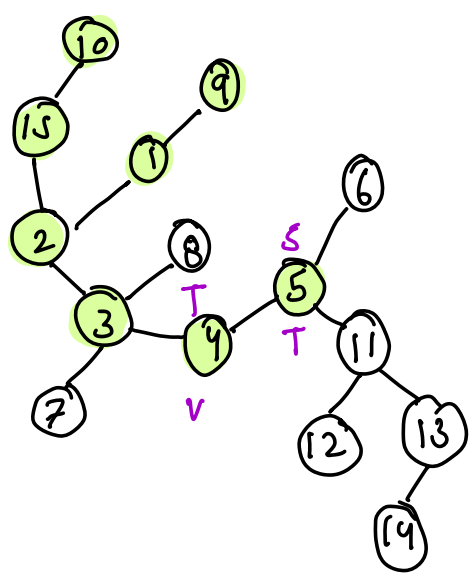
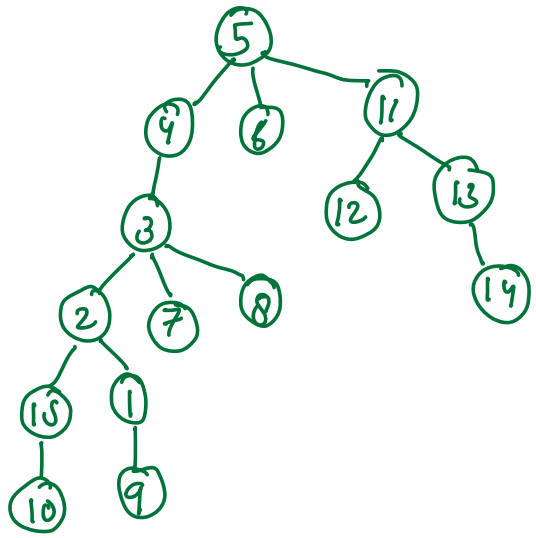
```
void dfs( list <int> g[], bool vis[], int s)
{

    vis [s] = true;
    for (int i=0; i<g [s].size(); i++)
    {
        int v = g [s] [i];
        dfs(g, vis , v);
    }
}
```

DFS (Depth first search)





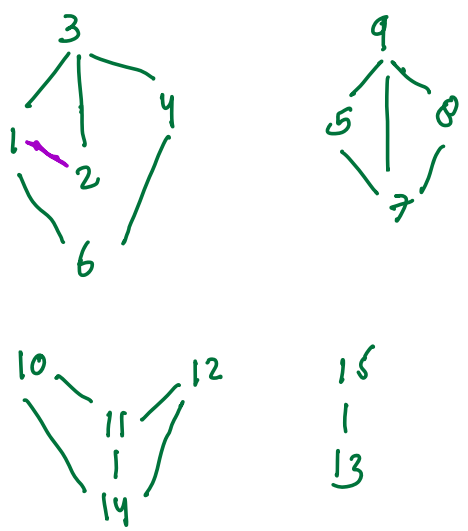

```
bool path (int N, int E, int u[], int v[], s, d)
{
    list <int> g[N+1]      // ToDo
    bool vis[N+1] = {F}
    dfs( g, vis, s)        // to fill vis[].

}
```

```
void dfs( list <int> g[], bool vis[], int s)
{
    if (vis[s] == True) return;

    vis[s] = true;
    for (int i=0; i<g[s].size(); i++)
    {
        int v = g[s][i];
        dfs(g, vis, v);
    }
}
```

BFS → shortest path

Q2 ⇒ Given Undirected graph find no. of connected Components. A component is said to be connected, if from every node we can visit all nodes, inside components

Ex: N=15, indicating 15 nodes In below graph How many connected components are there

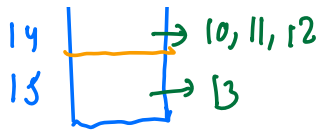


0	
1	→ 3, 2, 6
2	→ 1, 3
3	→ 1, 2, 4
4	→ 3, 6
5	→ 7, 9
6	→ 1, 4
7	→ 5, 9, 8
8	→ 9, 7
9	→ 5, 7, 8
10	→ 11, 14
11	→ 10, 12, 14
12	→ 11, 14
13	→ 15

1 → 15 → c = ~~1~~ ~~2~~ 4

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, ~~13~~, 14, 15

0	
1	T
2	T
3	T
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T
13	T
14	T
15	T




```
void dfs( list <int> g[], bool vis[], int s)
{
    if (vis[s] == True) return;

    vis[s] = true;
    for (int i=0; i<g[s].size(); i++)
    {
        int v = g[s][i];
        dfs(g, vis, v);
    }
}


int components (int N, E, u[], v[])
{
    list <int> g[N+1]      // To Do
    bool vis[N+1] = {F}
    int c=0
    for( i=1 ; i<=N; i++)
    {
        if( vis[i]==false)
        {
            dfs( g, vis, i)
            c= c+1;
```

```
        }
    }
    return c;
}
```

Q3 ⇒ **No. of Island**

Given a matrix of 1 & 0 find no of island are present ?

mat[][] → 1 : land
mat[][] → 0 : water

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

c=1

1	1	0
1	0	0
0	0	1

c=0

(0, 0) → right → (0, 1)
(0, 0) → down → (1, 0)
(0, 0) → left
(0, 0) → up

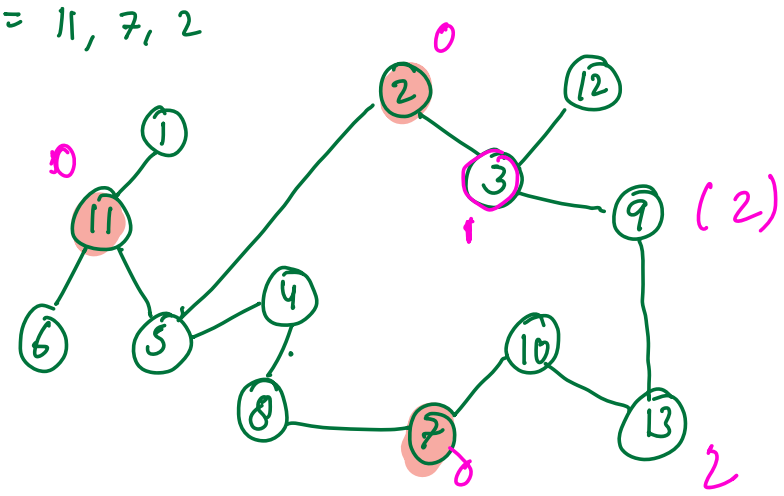
```
int x[] = {-1, +1, 0, 0}
int y[] = {0, 0, -1, +1}

void dfs(int mat[][], int i, j, int N; int M)
{
    if (i<0 || j<0 || i>=N || j>=M || mat[i][j]==0)
        return;
    mat[i][j]=0
    dfs(mat, i+1, j, N, M);
    dfs(mat, i-1, j, N, M);
    dfs(mat, i, j+1, N, M);
    dfs(mat, i, j-1, N, M);
}

for(k=0; k<4; k++)
    dfs(mat, i+x[k], j+y[k]
        N, M);

int island(mat[][], N, M)
{
    c=0
    for(i=0, i<N; i++)
    {
        for(j=0; j<M; j++)
```

```
        {
            if (mat[i][j] == 1)
            {
                c++;
                dfs(mat, i, j, N, M);
            }
        }
    }
    return c;
}
```

Q3 ⇒ **Multisource BFS**

Given N nodes & multiple source s_1, s_2 & s_3. Find length of shortest path for all nodes to any of the source nodes (s_1, s_2, s_3)

s = 11, 7, 2



~~11~~, ~~7~~, ~~2~~	~~1~~, ~~6~~, 5, 10, 8, 3	4
level 0	level 1	level 2

Q4 ⇒ **Rotten oranges**

mat[N][M] → 0 empty
→ 1 fresh oranges present
→ 2 rotten orange present

Every minute any fresh orange, adjacent to a rotten orange become rotten, find min time when all oranges become rotten. If not possible return -1

	0	1	2	3	4
0	1	0	1	2	1
1	1	1	1	1	1
2	0	2	0	1	0
3	0	1	1	1	1
4	1	1	1	2	0

(0,3) (2,1) (4,3)

3 hrs