GRAPH (NOTES BY RAMAN JOT SINGH)

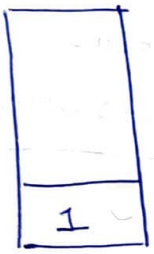
(1) BFS traversal

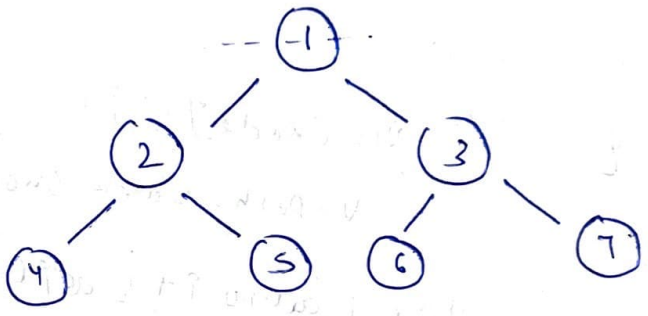
Koi bhi graph का Bfs traversal chahiye तो hme queue use karna होगा or sabse pehla उसमे intial value from which we want to start BFI store karni padegi. for any traversal we will keep a vissited array.

Bfs concept तब तक queue मे से element निकालो जब तक empty ना हो jay. जिस node को निकालो उसके adjacent nodes अगर visited नहीं है तो queue मे Dal Do.

← visited array intialised with zero.

q.push(1) (Starting node put in queue:





जब 1 को queue मे store करा then make visited array corresponding one

v[0] = 1

TC → O(N) + O(2E)
SC → O(N)

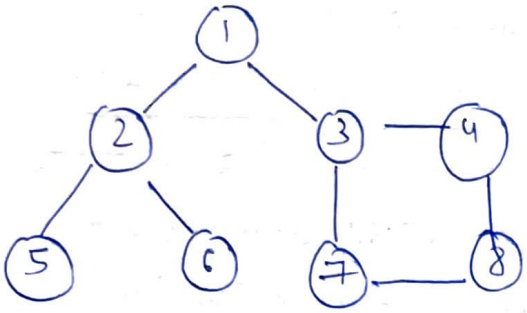
```
while (!q.empty())
{   int node = q.front();
    q.pop();
    bfs.push(node);
    for (auto it : adj[node])
    {   if (!v[it])
        {   v[it] = 1;
            q.push(it);
        }
    }
```

← queue मे जो भी value हो उसे final bfs के vector मे store करो then उस value q से remove करो or जो value store करी है उसके adjacent value जो visite na ho to queue मे push करो)

notes By:
Ramanjot- Singh

(2) Dfs traversal

Recursion: algo that goes into depth and returns back



एक visited array बनाया जिस index start करो BFS उसे उसमे 1 कर दोगा। then push list में at node जो भी nodes adjacent node to इसे उनपे Dfs call करे।

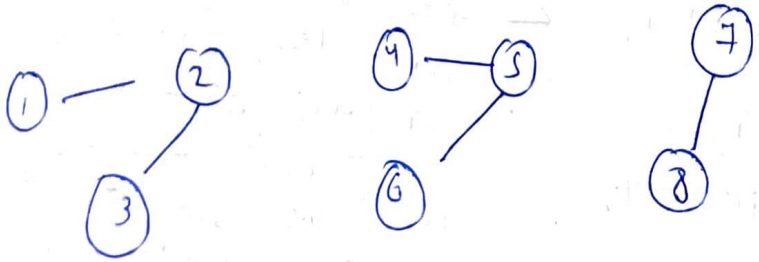
```
dfs(node)
{
    vis[node] = 1
    v.push_back(node);
    for(auto it : adj[node])
    {
        if(!vis[it])
        { dfs(it); }
    }
}
```

undirected

TC → O(N) + O(2×E)
O(N) + O(E)
directed

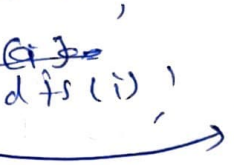
SC = O(N) + O(N) + O(N) ≈ O(N)

(3) Number of provinces.



The approach is to use visited array initialised with zero if a node is not visited bfs or dfs is called for it now when starting from a particular node say 1 dfs is called it will visit all the nodes. 1, 2 and 3 and mark the visited. ~~ab store~~ nahi ~~kr~~ rhe vah alag baat hai

```
for (i=0 ; i<=V ; i++)
{ if (V[i] == 0)
    { cout ++;
      bfs(i) || dfs(i);
    }
}
```



```
for (int i = 0 ; i<V ; i++)
{ for (int j = 0 ; i<V ; i++)
  { if (adj[i][j] == 1 && i != j)
    { adjLs[i].push_back(j);
      adjLs(j).push_back(i);
    }
  }
}
```

** Generic code to change Adjancy Matrix to list

SC → O(N) + O(N)
- O(N): visited array
- O(N): recursion stack space

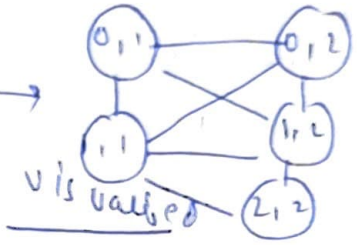
Partially it will run dfs for all the nodes starting from 1, 4 and 7 so now

TC → O(N) the inner loop will run for the journey
run, for O(V + 2E) times.
overall O(N) + O(V + 2E)

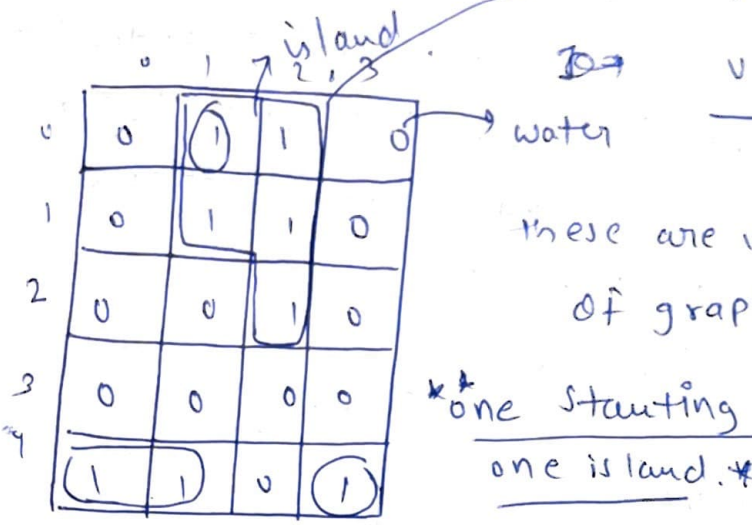
notes By!
Ramanjot Singh

(9)

Number of islands



Like this

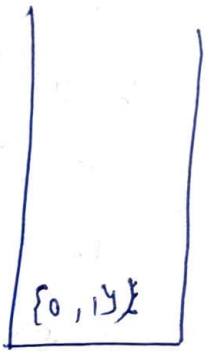


→ is visualised

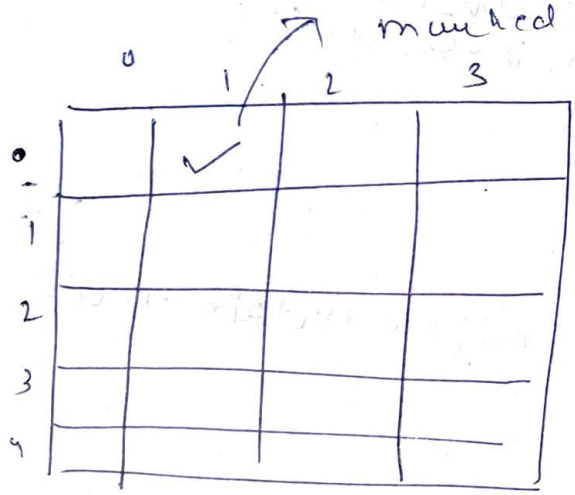
these are visualised as nodes of graph

one starting node will represent one island.

Starting {0, 1}

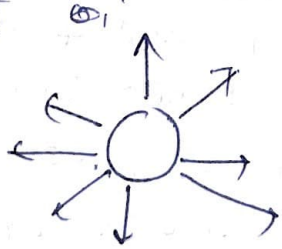


To traverse nodes adjacent we would use BFS technique a put them in queue. now we mark the vertex as visited so for that we will use 2D array as vertex no are in pair.



replica 2D array

now we need to remove {0,1} from queue and put all neighbors of {0,1} in queue. the neighbour so of {0,1} all these directions.



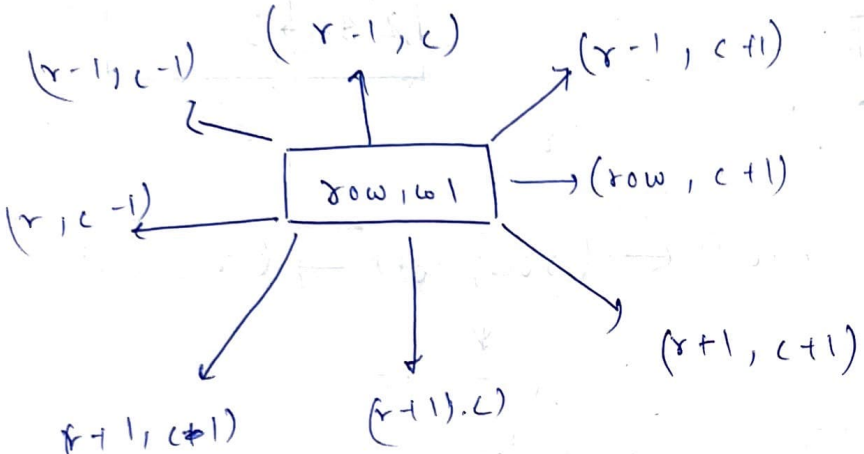
NOTES BY:
Ramanjot Singh

now repeat for rest

Pushed in Queue in matched visited

1,1
1,2
0,2

* Traverse the whole graph if the element is land and not visited call bfs for it and do count++.



Instead of writing we can run all we can run 2 for

```
for ( drow → ( -1, 1 ))
    ( for (dcol → (-1, 1 ))"
        { int nrow = row + drow
          int ncol = row + dcol )
```

SC → O(N²) + O(N²)

for visited array — it all of these are connected so queue may have

TC ≈ N²

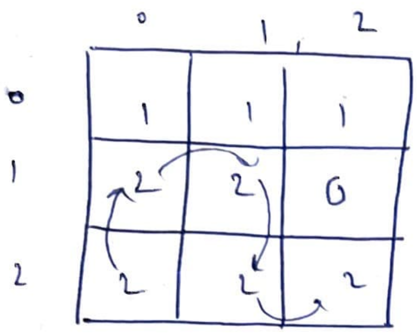
O(N²) for traversing the 2D array

for each place place loop checks 9 location O(N²) + O(9 × N²) ≈ O(N²)

(5) Flood fill Algo

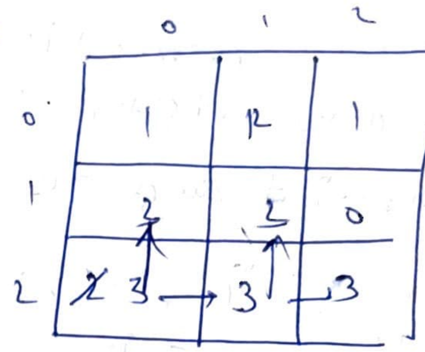
Notes By!
Ramanjot Singh

We are given ~~index~~ index of a place in grid, new color we need to do. We can only move in ↑ ← → ↓ direction and ~~color~~ change no having same no ~~as~~ selected place.



sr = 2 sc = 0

new col => 3



DFS(2, 0)
↑↓
DF(1, 0) → DFS(1, 1) × (already visited)
↑↓
DFS(1, 1)
↑↓
DFS(2, 1)
↑↓
DFS(2, 2)

(r-1, c)
↑
(r, c-1) ← (row, col) → (r, c+1)
↓
(r+1, c)

↓

now we will store row and col correspondingly and add to see positi...

row [-1, 0, 1, 0]
col [0, 1, 0, -1]

array store.

SC → O(N × M) + O(N × M)
↑ extra space ↑ stack space

TC → O(N × M) + O(4 × N × M) ≈ O(N × M)

(6) Rotten oranges

As we want to rotten the oranges symultaneously in minimum time ∴ bfs will be best algo.

	0	1	2
0	0	1	2
1	0	1	1
2	2	1	1

we will put those oranges which are intially rotten

Queue at time zero these where rotten.

visited 2D matrix

	0	1	2
0		2	2
1		2	2
2	2	2	2

t mai intial store कर लिया time
t → t+1
or max time bhi maintain rakh liya फिर
had मे return कर दिया

$tm = max(tm, t)$

** last मे ye bhi check करना की अगर grid मे voh 1 था पर visited मे 2 नहीं तो return -1 कर देना की सब नहीं change hue.

SC → O(NM) + O(NM) ≃ O(NM)

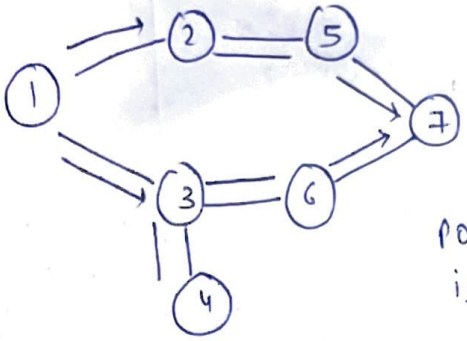
↑ visited array ↓ queue

TC → O(NM) + O(4×NM) ≃ O(NM)

↑ traversing array ↑ हर एक के लिए 4 बार call करना

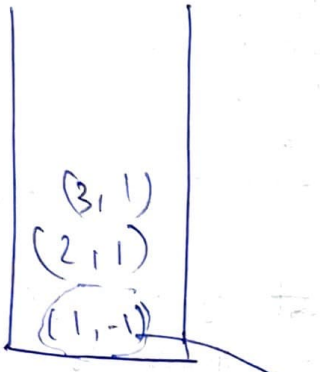
(7) Detect a cycle using (undirected graph using BFS.

NOTES BY!
Ramanjot Singh

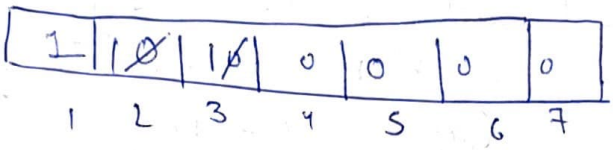


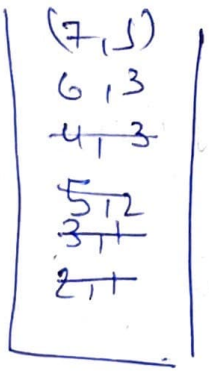
Intition is we started in two different direction and if you are colliding in some position then definately there is something as cycle..

** insert the node in queue with the parent and use a visited array.



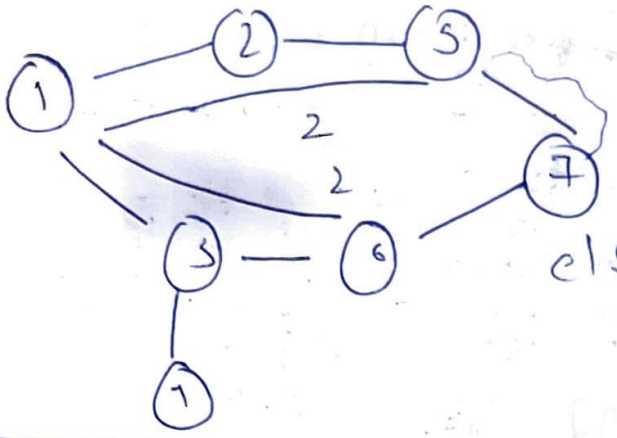
remove this and see adjacency list and put the adjacent nodes in queue is the with parent





mark them as visited as soon as you put in queue.

at 6 node we have (3,7) in adjacency list now 3 is parent for 6 6 came from 3 so it is visited but when we see 7 it is also marked visited as we are traversing in equal direction. it can only happen if some one touched it before now if 6 go and touch it. It means its a cycle - Some one at level 1 touched it before touched it.



** if someone is visited and it did not come from it

else if (parent != adjacent node)
{ return true; }

```
for including component standard way
for (i=1 ; i<=N ; i++)
  { if (!vis[i])
      { if (detect cycle(i)==true)
          { return true; }
      }
  return false
```

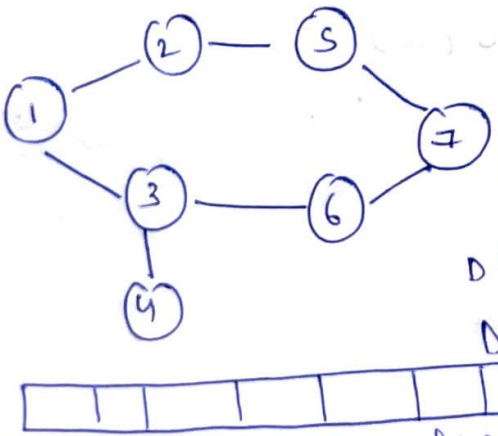
for ~~allotte~~ add every node we are traversing all its adjacent nodes. sum of all its adjacent nodes is degree (2E)

(we are not calling bfs for every for loop)

TC → O(N+2E) + O(N) every for loop

SC → O(N) + O(N) ≈ O(N)

(8) Detect a cycle in an undirected Graph using DFS.



DFS(1,-1)
DFS(2,1)
DFS(5,2)
DFS(7,5)
DFS(6,7)
DFS(5,6)
DFS(4,3)

1 already visited but it's a parent so we will not set cycle detected. (now there is which not parent but visited.

adj list

1 → {2,3}
2 → {1,5}
3 → {4,7,6}
4 → {3}
5 → {2,7}
6 → {3,7}
7 → {5,6}

if I am getting true no need to make dfs call further

```
dfs(node, parent)
{
    vis[node] = 1
    for(auto it : adj[node])
    { if (vis[it] == 0)
        {if(dfs(it, node) == true)
            return true;
        }
    else if (it != parent)
        return true;
    }
return false;
}
```

अगर यहां true return कर रहा है तो आगे call करने की जरूरत नहीं है

यहां पर true return कर रहा है (node का adjacent node) it's visited but parent नहीं matlab its acyclic.

DFS call we did not jota cycle)

DFS की call को हटा दें तो bs ये if के साथ we are checking a condition.

**also include the standard code of calling connected components.

$S(\rightarrow O(N) + O(N) \approx O(N)$

$T(\rightarrow O(N + 2q) + O(N)$