

1. Write program to do the following:
 - 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.

Code:

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

int a[10][10],n,vis[10];
int dfs(int root){
    int j;
    vis[root]=1;
    for(j=1;j<=n;j++)
        if(a[root][j]==1&&vis[j]!=1)
            dfs(j);
    for(j=1;j<=n;j++) {
        if(vis[j]!=1)
            return 0;
    }
    return 1;
}

void main()
{
    int i,j,root,ans;
    for(j=1;j<=n;j++)
        vis[j]=0;
    printf("\nEnter the no of nodes:\t");
    scanf("%d",&n);
    printf("\nEnter the adjacency matrix:\n");
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            scanf("%d",&a[i][j]);
    printf("\nEnter the source node:\t");
    scanf("%d",&root);
    ans=dfs(root);
    if(ans==1)
        printf("\nGraph is connected\n");
    else
        printf("\nGraph is not connected\n");
    getch();
}
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS D:\VS Code> cd "d:\VS Code\OS\" ; if ($?) { gcc ada.c -o ada } ; if ($?) { .\ada }

Enter the no of nodes: 4

Enter the adjacency matrix:
0 1 1 1
0 0 0 1
0 0 0 0
0 0 1 0

Enter the source node: 1

Graph is connected
█
```

Code:

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

int a[15][15],n;
void bfs(int);

void main() {
    int i,j,root;

    printf("\nEnter the no of nodes:\t");

    scanf("%d",&n);

    printf("\nEnter the adjacency matrix:\n");

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

        for(j=1;j<=n;j++)

            scanf("%d",&a[i][j]);

    printf("\nEnter the source node:\t");

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

```

    bfs(root);

}

void bfs(int root) {

    int q[15],f=0,r=-1,vis[15],i,j;

    for(j=1;j<=n;j++)

        vis[j]=0;

    vis[root]=1;

    r=r+1;

    q[r]=root;

    while(f<=r) {

        i=q[f];

        f=f+1;

        for(j=1;j<=n;j++)

        {

            if(a[i][j]==1&&vis[j]!=1) {

                vis[j]=1;

                r=r+1;

                q[r]=j;

            }

        }

    }

    for(j=1;j<=n;j++) {

```

```
if(vis[j]!=1)

printf("\nNode %d is not reachable",j);

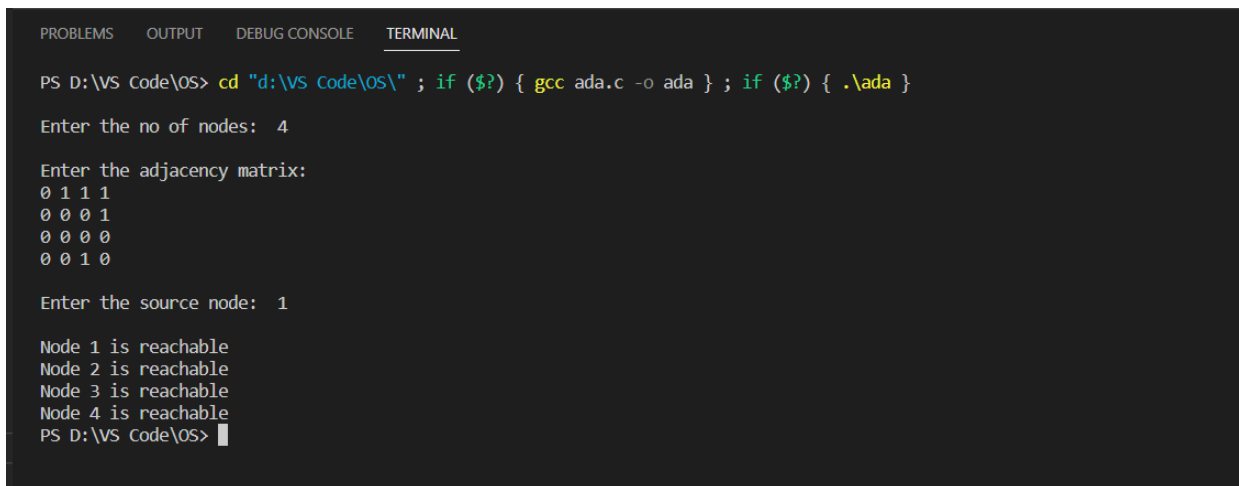
else

printf("\nNode %d is reachable",j);

}

}
```

Output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS D:\VS Code\OS> cd "d:\VS Code\OS\" ; if ($?) { gcc ada.c -o ada } ; if ($?) { .\ada }

Enter the no of nodes: 4

Enter the adjacency matrix:
0 1 1 1
0 0 0 1
0 0 0 0
0 0 1 0

Enter the source node: 1

Node 1 is reachable
Node 2 is reachable
Node 3 is reachable
Node 4 is reachable
PS D:\VS Code\OS> █
```

Observation:

1) DFS, BFS

#include <stdio.h>

void initGraph(Graph *graph, int vertices) {

graph->vertices = vertices;

for (int i = 1; i <= vertices; i++) {

for (int j = 1; j <= vertices; j++) {

graph->matrix[i][j] = 0;

}

}

}

void addEdge(Graph *graph, int src, int dest) {

graph->matrix[src][dest] = 1;

}

void BFS(Graph *graph, int start) {

bool visited[MAX_VERTICES] = {false};

int queue[MAX_VERTICES];

int front = 0, rear = 0, temp;

visited[start] = true;

queue[rear++] = start;

printf("BFS Traversal:");

```

while (front != rear) {
    int current = graph[front++];
    printf("%d", current);
    for (int i = 1; i <= graph->vertices; i++) {
        if (graph->matrix[current][i] == 1 && !visited[i]) {
            visited[i] = true;
            graph->next[i] = current;
        }
    }
    printf("\n");
}

```

```

void DFSUtil(int graph, int current, bool visited)

```

```

{
    visited[current] = true;

```

```

    printf("%d", current);

```

```

    for (int i = 1; i <= graph->vertices; i++) {

```

```

        if (graph->matrix[current][i] == 1 && !visited[i]) {

```

```

            DFSUtil(graph, i, visited);

```

```

        }
    }
}

```

```

void DFS (Graph *graph, int start) {
    bool visited [MAX_VERTICES] = {false};
    printf ("DFS starts at: %d\n", start);
    DFSUtil (graph, start, visited);
}

void DFSUtil (Graph *graph, int start, bool visited[]) {
    printf ("Adjacency list: ");
    for (int i = 1; i <= graph->vertices; i++) {
        for (int j = 1; j <= graph->vertices; j++) {
            printf ("%d ", graph->adjacency[i][j]);
            if (j % 10 == 0) printf ("\n");
        }
    }
    printf ("\n");
}

void NodeExistsFromFib (Graph *graph, const char *fibrow) {
    FILE *fpu = fopen (fibrow, "r");
    if (fpu == NULL) {
        printf ("Could not open file\n");
        return;
    }
}

```

```

int main() {
    int vertices;
    Graph g;

    printf("Enter the no. of vertices: ");
    scanf("%d", &vertices);

    int initGraph(&g, vertices);

    int choice, src, dest, start;

    do {
        printf("\n 1. Add edge in 2 DFS in 3, BFS\n\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch(choice) {
            case 1: printf("Enter edge (2 or destination): ");
                    scanf("%d %d", &src, &dest);
                    addEdge(&g, src, dest);
                    break;

```


Case 2:

```
printf ("Now starting vertex.");
break;
DFS (graph, start, 2);
break;
```

Case 3:

```
printf ("Now starting vertex.");
break;
DFS (graph, start, 3);
```

```
DFS (graph, start, 4);
break;
```

Case 4:

```
printf ("Now starting vertex.");
break;
```

Case 5:

```
printf ("Exiting \n");
break;
```

Default:

```
printf ("Invalid choice! please try again.");
break;
```

if

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

```
return 0;
```

}

Don't
put

