

ADA LAB PROGRAM

PROGRAM:-

- a) Print all the nodes reachable from a given starting node in a digraph using BFS method.
- b) check whether a given graph is connected or not using DFS method.

Modification:-

- given an undirected graph, print all components line by line.

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
void insertq (int q[], int node, int *f, int *a)
```

```
{ if (*f == -1) && (*a == -1)
```

```
{ (*f)++, (*a)++, q[*f] = node;
```

```
}
```

```
else {
```

```
    (*a)++, q[*a] = node;
```

```
}
```

```
}
```

```
int deleteq (int q[], int *f, int *a)
```

```
{ int temp;
```

```
temp = q[*f];
```

```
if (*f == *a) *f = *a = -1;
```

```
else (*f)++;
```

```
return temp;
```

```
}
```

```
void bfs (int n, int adj[][10], int visited[])
```

```
{ int q[20], f = -1, a = -1, v, i;
```

```
insertq (q, src, &f, &a);
```

```
while ((f <= a) && (f != -1))
```

```
{ v = deleteq (q, &f, &a);
```

```
if (visited[v] != 1)
```

```
{
```

```
    visited[v] = 1;
```

```
    printf("%d", v);
```

```
}
```

```
for (i=1; i<=n; i++)
```

```
if (1 adj[V][i] == 1) && (visited[i] != 1)
```

```
insert_q(q, i, &f, &x);
```

```
}
```

```
void DFS (int n, int cost [10][10], int u, int s[])
```

```
{ int v;
```

```
s[u] = 1;
```

```
for (v=0; v<n; v++)
```

```
{ if (1 cost[u][v] == 1) && (s[v] == 0)
```

```
DFS (n, cost, v, s);
```

```
}
```

```
int main () {
```

```
int n, v, s, adj [10][10], sec, visited [10], choice;
```

```
int cost [10][10];
```

```
int s [10], con, flag;
```

```
for (;;) {
```

```
printf ("1. Print the Reachable nodes \n .
```

```
2. Check the connectivity of the graph \n
```

```
3. exit \n");
```

```
printf (" Enter the choice :");
```

```
scanf ("%d", &choice);
```

```
switch (choice) {
```

```
case 1: printf ("enter number of vertices \n");
```

```
scanf ("%d", &n);
```

```
printf ("enter adjacency matrix \n");
```

```
for (i=1; i<=n; i++)
```

```
{
```

```
visited[i] = 0;
```

```

for (j=1; j <= n; j++)
    scanf ("%d", &adj[i][j]);
}
printf ("Enter starting vertex \n");
scanf ("%d", &src);
printf ("The nodes reachable are \n");
bfs (n, adj, src, visited);
break;

```

```

Case 2: printf ("Enter number of nodes \n");
scanf ("%d", &n);
printf ("Enter the adjacency matrix \n");
for (i=0; i < n; i++)
{
    for (j=0; j < n; j++)
        scanf ("%d", &cost[i][j]);
}

```

```

}
con = 0;
for (j=0; j < n; j++)
{
    for (i=0; i < n; i++)
        s[i] = 0;
    DFS (n, cost, j, s);
    flag = 0;
    for (i=0; i < n; i++)
    {
        if (s[i] == 0)
            flag = 1;
    }
    if (flag == 0)
        con = 1;
}
}

```

```
if (con == 1)
```

```
    printf ("Graph is connected\n");
```

```
else
```

```
    printf ("Graph is not connected\n");
```

```
    break;
```

```
default : exit(0);
```

```
}
```

```
}
```

```
}
```

4