBER VERTICES 4
V1 1 with V2 1 0/10
V1 1 with V2 2 0/11
V1 1 with V2 3 0/11
V1 1 with V2 4 0/11
V1 2 with V2 1 0/10
V1 2 with V2 2 0/10
V1 2 with V2 3 0/10
V1 2 with V2 4 0/11
V1 3 with V2 1 0/10
V1 3 with V2 2 0/10
V1 3 with V2 3 0/10
V1 3 with V2 4 0/10
V1 4 with V2 1 0/10
V1 4 with V2 2 0/10
V1 4 with V2 3 0/11
V1 4 with V2 4 0/10
ADJACENCY MATRIX IS
0 1 1 1
0 0 0 1
0 0 0 0
0 0 1 0
MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE1
ENTER THE SOURCE VERTEX :1
1 2 3 4 DO U WANT TO CONTINUE(Y/N) ? y
MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE2
ENTER THE SOURCE VERTEX :1
1 4 3 2 DO U WANT TO CONTINUE(Y/N) ?

15/6/23 Exp-1

DFS & BFS using adjacency matrix

Soln.

```
#include<stdio.h>
int q[20], top=-1, front=-1, rear=-1, a[20][20], vis[20],
    stack[20];
int delete();
void add(int item);
void bfs(int s, int n);
void dfs(int s, int n);
void push(int item);
int pop();
void main()
{
 int n, s, i, ch, j;
 char c, dq;
 printf("Enter the number of vertices");
 scanf("%d", &n);
 for (i=1; i<n; i++)
 {
  for(j=1; j<=n; j++){
  printf("enter 1 if %d has a node with %d else 0",
        i, j);
  scanf("%d", &a[i][j]);
  }
 }
 printf("The Adjacency matrix\n");
 for(i=1; i<=n; i++){
 for(j=1; j<=n; j++){
  printf("%d", a[i][j]); }
 printf("\n");
 }
 do
 {
  for(i=1; i<=n; i++)
  vis[i] = 0;
  printf("\n Menu");
  printf("1.B.F.S");
  printf("\n2. DFS");
  printf("Enter your choice");
  scanf("%d", &ch);
  printf("Enter the source vertex");
  scanf("%d", &s);
  switch(ch)
  {
  case 1 : bfs(s, n);
  break;
  case 2: dfs(s, n);
   break;
  }
 printf("Do you want to continue(Y/n)?");
  scanf("%c", &dq);
  scanf("%c", &c);
  }
  while((c=='Y' || c=='N'));
 }
void bfs(int s, int n)
{
 int p, i;
 add(s);
```

```
vis[s]=1;
p= delete();
if (p!=0)
printf("%d ", p);
while (p!=0)
{
for (i=1; i<=n; i++)
if ((a[p][i]!=0) && (vis[i]==0))
{
add(i);
vis[i]=1;
}
p= delete();
if (p!=0)
printf("%d ", p);
}
for (i=1; i<=n; i++)
  if (vis[i]==0)
  bfs (i, n);
}
void add (int item)
{
if (rear==19)
printf("Queue is full");
else
{
if (rear==-1)
{
q[++rear]=item;
front++;
}
else
q[++rear]=item;
}
```

```
}
int delete()
{
int k;
if ((front>rear)|| front==-1))
return (0);
else
{
k=q[front++];
return(k);
}
}
void dfs (int s, int n)
{
int i,k;
push(s);
vis[s]=1;
k= pop();
if (k!=0)
printf("%d ", k);
while (k!=0)
{
for (i=1; i<=n; i++)
if ((a[k][i]!=0) && (vis[i]==0))
{
 push(i);
 vis[i]=1.
}
k= pop();
if (k!=0)
printf("%d ",k);
}
```

```
for (i=1; i<=n ; i++)
if (vis[i]==0)
dfs(i, n);
}
void push(int item)
{
if (top==19)
printf("stack overflow");
else
stack[++top]=item;
}
int pop()
{
int k;
if (top==-1)
return(0);
else
{
k=stack[top--];
return(k);
}
}
```

Result:-

Enter number of vertices 4

M with v_1 1/0 : 0

v_1 v_2 : 1

v_1 v_3 : 1

v_1 v_4 : 1

v_2 with v_1 1/0 : 0

v_2 v_2 : 0

v_2 v_3 : 0

v_2 v_4 : 1

Given Adjacency matrix

0	1	1	1
0	0	0	1
0	0	0	0
0	0	1	0

v_1 v_6 = 0

v_1 v_7 = 0

v_2 with v_1 = 0

v_2 v_2 = 0

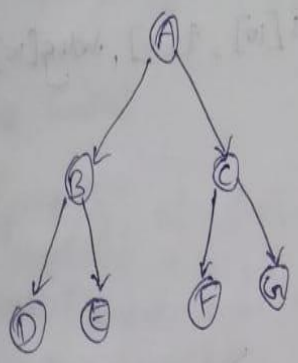
v_2 v_3 = 0

v_2 v_4 = 1

v_2 v_5 = 1

v_2 v_6 = 0

v_2 v_7 = 0;



BFS :- A B C D E F G

v_1 v_2 v_3 v_4 v_5 v_6 v_7

DFS :- A B D E C F G

v_1 v_2 v_4 v_5 v_3 v_6 v_7

Sept
16/6/23