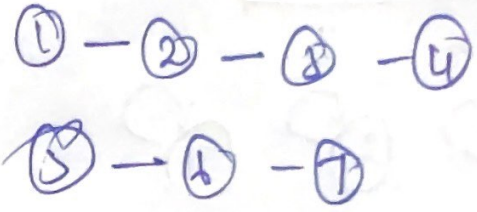
function in all smartphones named call since it is being declared in abstract base class.

## Disjoint set




Whether 4 and 7 belongs to same graph?

We can do DFS and say No. But it will take almost O(N).

eg. 

We can make it to constant time

1) union by Rank
2) union by size

### Union by Rank

ds {1, 2}

Step 1: Find ultimate parent of u and v.

Step 2: compare rank of ultimate parent of u and v.

Step 3: attach smaller rank component to larger one and increase the rank by 1.

edges.

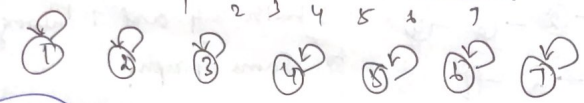
{1,2} {2,3} {4,5} {1,7} {5,6} {3,7}

Initially rank = | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1 2 3 4 5 6 7

parent = | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

1 2 3 4 5 6 7



{1,2}

p[1] = 1 = up_u

p[2] = 2 = up_v

rank[up_u] = 0 rank[up_v] = 0.

p[2] = 1

r[1] += 1

{4,5}

p[2] = 1 = up_u

p[5] = 5 = up_v

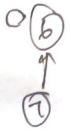
r[up_u] = 1

r[up_v] = 0

p[5] = 1

rank[1] += 1

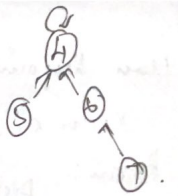
{6,7}



{5,6}

up_u = 4

up_v = 6



Now ultimate parent of 7 : 4

up of 5 : 4

So 7, 5 belongs to same tree. but it will take logn time. So we do path compression here.

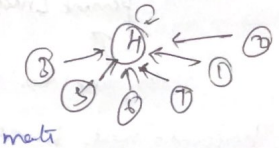
when 6 is attached to 4, all of its children will get attached to 4.

{3,7}

up_u = 1

up_v = 4

rank(1) = 1 r(4) = 2



how do we achieve it?

While finding ultimate parent of u and v, we first update the value of parent using recursion