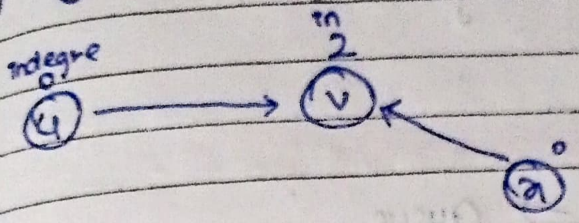
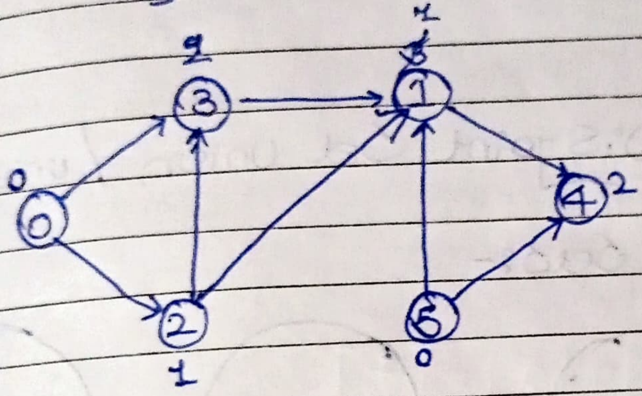
Graph

Topological Sorting using BFS (KAHN'S Algorithm)



Topo sort
→ u, w, v ✓
→ w, u, v ✓

→ जिसका indegree 0 हुआ, पहले उसको लिखेंगे।



**
Jis-Jiska indeg 0 hai queue mai daal denge

Queue

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

⟹ 0, 5, 2, 3, 1, 4

→ 0 से 2, 3 visit kar sakte hu
2 ka indegree 1 से 0 ho gya or 3 ka 1   fir ~~but~~ 2 ko queue
mai daal denge indegree 0 ho gya kyuki

→ 5 से 1, 4 visit kar sakte hai   1 ki indegree 2 ho gyi
or 4 ki 1

→ 2 से 3 or 1 visit kar sakte 3 ki indegree 1 ho gyi
thi vo ab 0 ho gyi queue mai daal denge or 1 ki 1
ho gyi

→ 3 से only 1 ja sakte 1 ki indegree 1 thi vo ab 0 ho
gyi queue mai daal diya

→ similarly from 1 to 4

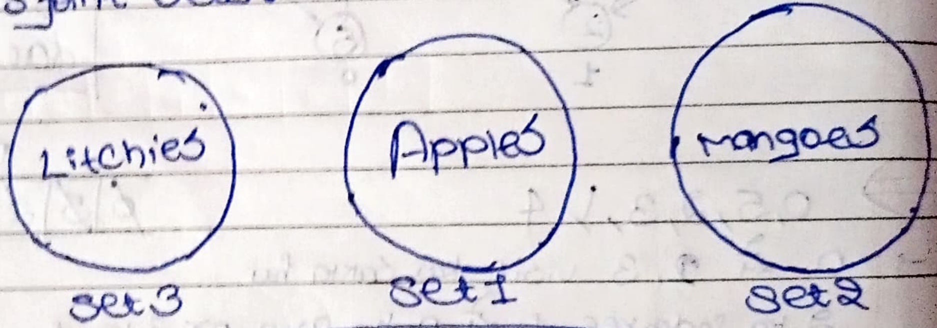
```
1) for (int u=0; u<N ;u++){
       for (int & v: adj[u]){
           indegree[u]++;
       }
   }
```

2) indegree == 0 → queue

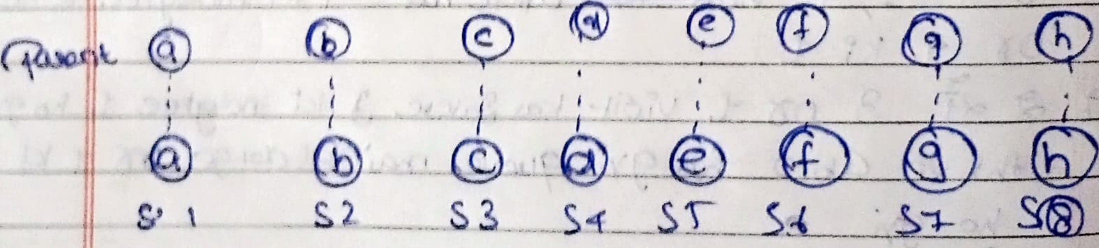
(3) simple BFS

→ DSU (Disjoint Set Union / union find)

→ Disjoint Sets:-



$\boxed{S_1 \cap S_2 \cap S_3 = \text{NULL}}$



Rule → 1) combine Two given sets

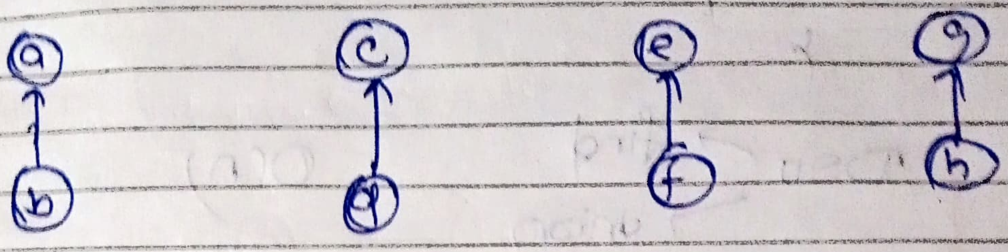
Element

2) Tell if two members (b, g) Belong to same set or not

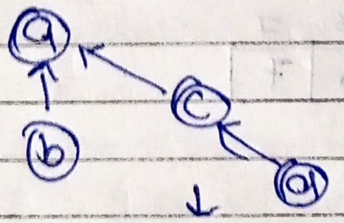
Element	a	b	c	d	e	f	g	h	
Parent	a	b	c	d	e	f	g	h	initiap

Step 1

union (a,b) union (c,a) union(e,f) union (g,h)

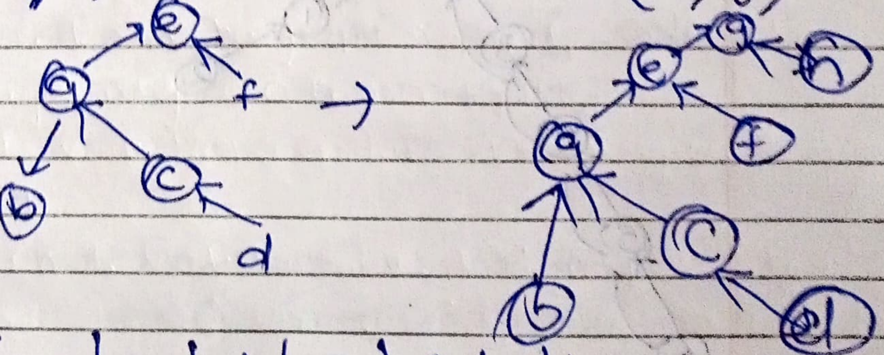


Step ② union (a,c)



union(a,e)

union (e,g)



Element	a	b	c	d	e	f	g	h
Parent	e	a	a	c	g	e	g	g

```
int find (int i, vector <int> & Parent) {
    if (i == parent [i]) return i;
    return find (parent [i], parent);
}
```

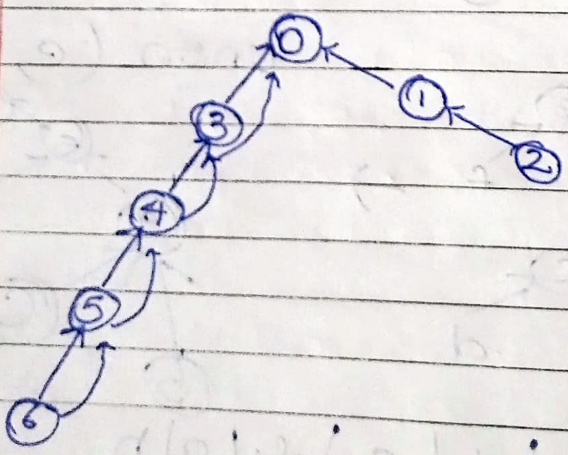
```
void union (int x, int y, vector <int> & Parent
    int x_Parent = find (x, Parent);
    int y_Parent = find (y, Parent);

    if( x_Parent != y_Parent){
        Parent [x_Parent] = y_Parent;
    }
}
```

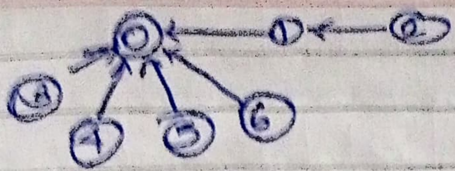
DSU → find / → union   O(n)

Path Compression

0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7



→ first time 6 Pr gaya find Parent call kiya vo 5 pr jayega call karega than 4, 3, 0, 0 pr aate hi bolega ki ha mai hi leader hu fir jab wapis aata jayega sab ko individual Bata denge Suno tumhara jo leader hai vo 0 hai



So now code for Path Compression is

```
find (int i, Parent){
    if(Parent[i]==i){
        return i;
    return Parent[i]= find (Parent[i], Parent);
```

for Rank

```
void union(int x, int y, vector<int>& Parent, vector<int>
            & rank){
    int x_Parent = find (x, Parent);
    int y_Parent = find (y, Parent);
    if(x_Parent == y_Parent) return;
    if ( rank [x_Parent] > rank [y_Parent]){
            Parent [y_Parent]= x_Parent;
    }
    else if( rank [x_Parent] < rank [y_Parent]){
            Paent [x_Paent] = y_Parent;
    }
    else{
        Paunt [x_Paant] = y_Parent;
          rank [y_Parent] += 1;
      }
}
```