Graph → II

Till Now → Graph
↳ Terms
↳ BFS
↳ DFS
↳ Adj Matrix
↳ Adj List

Q-1/w → 1 → find No of Disconnected Component in graph

or

find No of Island
↳ Done

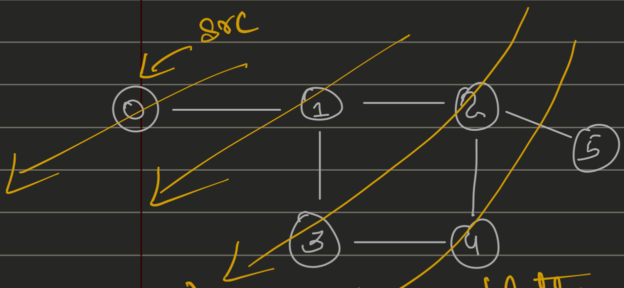
2 → Rotten Tomatoes or Rotten Orange

Today Goal → Cycle Detection

1. → Undirected graph → BFS
   ↳ DFS
2. → Directed → BFS
   ↳ DFS

4 Cases

→ Undirected Graph



adj list

0 : {1}
1 : {0, 2, 3}
2 : {1, 4, 5}
3 : {1, 4}
4 : {2, 3}
5 : {2}

→ q.push(src)
vis[0] = T
→ parent[0] = -1

Queue

~~0~~	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~

cycle complete

Parent

0 : -1
1 : 0
2 : 1
3 : 1
4 : 2
5 : 2

diff

Visited

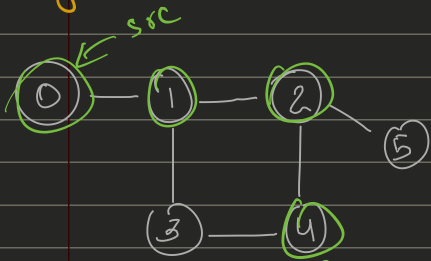
0 → ~~F~~ T
1 → ~~F~~ T
2 → ~~F~~ T
3 → ~~F~~ T
4 → ~~F~~ T
5 → ~~F~~ T

→ kisi bhi node pe uske parent ke alwa kisi aur se bhi aa pe rhe Toh ya pe cycle present hai.

2 : {1, 4, 5}   3 : {1, 4}

$$\text{if } (\text{vis}[c] == \text{true} \ \&\& \ \text{child} \ != \text{parent}[\text{front}])$$

cycle present



Adj List

0:	{1}
1:	{0, 2, 3}
2:	{1, 4, 5}
3:	{1, 4}
4:	{2, 3}
5:	{2}

Queue

→ q.push(src)
vis[0] = T
parent = -1

~~0~~	~~1~~	~~2~~	~~3~~	~~4~~	5
0	1	2	3		

Parent

0:	-1
1:	0
2:	1
3:	1
4:	2
5:	2

Visited

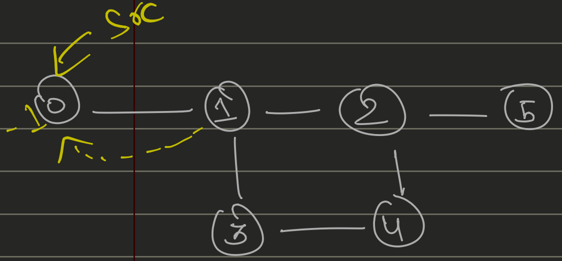
0 →	~~F~~ T
1 →	~~F~~ T
2 →	~~F~~ T
3 →	~~F~~ T
4 →	~~F~~ T
5 →	~~F~~ T

1 ← parent[1]

node → cycle present

Impt

$$\text{Cond}^n \rightarrow \text{vis}[\text{nbr}] == \text{true} \ \&\& \ \text{nbr} \ != \text{parent}[\text{frontnode}]$$



Adj List

nbr

0:	{1}
1:	{0, 2, 3}
2:	{1, 4, 5}
3:	{1, 4}
4:	{2, 3}
5:	{2}

q.push(src)
vis[src] = T

nbr != parent[frontNode]
0 != parent[1]
0 != 0 → F
No cycle

Queue

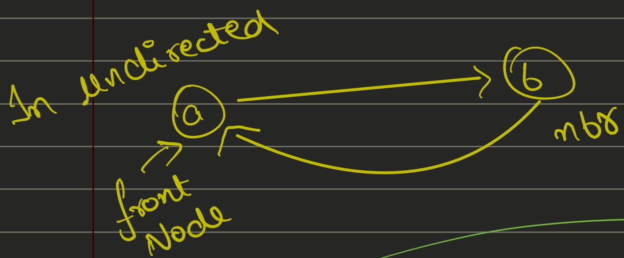
~~0~~	~~1~~	~~2~~	3	4	5

Parent

0 →	-1
1 →	0
2 →	1
3 →	1
4 →	2
5 →	2

Visited

0 →	~~F~~ T
1 →	~~F~~ T
2 →	~~F~~ T
3 →	~~F~~ T
4 →	~~F~~ T
5 →	~~F~~ T

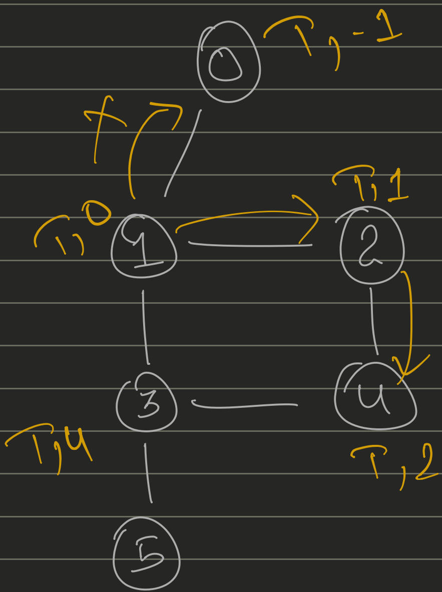


vis[4] = true → True &&
4 != parent[3]
4 != 1 → True

→ cycle Present

DFS →

Ignore Parent vala Case



dfs(0)

↓

dfs(1)

~~dfs(0)~~ dfs(2)

dfs(4)

↓

dfs(3)

dfs(1) dfs(5)

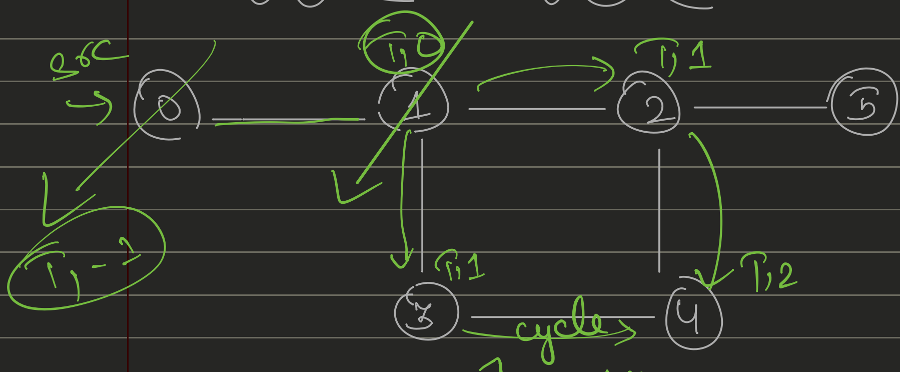
Already True ← dfs(1)

next node != parent

↓

Cycle Prent

Another BFS Method



1 -> {0, 2, 3}

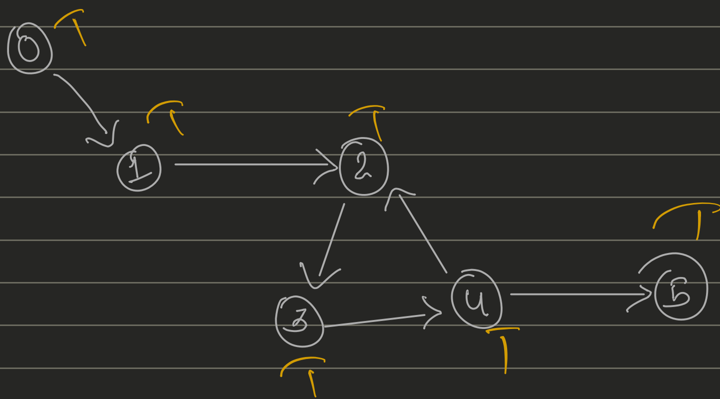
(S-1) parent[front Node] == nbr
↓
Continue

(S-2) -> if (!vis[nbr])
↳ q.push
↳ vis

(S-3) -> else if (vis[nbr] == True)
↳ cycle present

Directed Graph cycle Detection

(1) → DFS

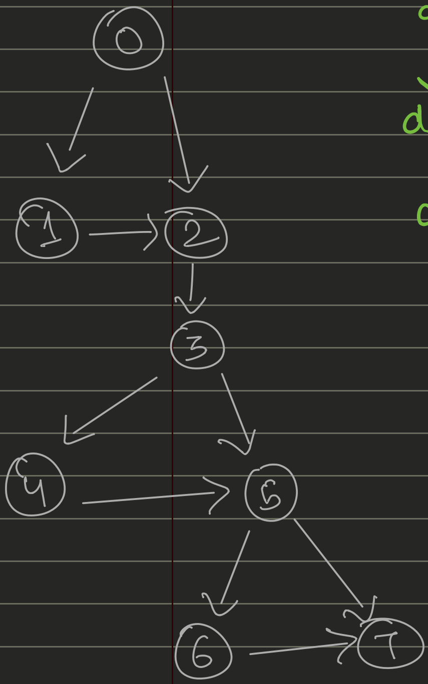


dfs(0)
↓
dfs(1)
↓
dfs(2)
↓
dfs(3)
↓
dfs(4) → dfs(2) → same No. ki call Dubara Toh cycle Present
↓↑
dfs(5)

dfs(0) → T
dfs(1) → T
dfs(2) → T
dfs(3) → T

dfs(4) → T
dfs(5) → ~~T~~F

Use Backtracking



dfs(0)

↓

dfs(1)

↓

dfs(2)

↓

dfs(3)

↓

dfs(4)

↓

dfs(5)

↓

dfs(6)

↓

dfs(7) → dfs(5)

Already T

cycle Present

visited

0 → ~~F~~ T	7 → F
1 → ~~F~~ T	
2 → ~~F~~ T	
3 → ~~F~~ T	
4 → ~~F~~ T	
5 → ~~F~~ T	
6 → ~~F~~ T	

dfs Track

dfs(0) → ~~F~~ T	dfs(6) → ~~F~~ T
dfs(1) → ~~F~~ T	dfs(7) → ~~F~~ T
dfs(2) → ~~F~~ T	
dfs(3) → ~~F~~ T	
dfs(4) → ~~F~~ T	