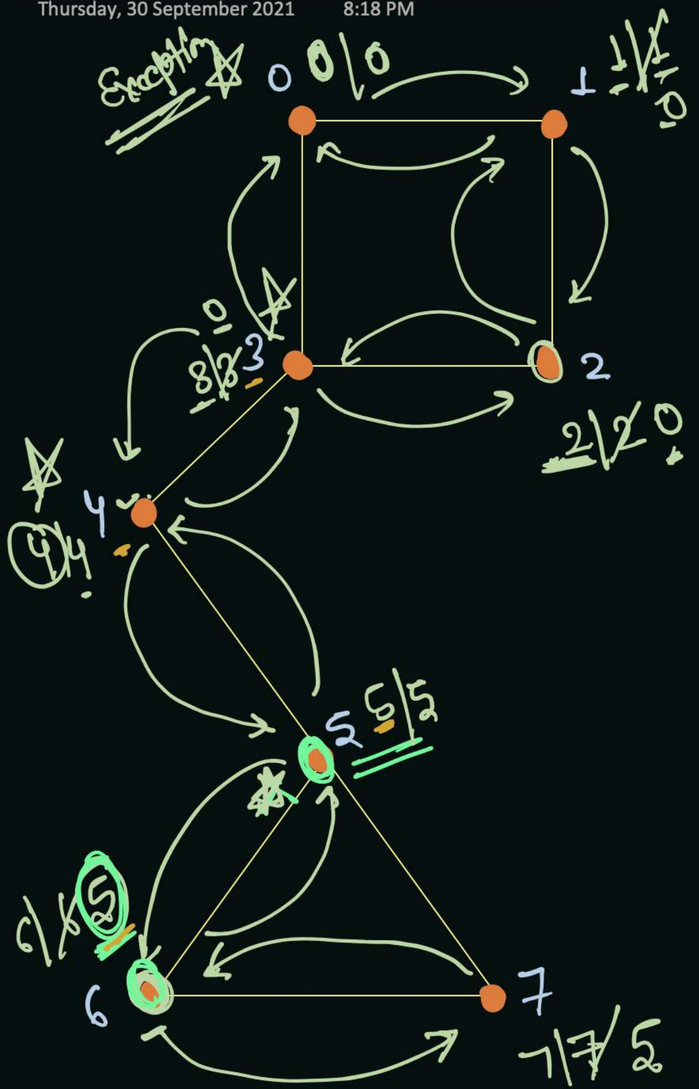
Articulation Point Revision

Thursday, 30 September 2021 8:18 PM



Given → connected graph.

→ No. of articulation point in it?

Articulation point → Articulation point is a point with if we remove it, graph will dis connected.

Requirement →

time=0

① Parent array

② Discovery time → Time at which vertex is discovered.

③ low time. → Time of lowest discovered vertex which is reachable from current point

there is Exception of A.P. in Starting point/ Root.

type of neighbour →

① → neighbour is parent

② → neighbour is visited but not parent

③ → neighbour is unvisited

nbr Loop

```
if( parent[src] == nbr ) {
    //nothing to do
} else if( vis[nbr] == true ){
    low[src] = Min( low[src], disc[nbr] );
} else {
    → articulationpoint( nbr );
    → low[src] = Min( low[src], low[nbr] );
    if( disc[src] <= low[nbr] ) {
        artipoint[src] = true;
    }
}
```

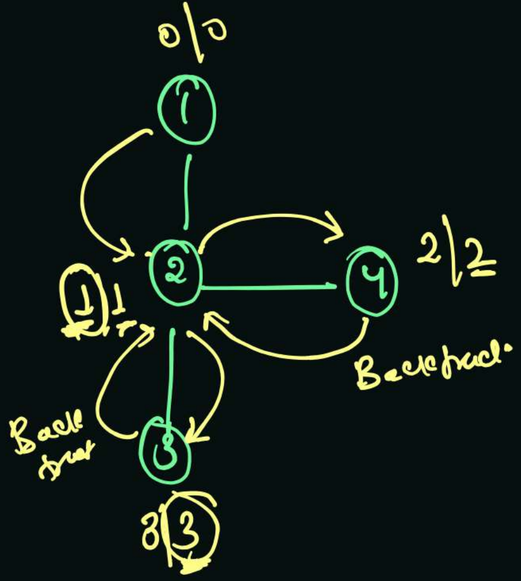
How to manage Root

① why??

② why not - count++ ??

```java
private static void dfsArticulation(ArrayList<ArrayList<Integer>> graph,
int src, boolean[] vis, int[] parent, int[] disc, int[] low, boolean[] arti) {
    disc[src] = low[src] = time;
    time++;
    vis[src] = true;
    int count = 0; // count for finding original source status about articulation
    for(int nbr : graph.get(src)) {
        if(vis[nbr] == true && parent[src] != nbr) {
            // visited but not parent
            low[src] = Math.min(low[src], disc[nbr]);
        } else if(vis[nbr] == false) {
            // unvisited neighbour
            parent[nbr] = src;
            dfsArticulation(graph, nbr, vis, parent, disc, low, arti);
            low[src] = Math.min(low[src], low[nbr]);
            count++;
            if(parent[src] == -1 ) {
                // starting source / root / original source
                if(count > 1)
                    arti[src] = true;
            } else {
                if(disc[src] <= low[nbr]) {
                    arti[src] = true;
                }
            }
        }
    }
}
```

Parent nbr can be skipped.



src = 1

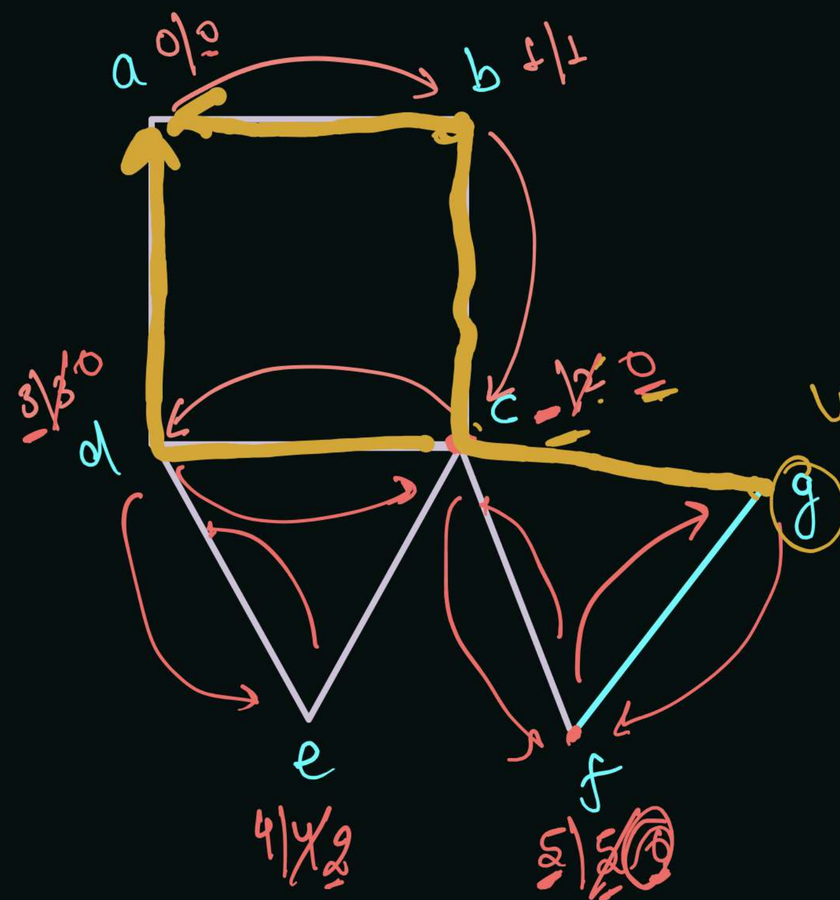
Time = 0 1 2 3 4

A.P. count = 0 1 2

if(disc[src] <= low[nbr])
AP.count++;

there we are incrementing count more than its actual count, thats why we use boolean array to mark articulation point

klhy we min low[src] with dis[nbr] of visited neighbour but low[src] to low[nbr] for unvisited neighbour??



time = 0 1 2 3

4 5 6

lowtime is updated but it required 'c' vertex to reach at 'd'

what are we here try to do is.

Visited but not parent

low[src] dis[nbr]

low[src] = min [low[src], low[nbr], dis[nbr]]

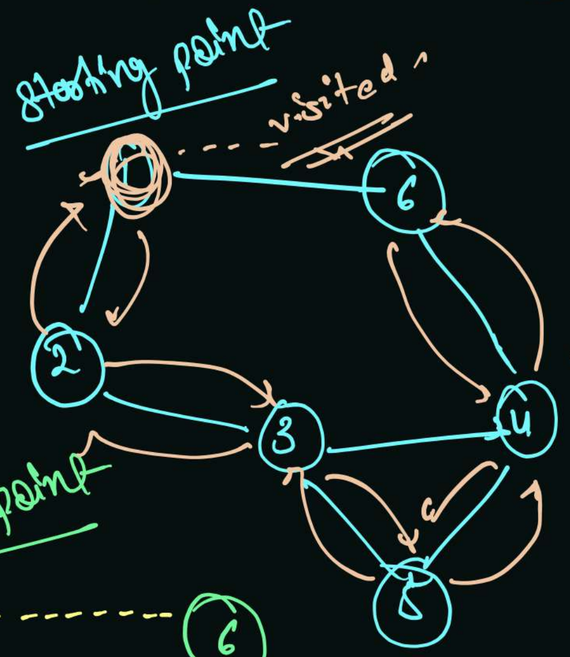
unVisited → Remain same Distance

low[src] = min [low[src], low[nbr]]

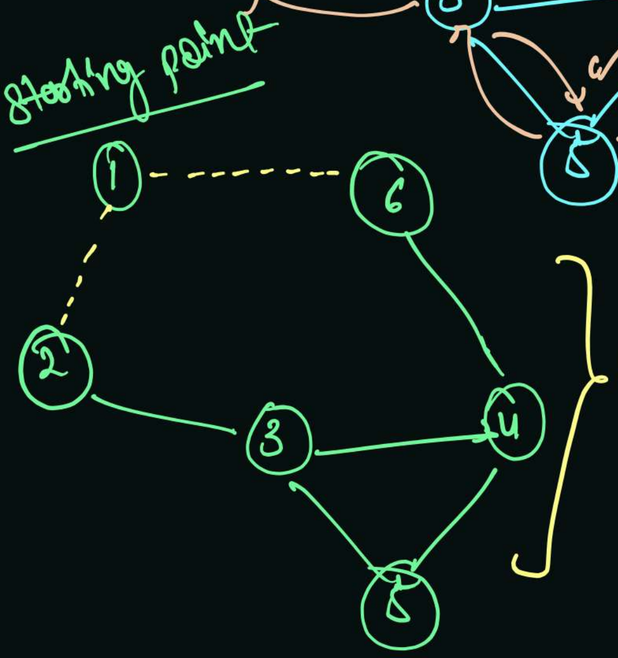
```
if( disc[src] <= low[nbr] ) {
    artipoint[src] = true.
}
```

How to Manage Root →

Example - ①

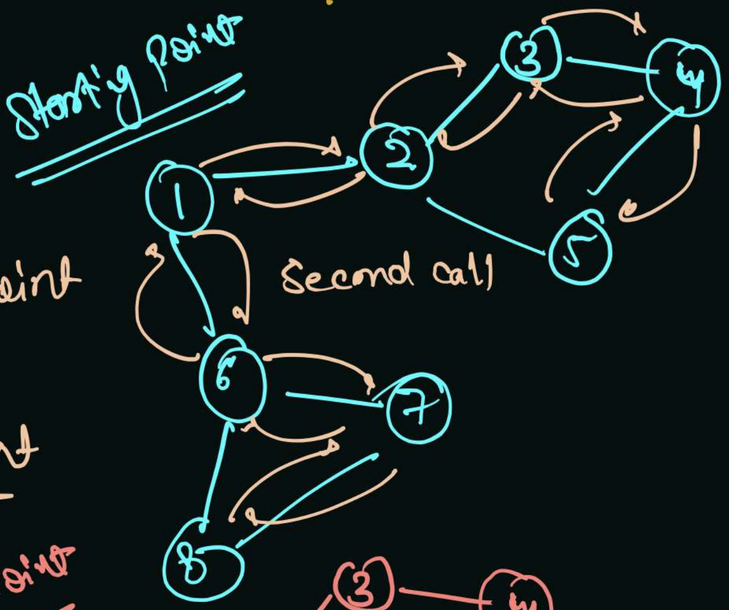


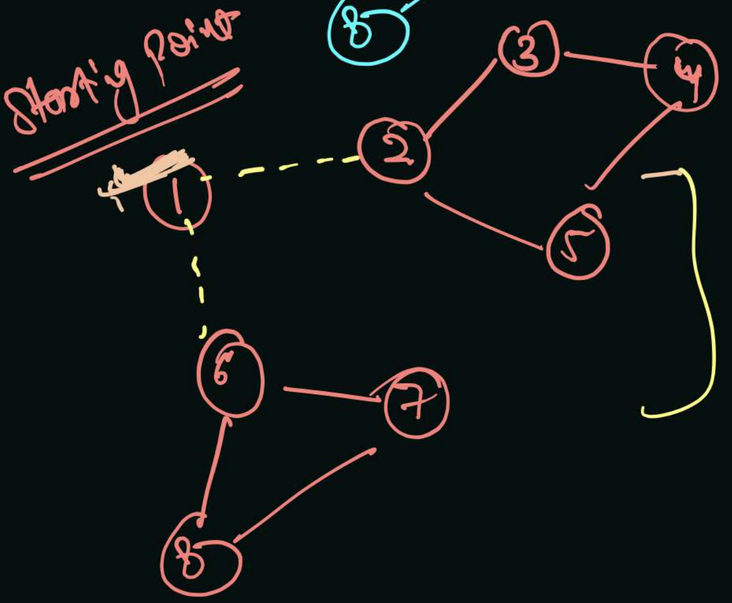
⟸ if we are calling nbr more than one time from initial point then starting point is articulation point



After removing ① graph is still connected

Example ②

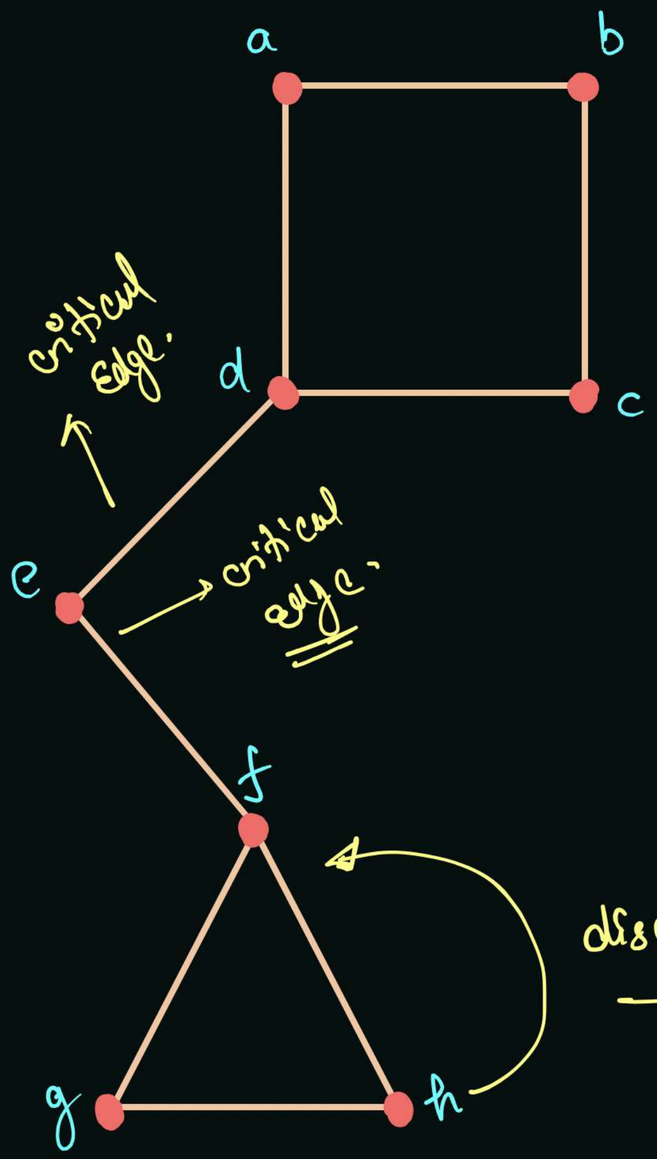




After removing ① graph will disconnected.

Critical Connection

Thursday, 30 September 2021 8:18 PM



connected graph, no. of critical connection??

Critical connection → If we remove an Edge from graph then if graph will convert from connected to disconnected than Edge is "critical connection".

```
if( disc[src] < low[nbr]) {
    [src → nbr] → Edge → critical Edge
}
```

for Articulation Bridge / Critical Connection:

disc[src] <= low[nbr]

→ for Articulation Point

Disjoint Union Set

Thursday, 30 September 2021 10:21 PM

→ It is a concept (Not Exactly belong from graph) but vastly useful in graph.

Key points for DSU → ① Similarity / Grouping ② Transitive similarity.

$a \to b$, $b \to c$ ⇒ $a \to c$

Not necessarily Edge

① 1 - 2

② 3 - 4

③ 1 - 4

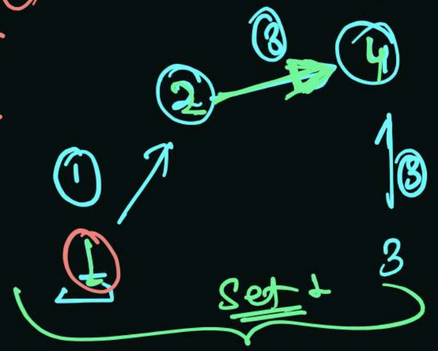
④ 5 - 6

two important part for DSU are:

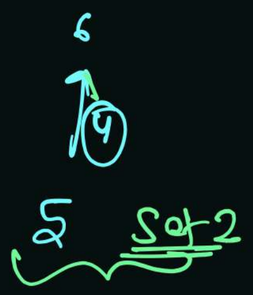
① Union , ② find.

Union b/w set leaders.

logically:-



Set leader of



Values will point to set leader of val2

Requirement → Set leader.

set leader / parent

have no parent

① find set leader

② union of set leader

n=11 no. of nodes

Set leader Rave no parent = set leader is pointing itself.

Set leader

actual view →

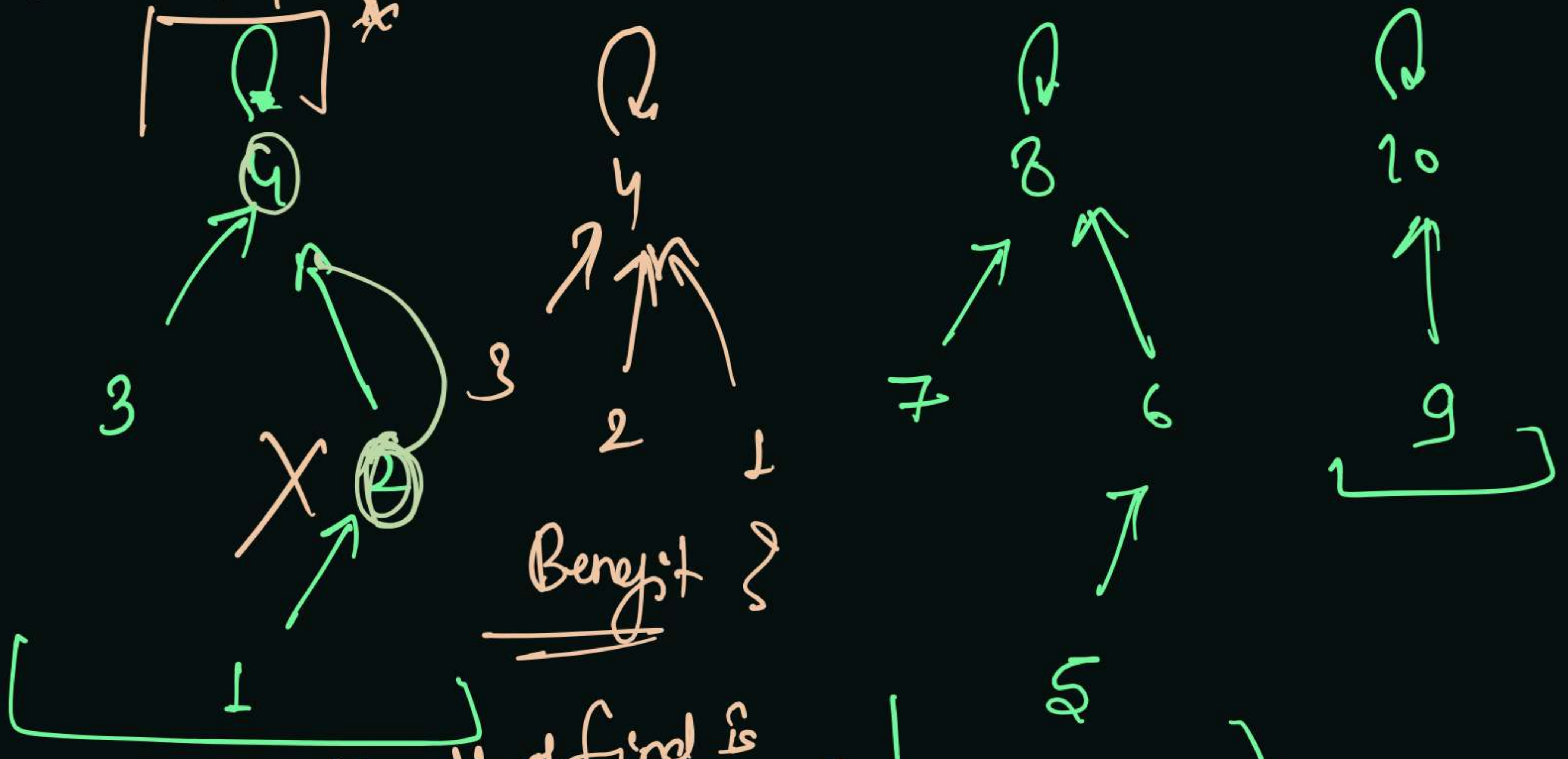
✓ 1 - 2
✓ 3 - 4
✓ 2 - 3
4 = 1 - 4 - 4
✓ 5 - 6
✓ 7 - 8
✓ 5 - 7
✓ 9 - 10

✓ 1 - 3 → query

these are alway not an Edge but virtual view → pair of vertex belongs from same group. Setleader of V1 →

How to find set leader

Not Graph



Complexity of find is reduce ?

optimisation

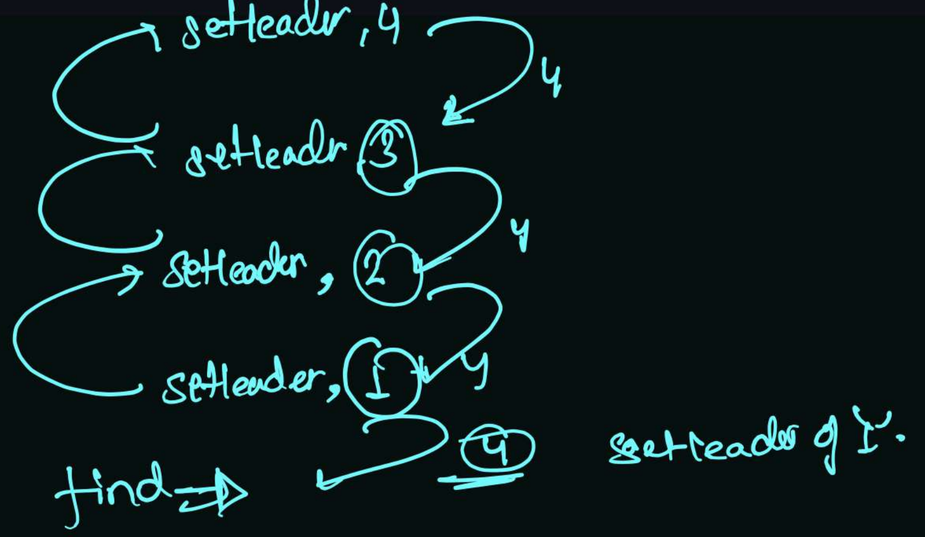
Setleader[x] == x then it will contribute in Conf.

find what is setleadr of 'i' ?


```java
public static int find(int[] setleaders, int x) {
    if(setleaders[x] == x) {
        return x;
    }

    int temp = find(setleaders, setleaders[x]);
    return temp;
}
```

Base case



union.

union

```java
// n-> no. of vertex
int[] setleaders = new int[n];
for(int i = 0; i < n; i++) {
    setleaders[i] = i;
}

for(int[] pair : pairs) {
    int v1 = pair[0];
    int v2 = pair[1];

    int slofv1 = find(setleaders, v1); // slofv1 -> set leader of val1
    int slofv2 = find(setleaders, v2); // slofv2 -> set leader of val2

    // set leader of val1 point to set leader of val2 -> logically correct by not optimised
    setleaders[slofv1] = slofv2;
}
```

value
value

union

without optimisation.

optimisation → ① Path compression] → Before optimisation: O(V) ⟷ log(V); after optimisation individually
② Union by Rank → O(V) ⟷ Log(V)

included both → O(αV)

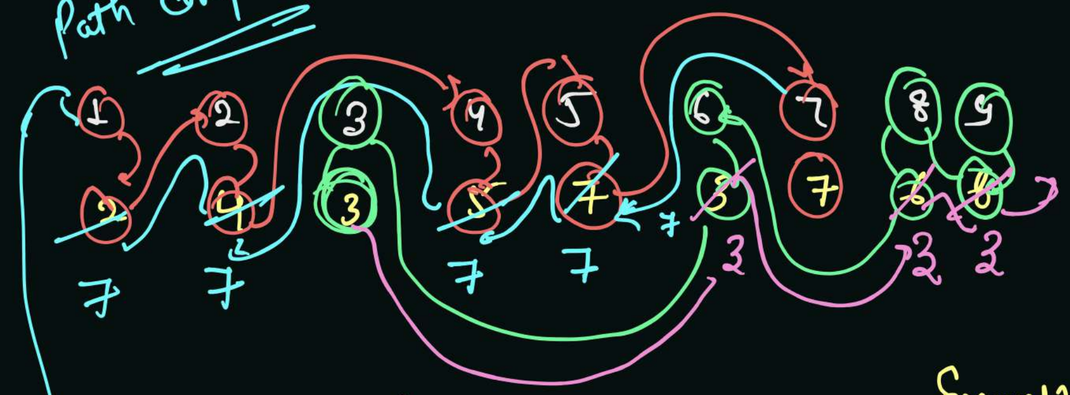
where α(V) < 4

logically we are correct, Now we optimising our code

→ Amortised Analysis

Lifestyle update

Path Compression



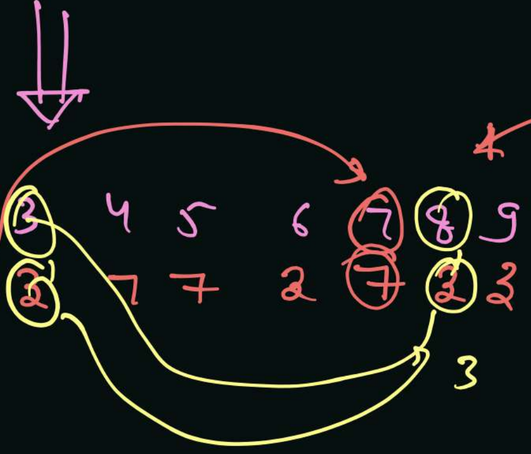
Example-1

val1 = 1

val2 = 9

setleader of val1 = 7

setleader of val2 =



Example-2

setleader of 2 ⇒ 7

setleader of 8 ⇒ 3

optimisation

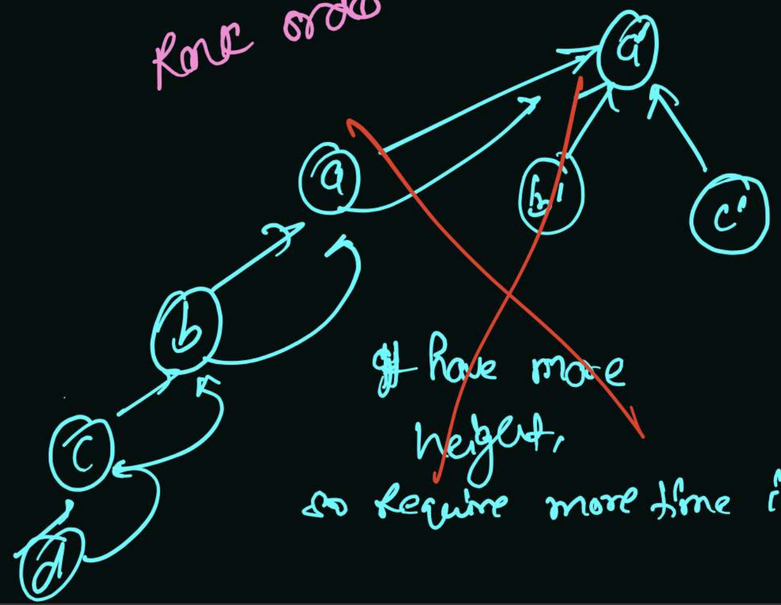
Union by Rank →

In Union → Merge trees according to their Ranks?

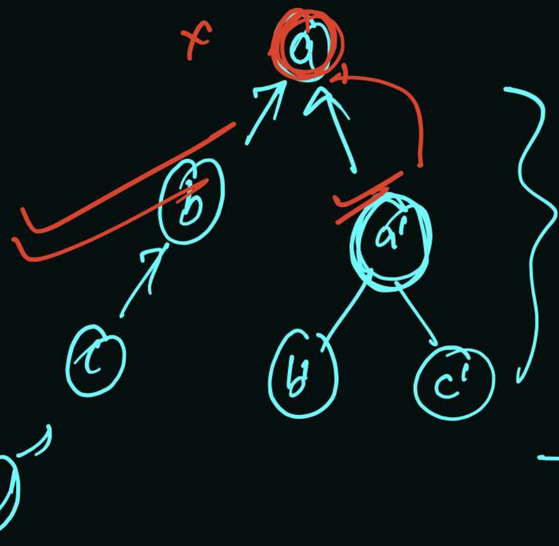
↳ height

with Rank order

Min Rank set leader will point to max Rank set leader.



Rank more height, so require more time in 'find'



less height compare from other one.

→ optimised on

Static graph — DFS vs DSU — Identify no. of elements in single group — Similarly — (V+E)

n = 7

Dynamic graph — Add edges — DFS vs DSU — (D+E+P.U)

Components → 7

6 queries

n = 7

Query

Query	Components	
0 - 1	→ 6 (V)	
3 - 2	→ 5	(1) Amortised Analysis
4 - 6	→ 4	(1)
1 - 2	→ 3	1
0 - 3	→ 3	1
2 - 3	→ 3	1
4 - 5	→ 2	1
5 - 6	→ 2	1
3 - 4	→ 1	1

DFS — K × (V+E)

DSU — (V+E+DSU)

α(V) ≤ 4

2nd Oct — Gandhi

arr → o/p →

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

DSU (1)

DFS (1)

→ After solving all queries → no. of group. — Static graph

→ with each pair → no. of pairs — Dynamic graph