DFS

```
#include<stdio.h>
#include<bits/stdc++.h>
using namespace std;
void DFS(int);
int G[10][10],visited[10]={0},n;
int main()
{
	int i,j;
	printf("Enter number of vertices:");
	scanf("%d",&n);
	printf("\nEnteradjecency matrix of the graph:\n");
	for(i=0;i<n;i++)
	{
		for(j=0;j<n;j++)
			scanf("%d",&G[i][j]);
	}
	DFS(0);
}
void DFS(int i)
{
	int j;
	printf("\n%d",i);
	visited[i]=1;
	for(j=0;j<n;j++)
	if(visited[j]==0&&G[i][j]==1)
```

```
        DFS(j);
}
```

INPUT:

Enter number of vertices:10

Enteradjecency matrix of the graph:

0 1 0 1 0 0 0 0 0 0

0 0 1 0 1 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 1 0

0 0 0 0 1 1 0 0 0 0

0 0 0 0 0 0 1 0 1 0

0 0 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 1 0 0

0 0 0 0 0 0 0 0 0 0

Output:

0 1 2 4 5 6 9 8 7 3

BFS

```
#include<bits/stdc++.h>

using namespace std;

int visited[10];
int vertex[10][10];

void BFS();

int main()
{
   int edge;
   cout<<"Enter no of Edges :\n";
   cin>>edge;
   cout<<"enter nodes:"
      int index,j;
   for(int i=0; i<edge; i++)
   {
      cin>>index>>j;
      vertex[index][j] = 1;
      vertex[j][index] = 1;
   }

}
```

```
void BFS()
{
   queue<int> que;
   que.push(0);
   visited[0] = 1;
   while(! que.empty())
   {
      int v = que.front();
      for(int i=0; i<10; i++)
      {
         if(vertex[v][i] == 1)
            if(visited[i] == 0)
            {
               visited[i] = 1;
               que.push(i);
            }
      }
      cout<<v<<" ";
      que.pop();
   }
}
```

INPUT:

```
Enter no of Edges :
10
Enter Nodes:
0 1
```

0 3

1 2

1 4

3 4

3 5

4 8

5 6

5 8

6 9

OUTPUT:

0 1 3 2 4 5 8 6 9

Process returned 0 (0x0) execution time : 91.131 s

Press any key to continue.