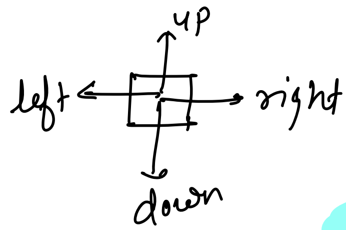
Number Of Islands

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

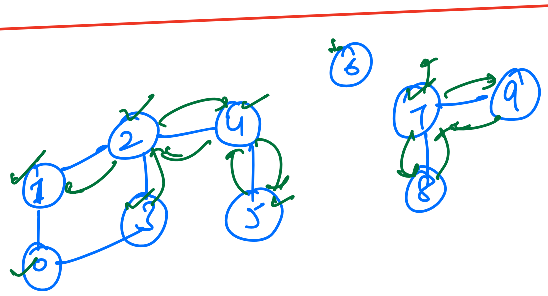
ans = 5.

1 → land
0 → water.



Every cell with 1 is node of a graph.

Number of Islands = No. of connected components.



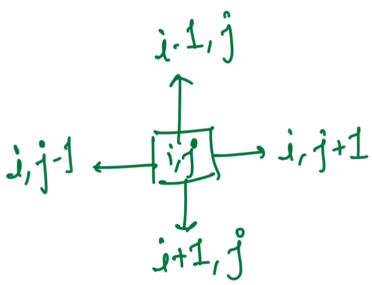
No. of components = No. of times we are calling dfs.

visited

0	1	2	3	4	5	6	7	8	9
t	t	t	t	t	t	t	t	t	t

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

```
boolean visited[N][M];
count = 0
for(i=0; i<N; i++){
    for(j=0; j<M; j++){
        if(arr[i][j] == 1 &&
           visited[i][j] == false){
            dfs(arr, i, j, visited)
            count++
        }
    }
}
return count;
```



$$\begin{array}{ccccc} & k=0 & k=1 & k=2 & k=3 \\ dx[\,] = \{ & -1, & 0, & +1, & 0 \} \\ dy[\,] = \{ & 0, & -1, & 0, & +1 \} \end{array}$$

```
for(k=0; k<4; k++){
    ni = i + dx[k]
    nj = j + dy[k]
}
```

$\begin{pmatrix} i-1 \\ j \end{pmatrix} \begin{pmatrix} i \\ j-1 \end{pmatrix} \begin{pmatrix} i+1 \\ j \end{pmatrix} \begin{pmatrix} i \\ j+1 \end{pmatrix}$

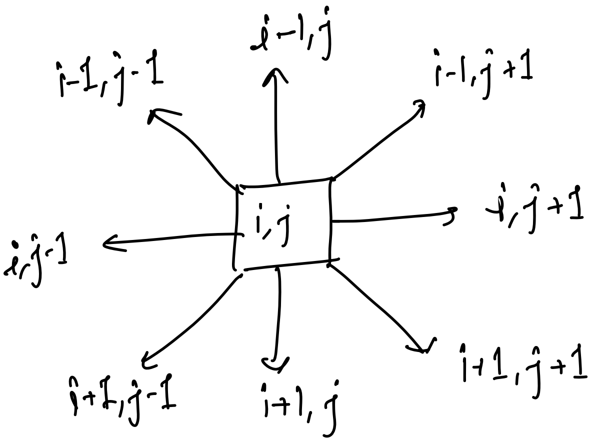
```
void dfs( arr[][], i, j, visited[][]){

    visited[i][j] = true;

    dx[] = { -1, 0, +1, 0}
    dy[] = { 0, -1, 0, +1 }

    for( k=0; k<4; k++){
        ni = i + dx[k]
        nj = j + dy[k]

        if( ni >=0 && ni<N && nj >=0 && nj<m
            && arr[ni][nj] == 1 && visited[ni][nj]==false){
                dfs(arr, ni, nj, visited)
        }
    }
}
```

[T.C → O(N*m)
S.C → O(N*m)]

#todo → bfs.

Small Variation

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



dx[8] → [-1 , -1 , 0 , +1 , +1 , +1 , 0 , -1]

dy[8] → [0 , +1 , +1 , +1 , 0 , -1 , -1 , -1]

$\boxed{\begin{array}{l} T.C \rightarrow O(N*m) \\ S.C \rightarrow O(N*m) \end{array}}$

```
for( k=0; k<8; k++) {
    ni = i + dx[k]
    nj = j + dy[k]
    if( ni >= 0 && ni < N && nj >= 0 && nj < m &&
        arr[ni][nj] == 1 && visited[ni][nj] == false){
            dfs( arr, ni, nj, visited)
    }
}
```

Rotten Oranges

Given a matrix [N*m] containing only 0's, 1's and 2's.

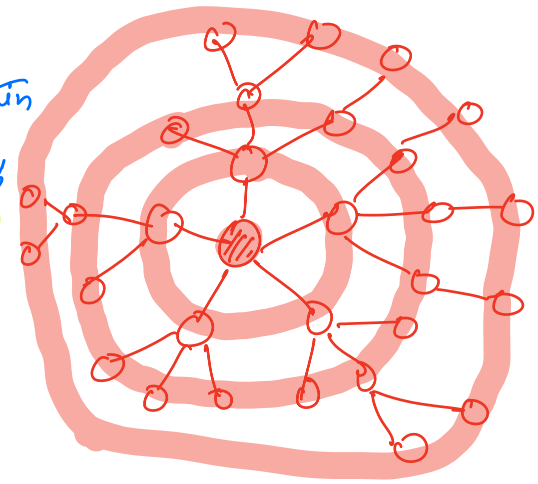
0 → empty cell, 1 → fresh orange, 2 → rotten orange.

Every minute, all the fresh oranges adjacent to rotten oranges become rotten.

In how many time will all oranges become rotten?
If it is not possible, return -1.

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

time = ~~1 min~~ ~~2~~ ~~3~~ ~~4~~ 5



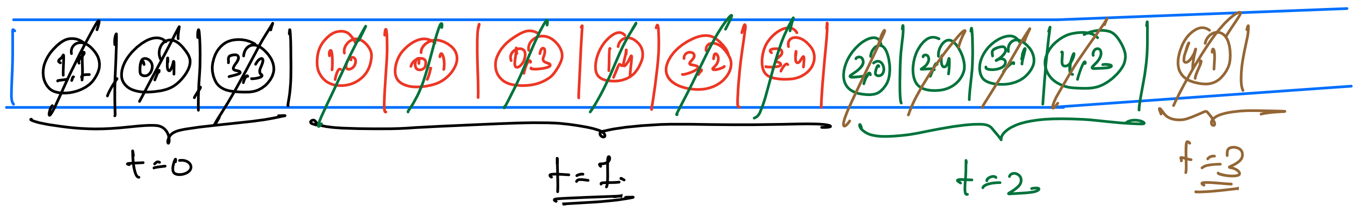
[nodes → all cells with oranges.
edges → connection b/w oranges]

↓

B.F.S.

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

Multi-sourced B.F.S



pair<i,j>

pseudo code.

```
Queue <pair> q;

for( i=0; i<N; i++){
    for( j=0; j<M; j++){
        if (arr[i][j] == 2){
            q.enqueue( i,j);
        }
    }
}
```

enquening all sources.

$T.C \rightarrow O(N*M)$
$S.C \rightarrow O(N*M)$

```
time = 0
while( q.isEmpty() == false) {
    x = q.size();
    for( i=1; i<=x ; i++){
        rp = q.dequeue()
        // explore all 4 neighbours
        // if any fresh orang is found as neighbour
        // then make it rotten & enqueue its coordinate
    }
    time++
}
return ?
```

// Please do check if any fresh orange is left or not.

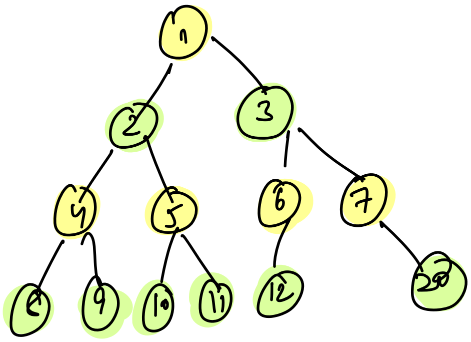
Graph Coloring.

Francis Guthrie → [1852]

Minimum no of colors required to
→ Color all the nodes of a graph, such that no two adjacent nodes are of same color

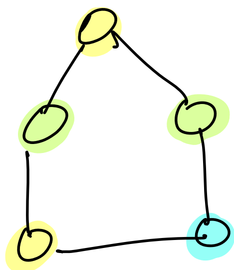
⟹ Chromatic Number

① Tree.

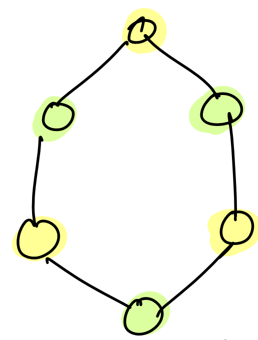


Chromatic no → 2.

②. Cycle Graph [whole graph is cycle]



C.N=3.



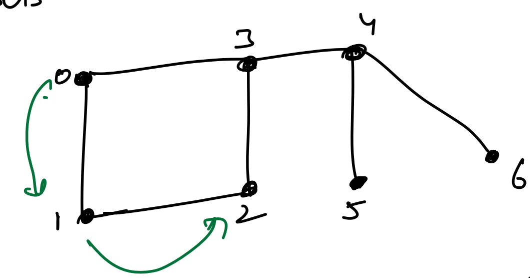
C.N=2.

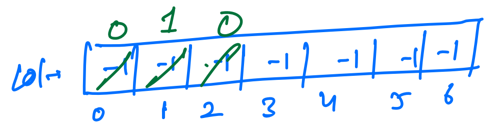
In general, [C.N = 2 + (N%2)]

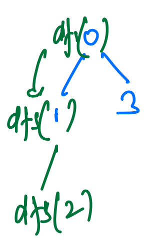
③ Bi-partite Graph

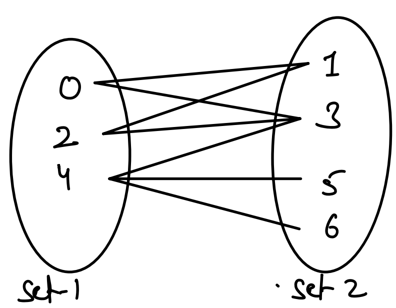
* Any graph having C.N = 2. Eg → trees, even length cycle graph etc.

** A graph is called bi-partite, if we can divide all the nodes into two sets such that all the edges are across the sets.









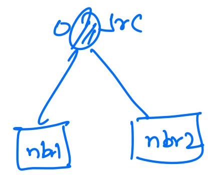
Bi-partite

Q: Check if given graph is bi-partite or not?

$\forall i$, col[i] = -1   //No color

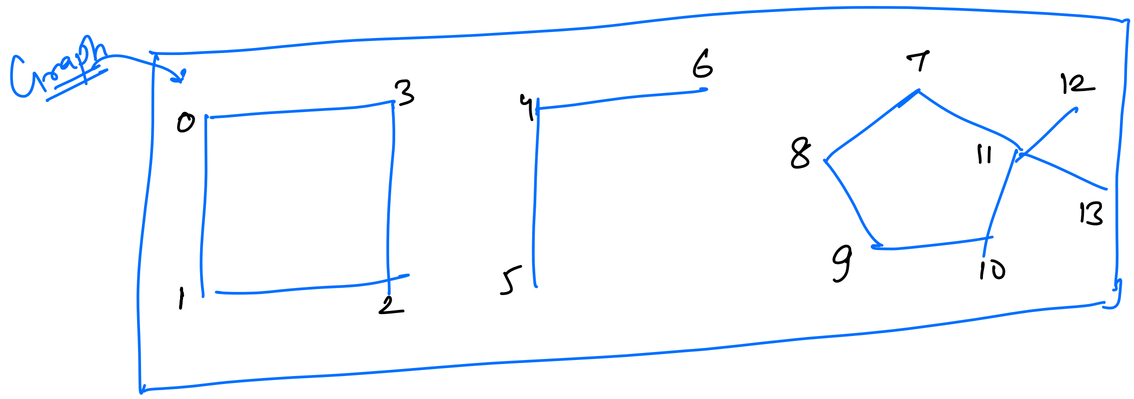
col[src] = 0

col[i] → (= 0) set-1
col[i] → (= 1) set-2




```
boolean dfs ( int src) {
{
    for( int nbr : A[src]) {
    {
        if( col(nbr] == -1) {
        {
            // color nbr with diff. color than the src
            col[nbr] = 1 - col[src]
            if( dfs(nbr) == false){
            {
                return false
            }
        }
        else if (col[nbr] == col[src]){
        {
            return false
        }
    }
    return true
}
```

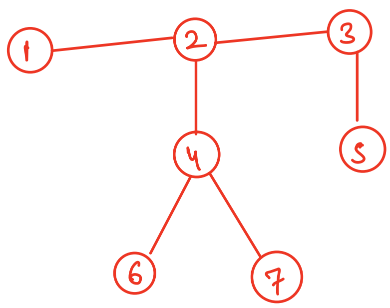
$\begin{bmatrix} T.C \to O(N+E) \\ S.C \to O(N) \end{bmatrix}$



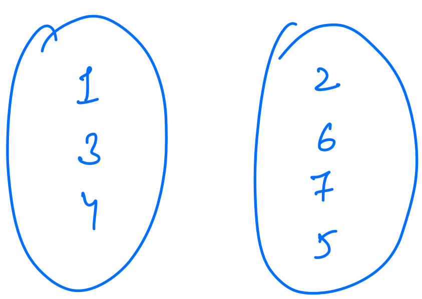
* If any component of graph is non-bipartite then the whole graph will be non-bipartite.

```
for( i=0 ; i< n ; i++){
        if (col[i] == -1){
              if ( dfs( i) == false){
                       return false.
              }
        }
}
return true;
```

Q) A country consists of N cities connected by (N-1) roads. King of that country wants to construct maximum roads such that cities can be divided into two sets and there is no road between cities in the same set. Find maximum no. of new roads that can be created?



[7 - cities
6 - roads]



set-1 color [0]   set-2 color [1]

Max. no. of roads = n. of maximum edges we can have across S1 & S2.

= no. of nodes in set-1 * no. of nodes in set-2

[Ans = Max no. of roads - (N-1)]