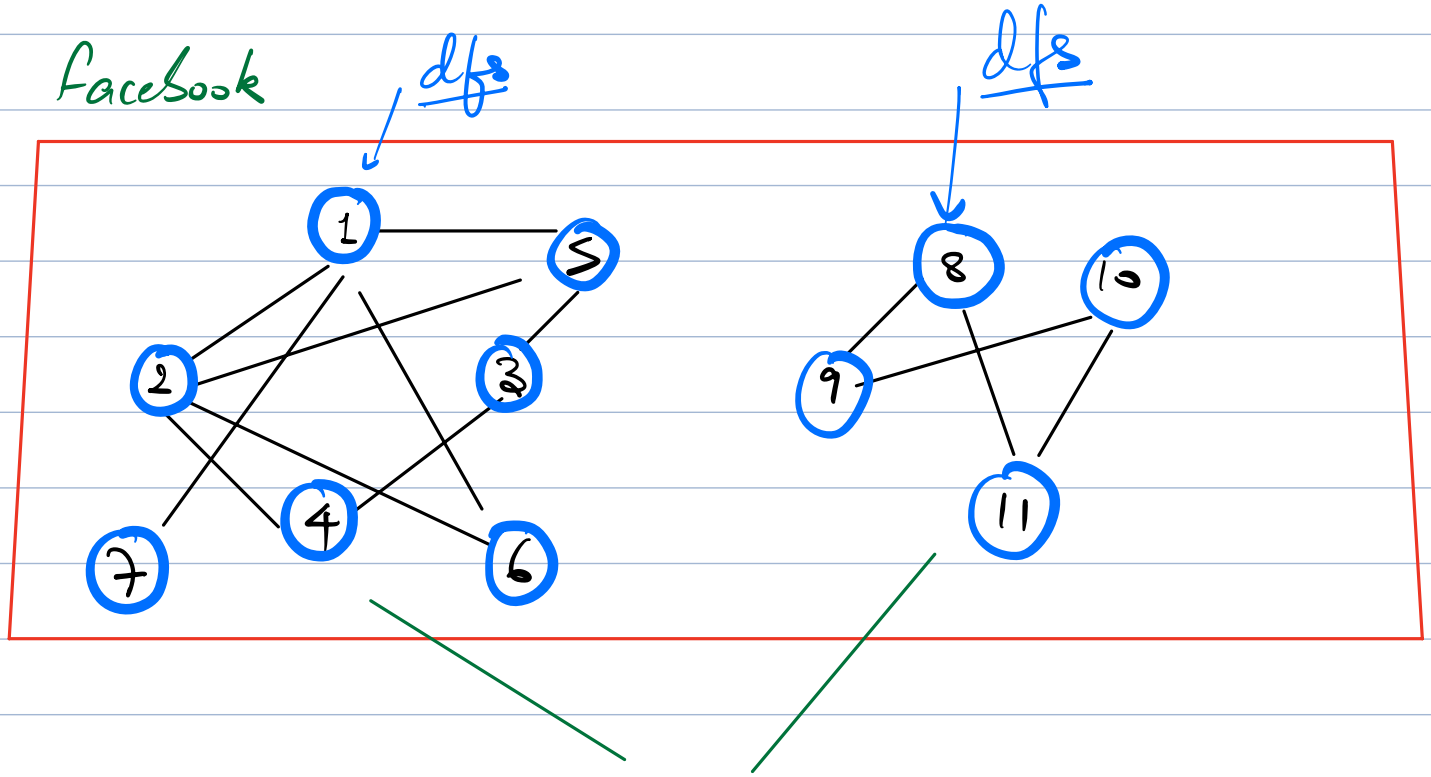
Connected Components

PS Session → DP, Graphs

Tomorrow 4PM

Facebook



Q) Find the no. of CC present in a graph with V nodes numbered from 1 to N & E edges. (Undirected)

Code

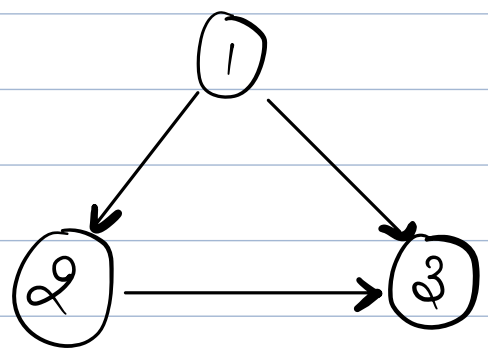
1) Create adjacency list.

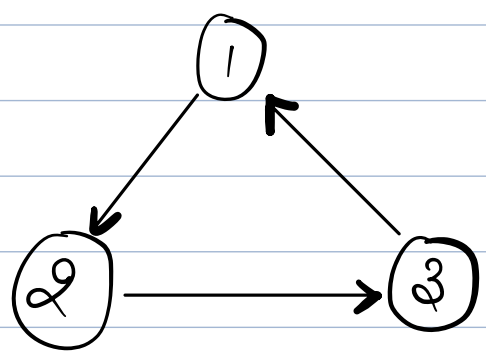
visited [V+1] = {false}

```
count = 0;
for (i = 1; i <= V; i++) {
        if (visited[i] == false) {
                count++;
                dfs(i);
        }
}
```

$T.C. = O(V + E)$

Strongly Connected Components.



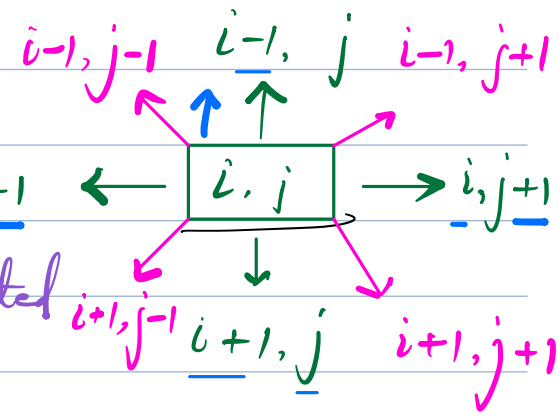


Strongly Connected Components

No. 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

$N \times m$



1 → Islands
0 → Water

i /= 0, 1 ⇒ Visited
↳ 2

Count the no. of islands given.

```
count = 0;
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        if (M[i][j] == 1) {
            count++;
            dfs(i, j);
        }
    }
}
```

Row & Col Modifiers.

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

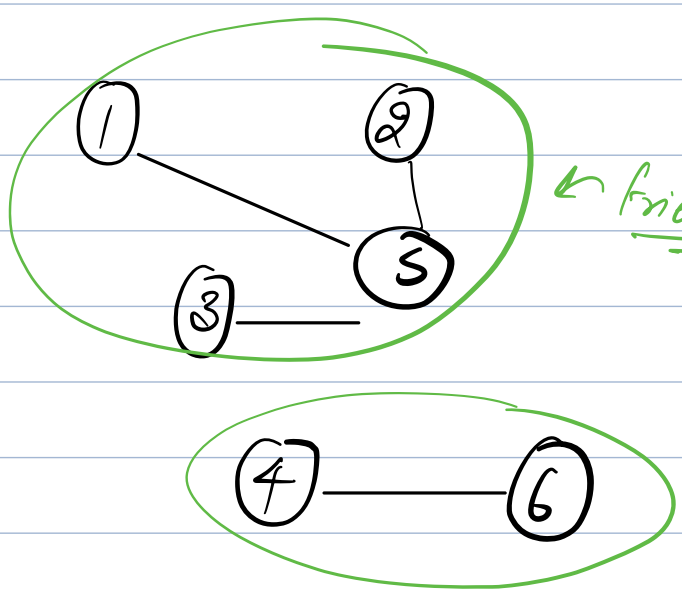
```
void dfs (i, j) {
    M[i][j] = 2;        // Mark as Visited.

    for (K=0; K < row.size() ; K ++) {

        // row_index = i + row[K];
        // col_index = j + col[K];

        if ( check (row_index, col_index) == True) {

            if (M[row_index][col_index] == 1

                dfs (row_index, col_index);

            }

        }

    }

}
```

```
bool check (i, j) {
    if (i<0 || j<0 || i>=N || j>=N) {
        return false;
    }
    return True;
}
```

$$T.C = O(N^2)$$

Audible → Prime Users (Free Audible Subscription)

Bonus → Refer & Join



← friends

Count the no. of such groups

$2+(-1),\ 3+0$

$2+0,\ 3+(-1) \leftarrow 2, 3 \rightarrow 2+0,\ 3+1$

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

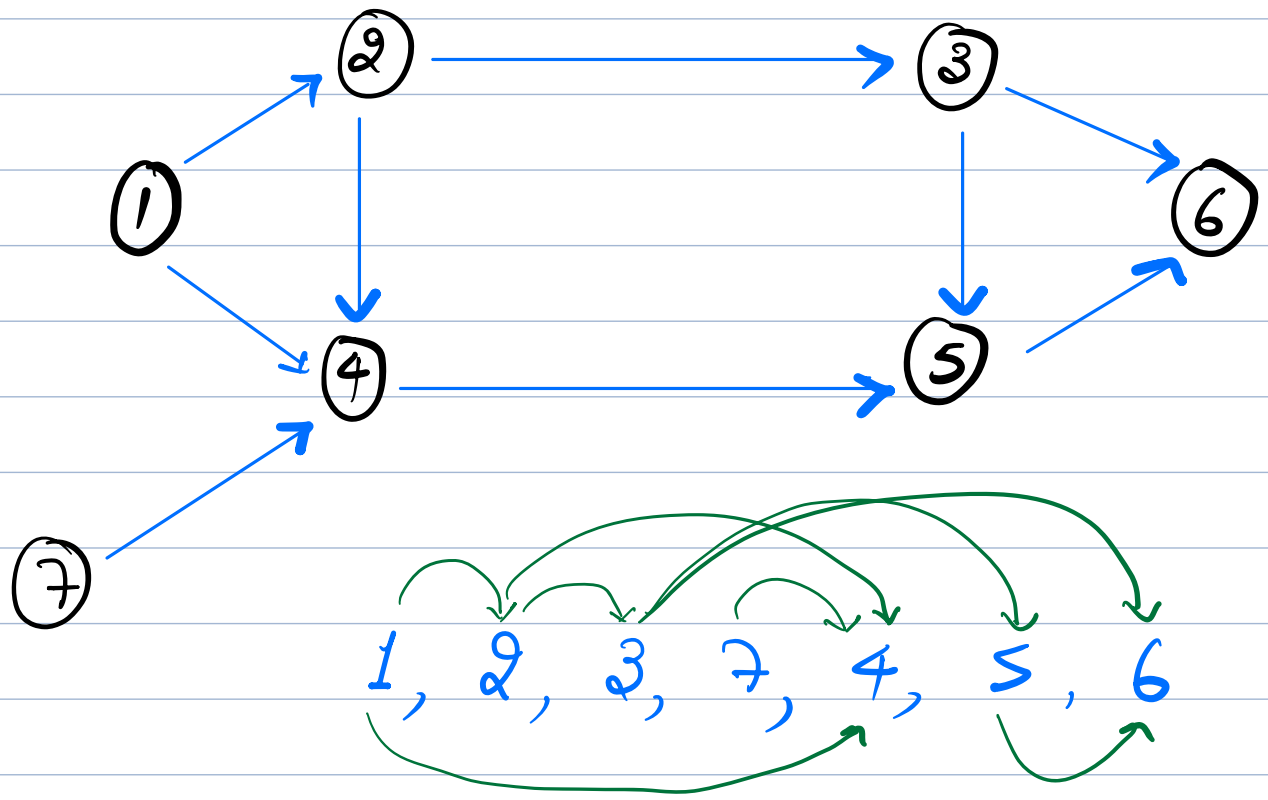
2+1, 3+0

3	1	1	0	~~0~~	0
4	1	0	1	1	1

Topological Sort

Only for Directed Acyclic Graph.

Linear ordering of vertices (Nodes) such that for every directed edge (u, v), vertex u comes before vertex v.



1, 7, 2, 4, 3, 5, 6

Project → Packages 1
- Packages 2 → Design System
- Packages 3 (Utility)
- Packages 4 (Wrapper → DB)

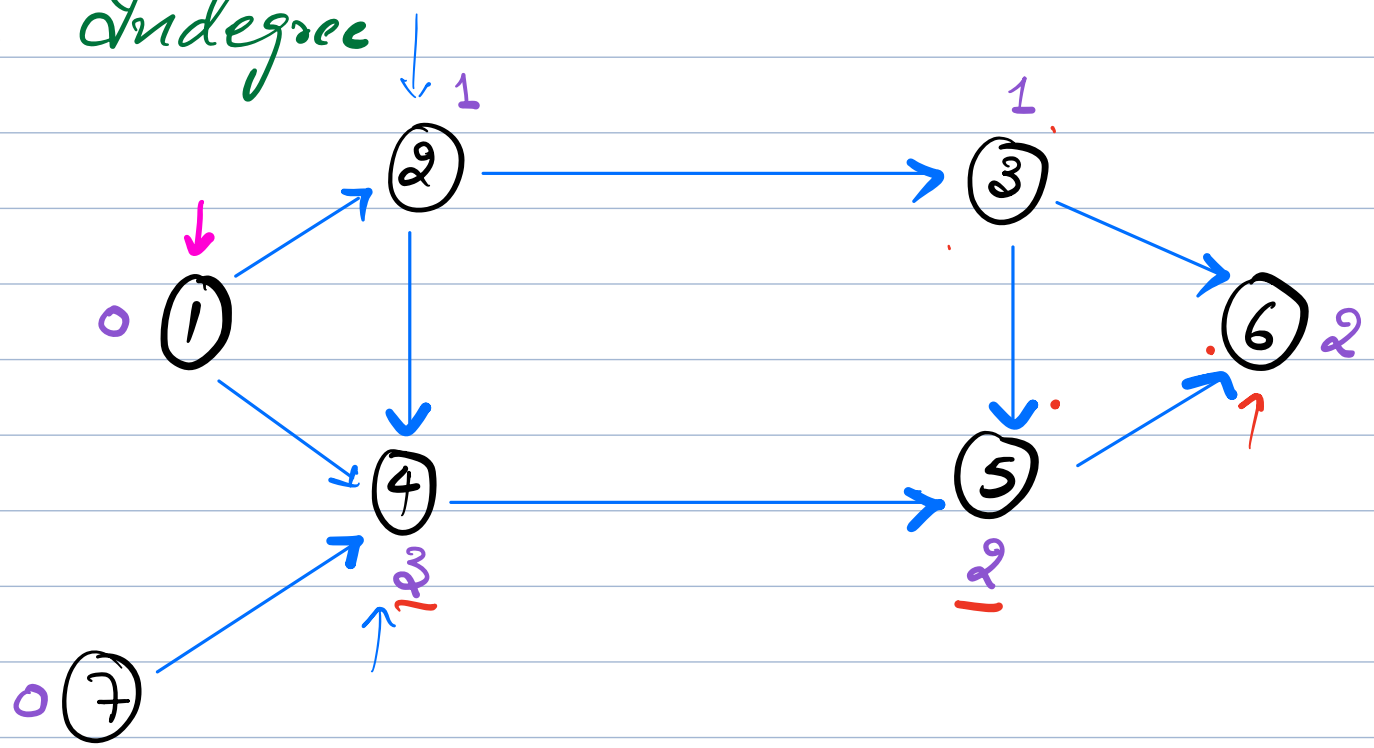
⋮

Error ⇒ Circular dependency ✗ Not Cyclic

Order in which packages needs to be built ⇒

Find Topological Sorted Order

1) Indegree



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

1, 2, 3, 7, 4, 5, 6

~~1~~
~~7~~
~~2~~
~~3~~
~~4~~
~~5~~
~~6~~

→ HashSet

```
indegree[N+1] = {0}

for (i = 1; i ≤ N; i++) {
    for (all nodes u connected to i) {
        indegree[u]++;
    }
}
```

(Adjacency List)

V+E

```
HashSet<int> zerodegree;

for (i = 1; i ≤ N; i++) {
```

```
        if (indegree[i] == 0) {
            zerodegrees.add(i);
        }
    }

while ( zerodegrees.size() != 0) {
    int node = zerodegrees.getElement();
    print (node);
    for (all u connected to node) {
        indegree[u]--;
        if (indegree[u] == 0) {
            zerodegrees.add(u);
        }
    }
}
```

$V + E$

$T.C. = O(V+E)$

H.W. Outdegree ??

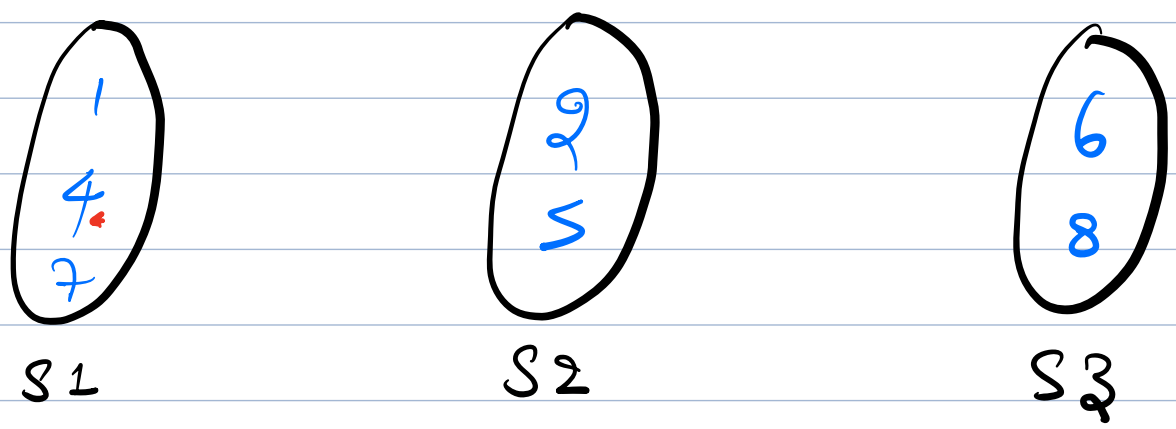
No need to create outdegree array.

How to use Outdegree

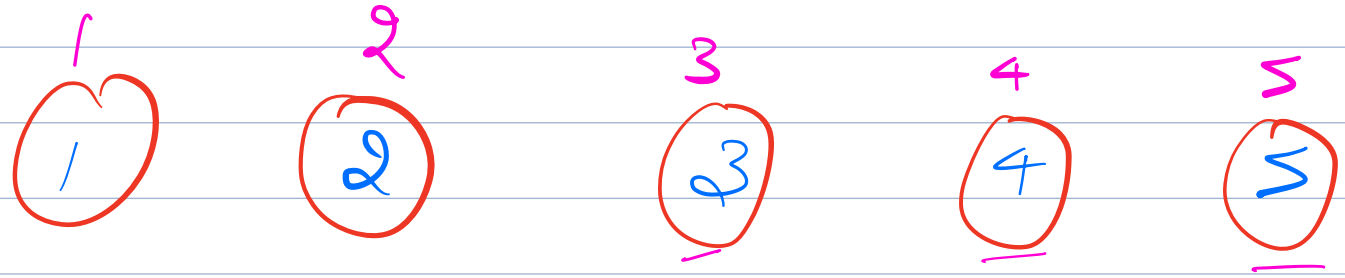
DSU (Disjoint Set Union)

1, 3, 5
2, 4, 8

Disjoint Sets

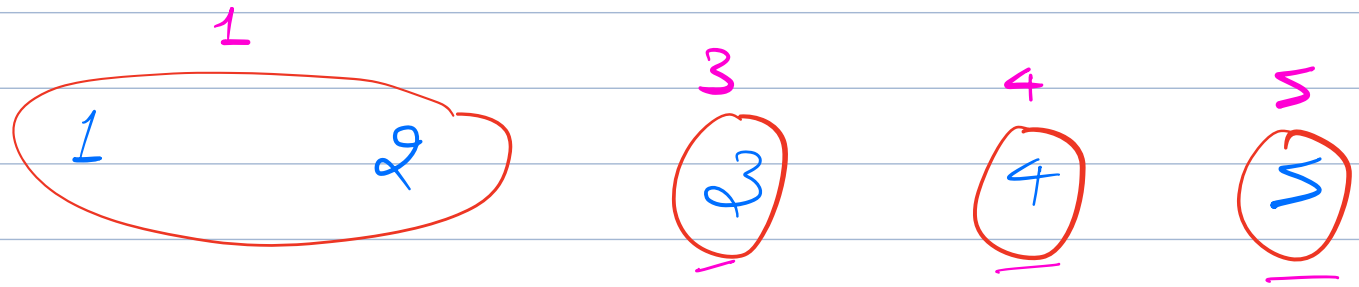


Given N elements → N different sets
no. from 1 to N.

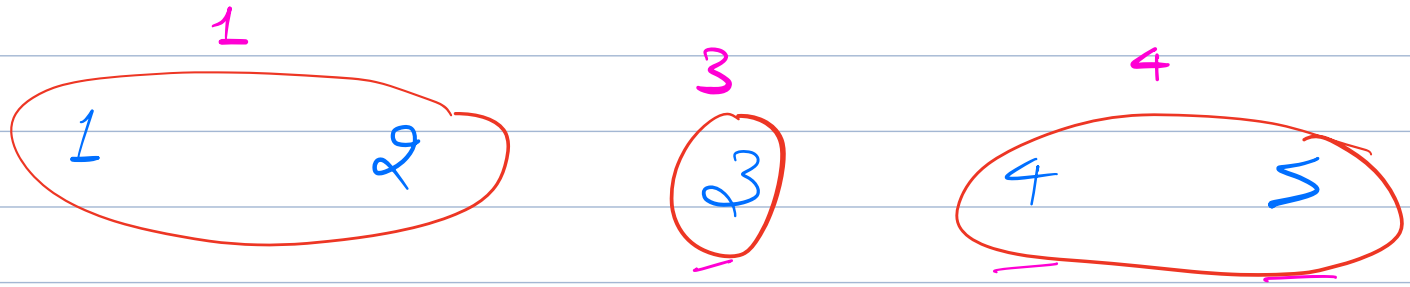


Queries → (u, v) ⇒ If u, v belongs to different sets, merge (union) the 2 sets & return True. False otherwise

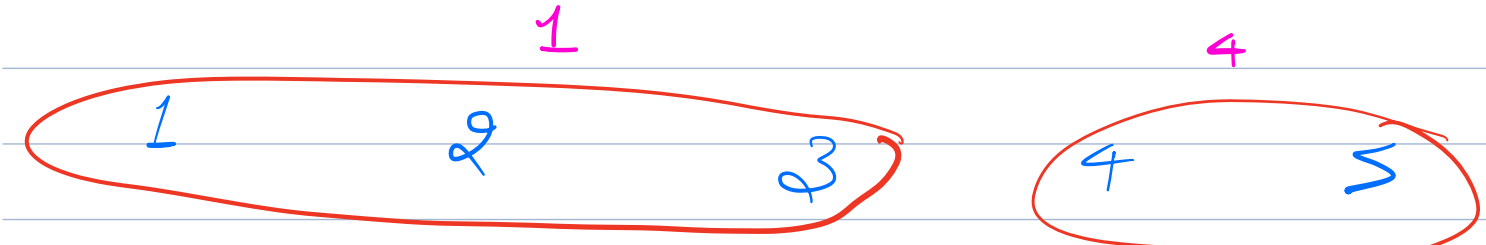
<1, 2> ⇒ True.



<4, 5> ⇒ True.



<2, 3> ⇒ True



$\langle 1, 3 \rangle \Rightarrow$ false

Merge $\Rightarrow$ S_1 , S_2

$\underbrace{S_1, S_2, S_3 \ldots\ldots S_n}$

$X \rightarrow$

(Array) Map $\Rightarrow$ $\langle$ Node, SetNo $\rangle$

$1 \rightarrow N$

$\langle 2, 5 \rangle$

Index of arr

$\langle 1, 1 \rangle$
$\langle 2, 2 \rangle$
$\langle 3, 3 \rangle$
$\langle 4, 3 \rangle$
$\langle 5, 3 \rangle$
$\langle 6, 2 \rangle$

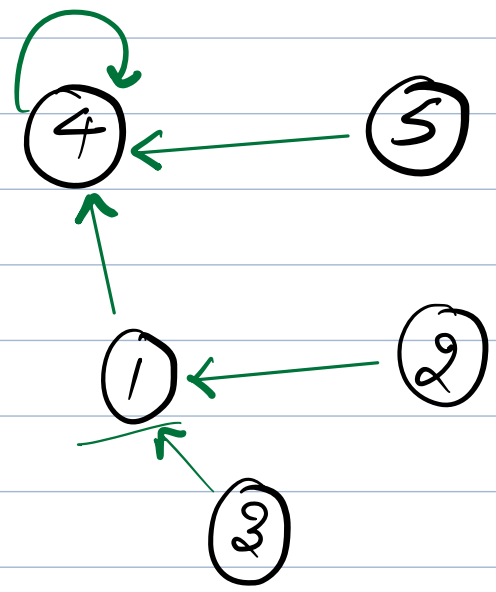
Set $\Rightarrow$ Tree / Elements $\Rightarrow$ Nodes

Identify ⇒ Root of Tree

~~Node {~~
~~int data;~~
~~Node parent;~~
~~}~~

	0	1	2	3	4	5
parent =	~~X~~	4	1	1	4	4

<1, 2> <2, 3>
<4, 5> <3, 5>



1) <1, 2>

parent[1] = 2 parent[2] = 1

2) <2, 3>

root(2) = 1
root(3) = 3

parent[1] = 3 parent[3] = 1

3) $\langle 4, 5 \rangle$

root(4) = 4
root(5) = 5

parent[4] = 5 parent[5] = 4.

4) $\langle 3, 5 \rangle$

root(3) = 1
root(5) = 4

parent[1] = 4 parent[4] = 1

```
int root(u){
    while(parent[u]!=u){
        u = parent[u];
    }
    return u;
}
```

T.C. = O(Height)

⇓

O(N)

```
bool union(u, v){
    x = root(u);
    y = root(v);
    if(x == y){
        return false;
    }
    parent[x] = y;
    return true;
}
```

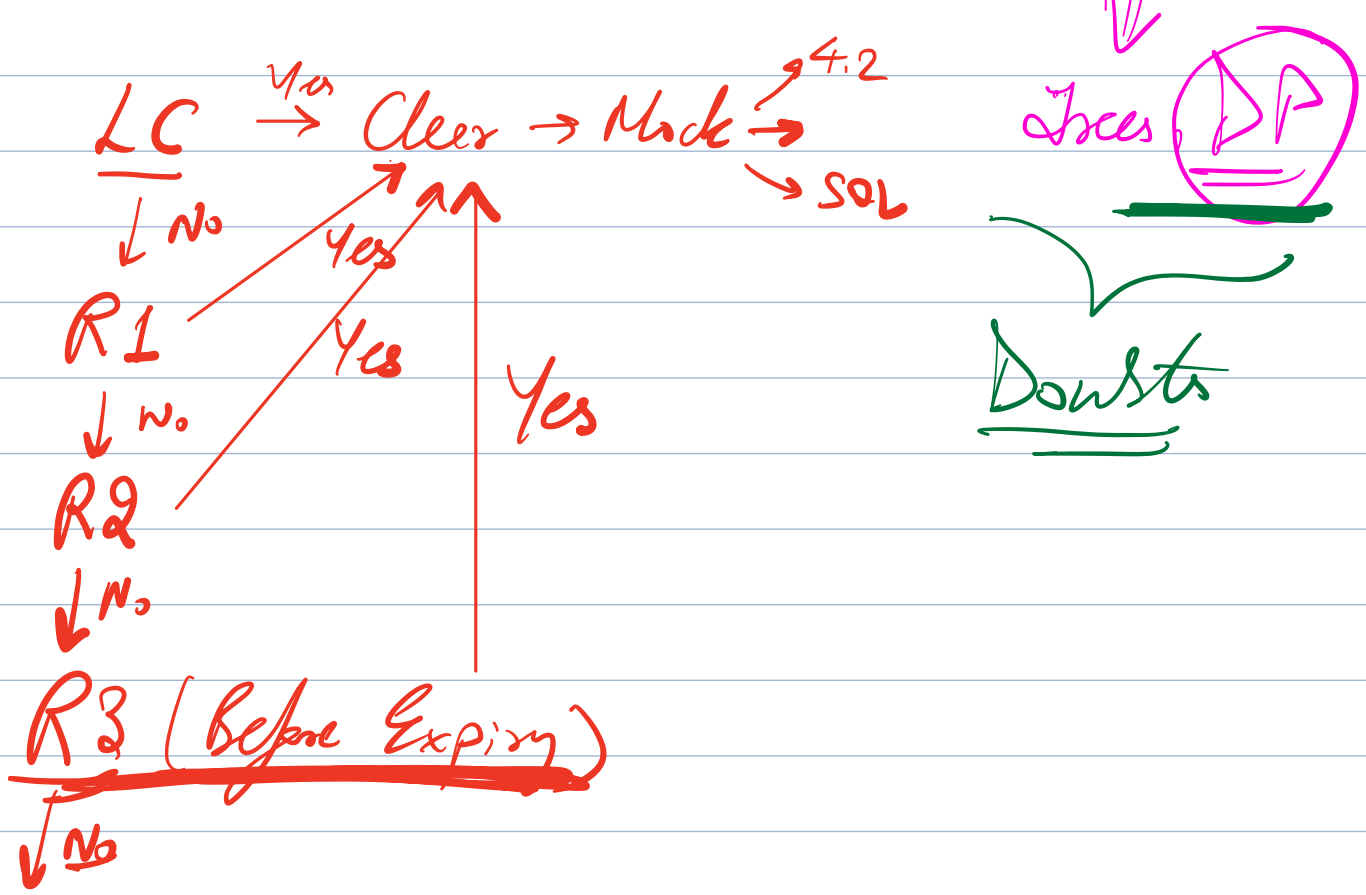
O(N)

1) Optimise

2) Application in Graph

⇓ 4PM PS session

⇓


LC
Yes
Clear
Mock
4.2
SQL
No
R1
Yes
Yes
No
R2
No
R3 (Before Expiry)
No
Yes
Focus
DP
Doubts