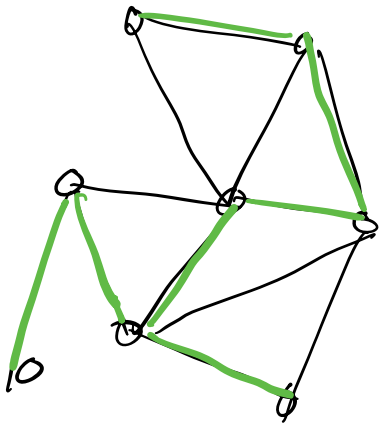
Delivery guy.

N houses → 1 CC.

Can you remove some roads s.t. it's still possible to deliver to all houses

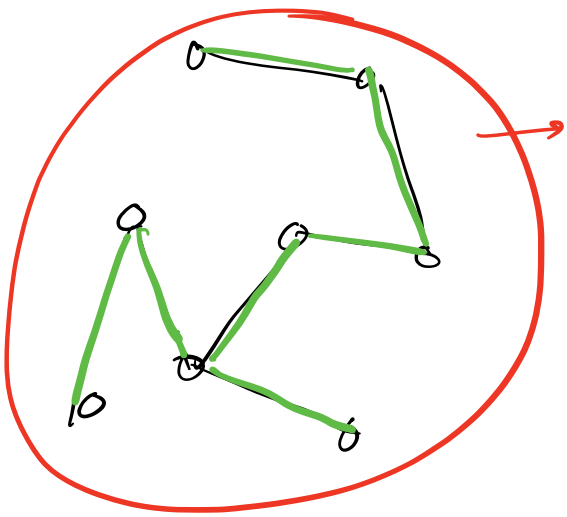
OR

the graph left should have 1 CC



What is the MAX # edges that we can remove

$G \rightarrow \{V, E\}$



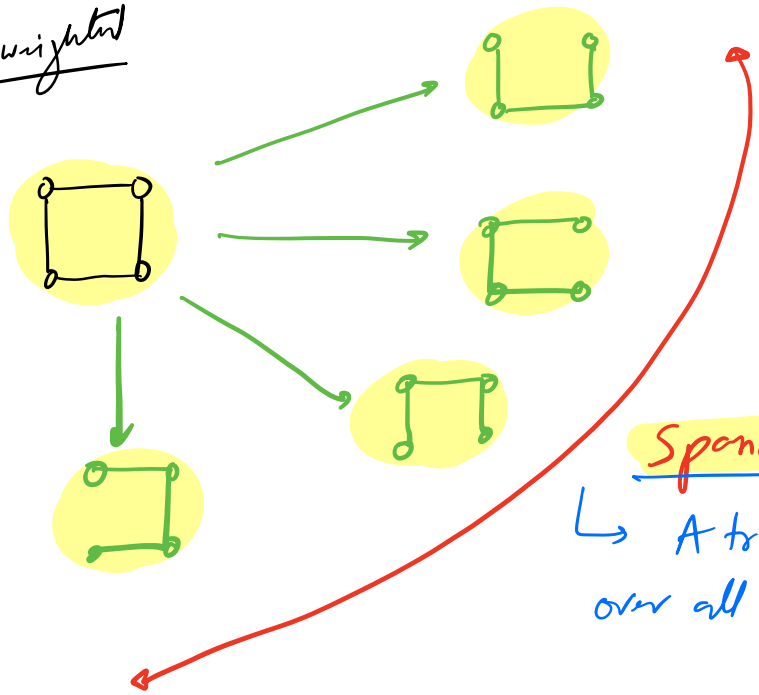
V

$E' = V-1$

Edges we can remove

$= E - (V-1)$

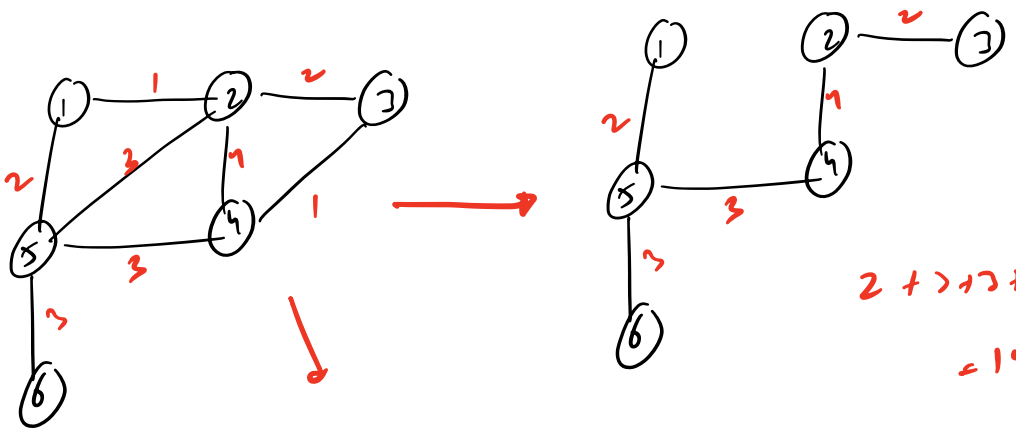
Unweighted



Spanning tree

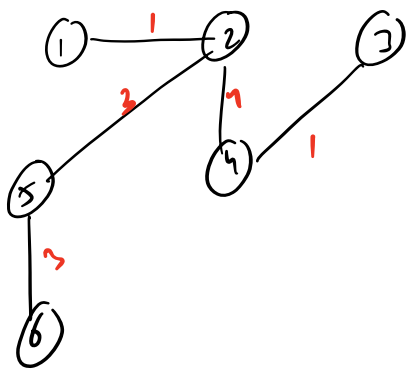
↳ A tree that spans over all the nodes!

Weighted



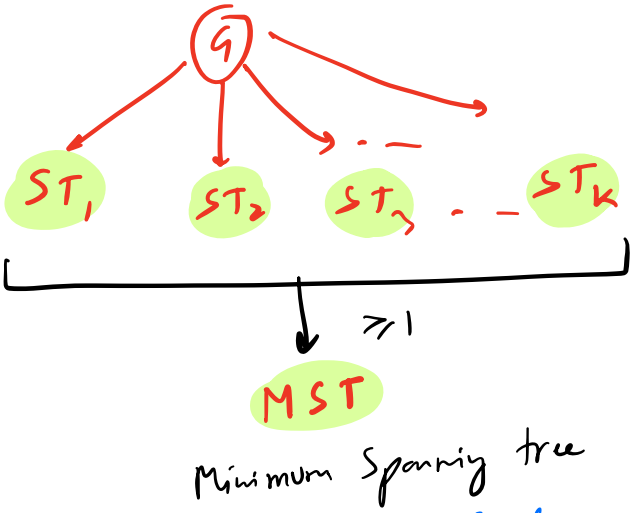
$2+2+3+4+2$

$= 13$ //



$1+3+4+1+3$

$= 12$ //



Minimum Spanning tree

→ the ST with the least sum of edges!

⊙ 2 Algo's for MST

1) Prim's Algo
2) Krustkal's Algo → DSU

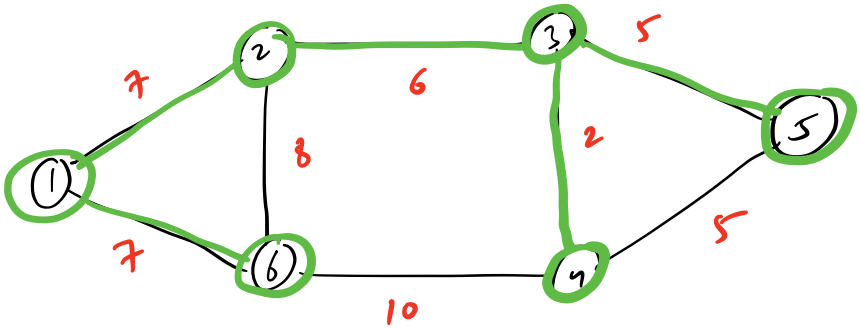
All the discussion going forward would be for Simple graph

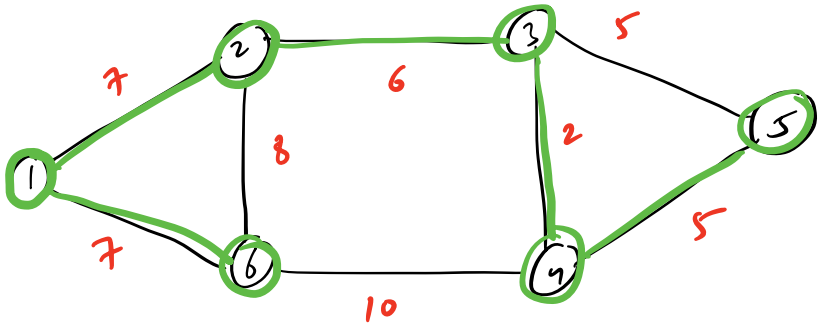
1) No self loop

2) No multiple edges

1) Prim's Algo's →

1)





<wt, <from, to>>

<1, <1, 5>>

<7, <1, 2>> X

<6, <1, 6>>

<5, <5, 2>>

<2, <5, 7>>

<8, <7, 8>>

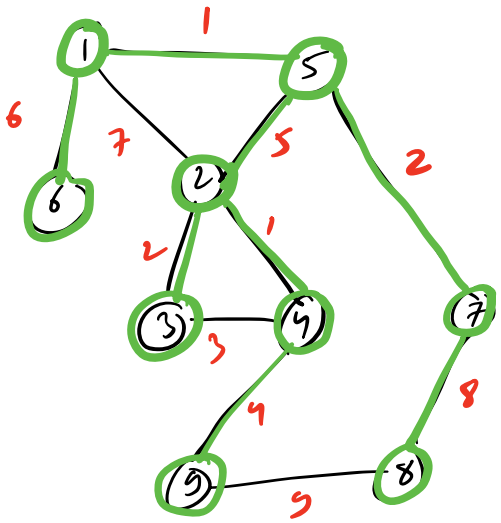
<2, <2, 3>>

<1, <2, 4>>

<3, <4, 5>>

<4, <4, 9>>

<9, <9, 8>>



Req of DS

1) GET MIN
2) INSERT
3) DEL MIN

MIN HEAP

vis [] + E

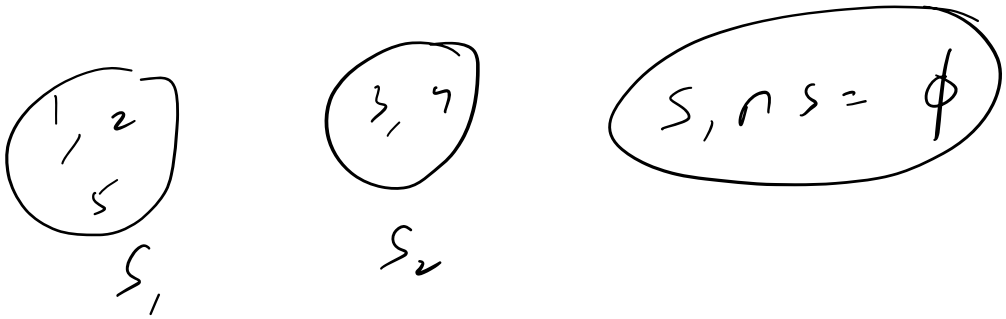
TC = O(V+E log E)

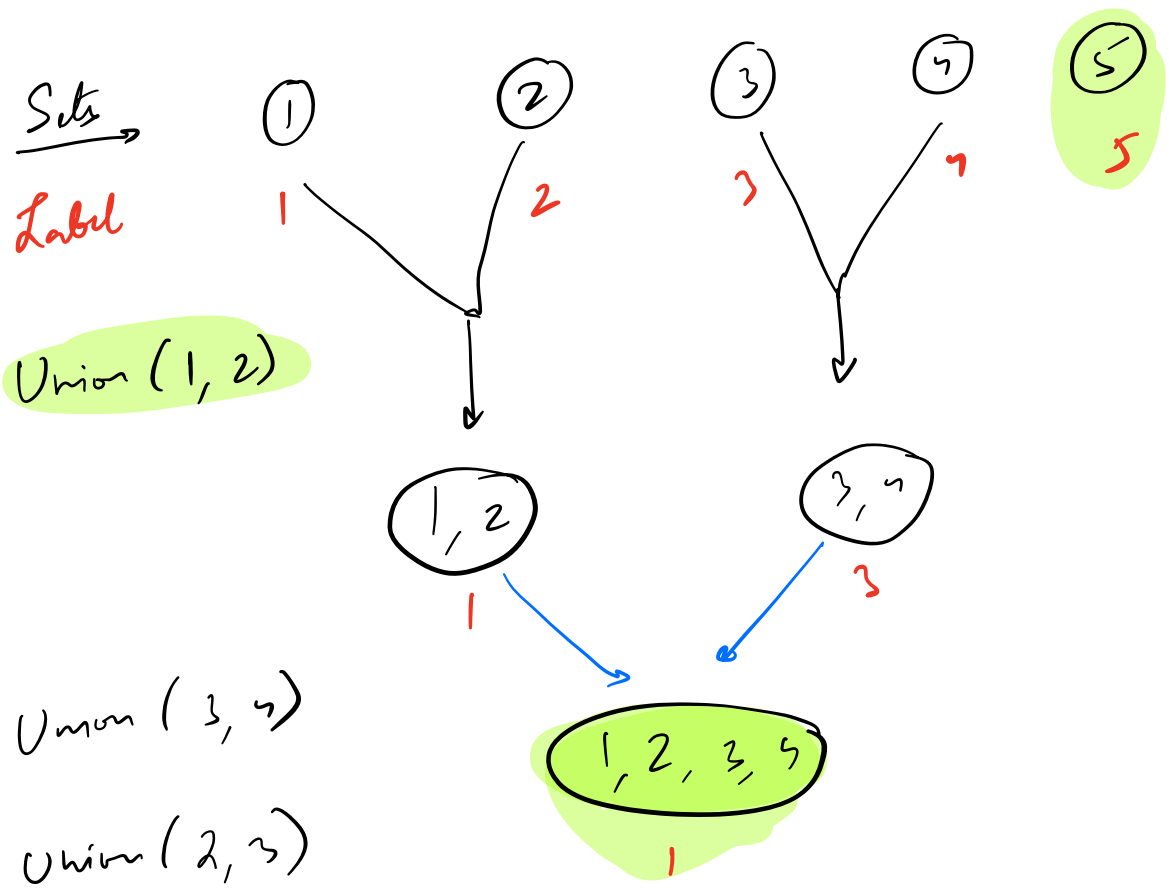
SC = O(V+E)

DSU

[Disjoint Set Union]

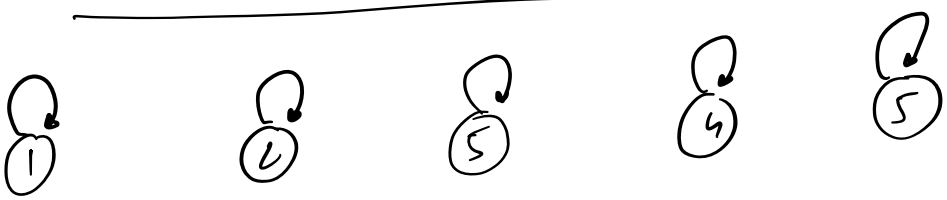
Disjoint set → do not have anything in common



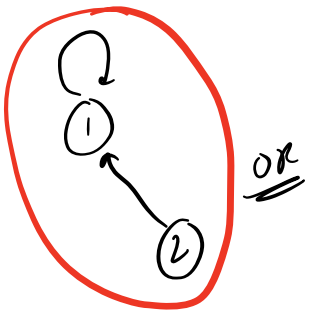


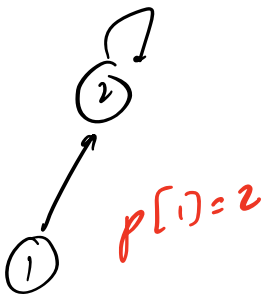
① Consider all the sets as trees →

int p[N];
∀i p[i] = i;



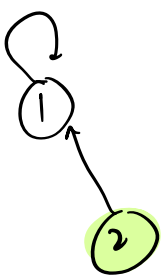
union (1, 2)

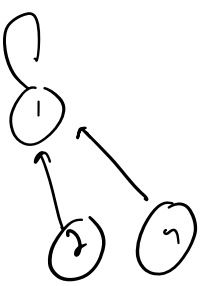




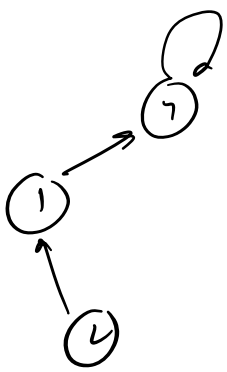
p[2] = 1

union (2, 4)





or



CODE

int p[N+1] ⇒ ∀i p[i] = i ; INITIALIZE !

```
int find(int x) {
    while (x != p[x]) {
        x = p[x];
    }
    ret x;
}
```

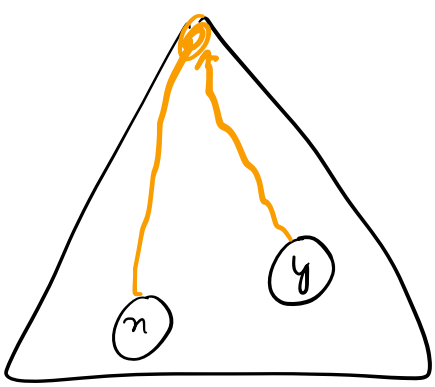
TC = O(H)

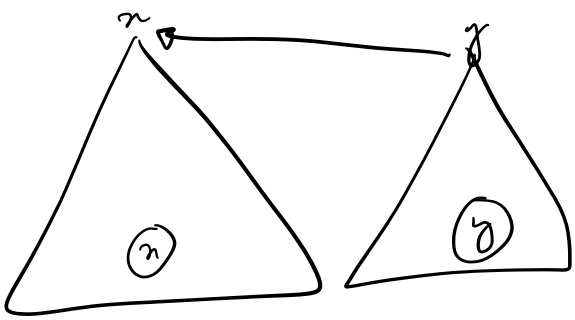
N

```
union(x, y) {
    x = find(x);
    y = find(y);
    if (x == y) ret;
    p[y] = x;
}
```

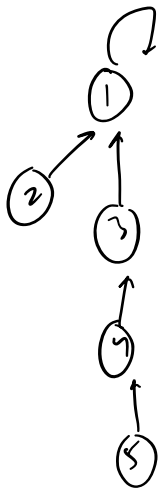
TC = O(H)

N



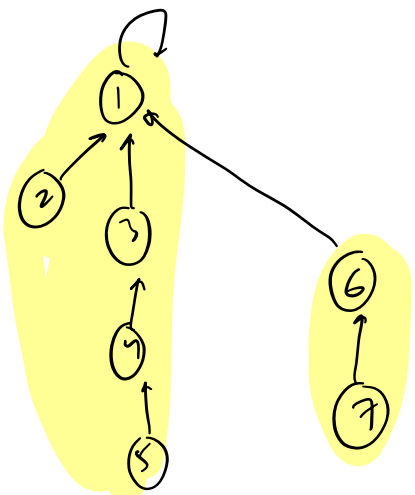


Optimize →



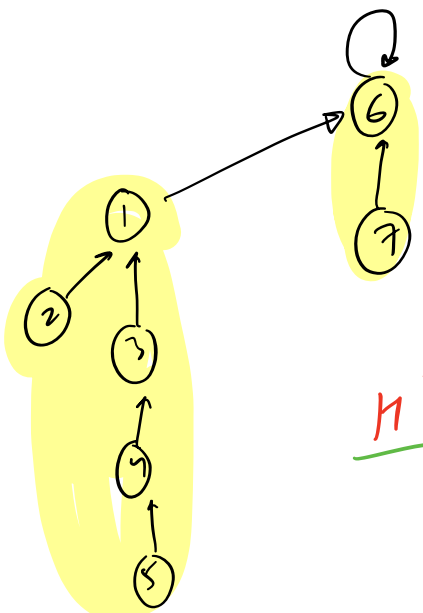
H = 3

H = 1



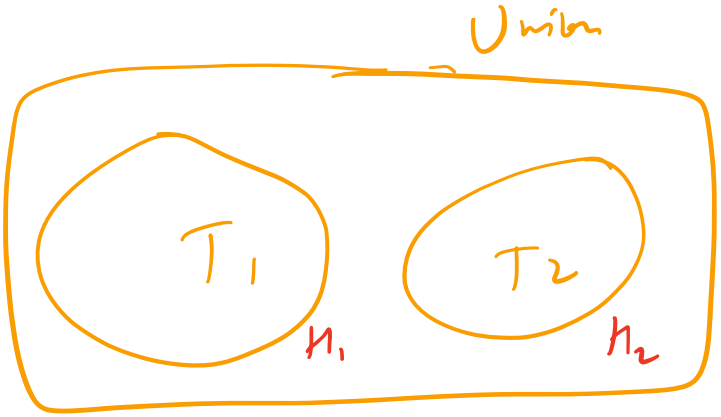
H' = 3

3 + 0



H' = 4

3 + 1

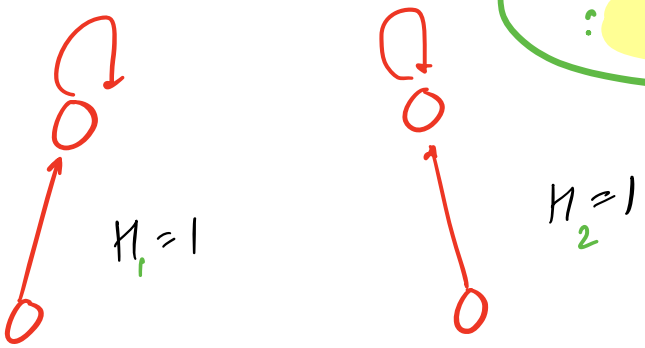


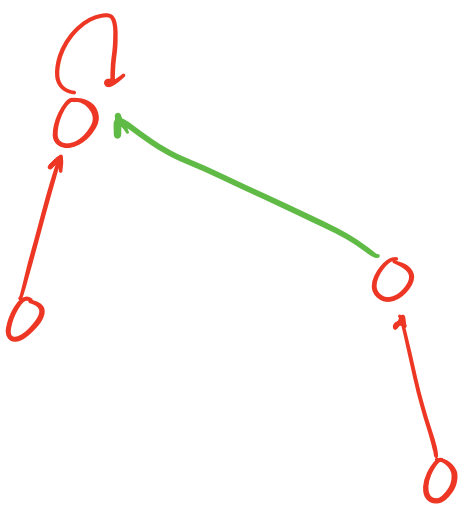
$H' = \max(H_1, H_2)$

: $H_1 \neq H_2$

$H' = H + 1$

: $H_1 = H_2$

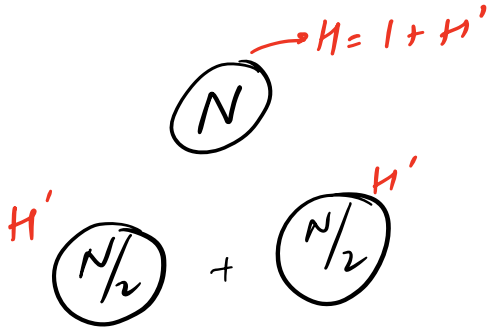




$H' = H + 1$

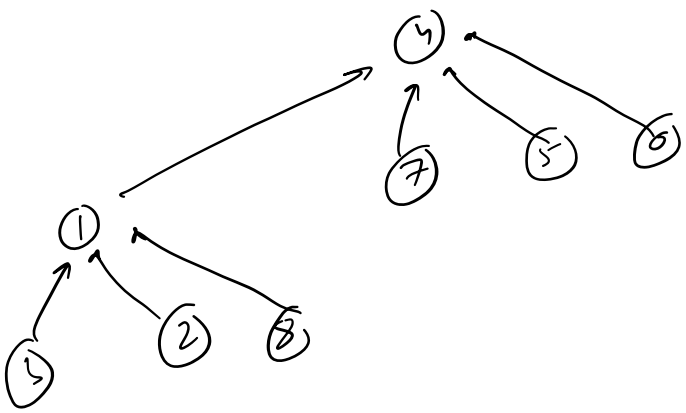
$= 2$

N nodes



$$H(N) = H(N/2) + 1$$

$$H(N) = \log_2(N)$$



CODE

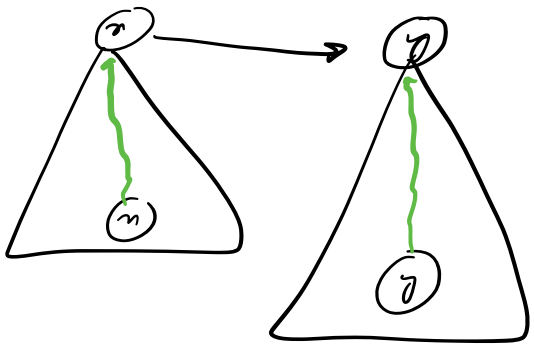
find() → NO CHANGE

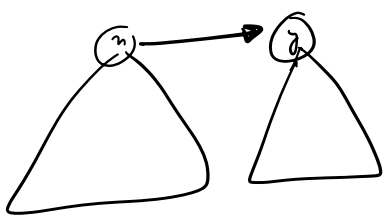
int H[N] → {0}

$p_i = i \ \forall i$

UNION BY HEIGHT

```
Union ( x, y) {
    x = find(x);
    y = find(y);
  if(x==y) ret;
  if ( h[y] > h[x]) {
       p[x] = y;
  }
  else if( h[x] > h[y]) {
       p[y] = x;
  }
  else {
       p[x] = y;
       h[y] ++;
  }
}
```

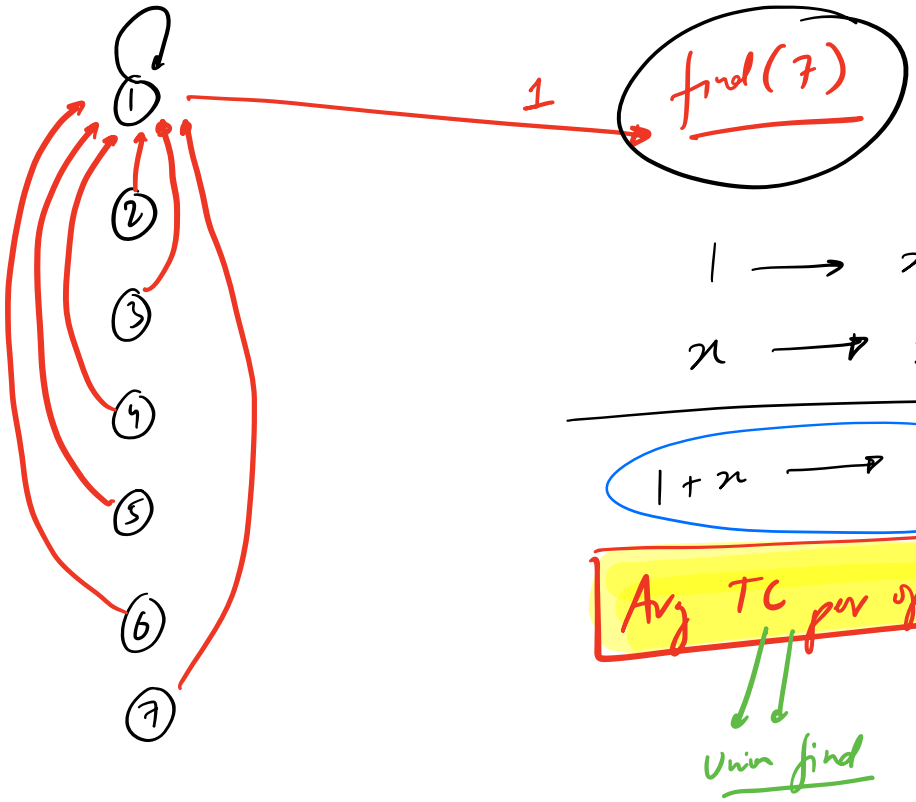




$H \leq \log_2 N$

$TC = O(f(N))$ → Union, find

① Path Compression



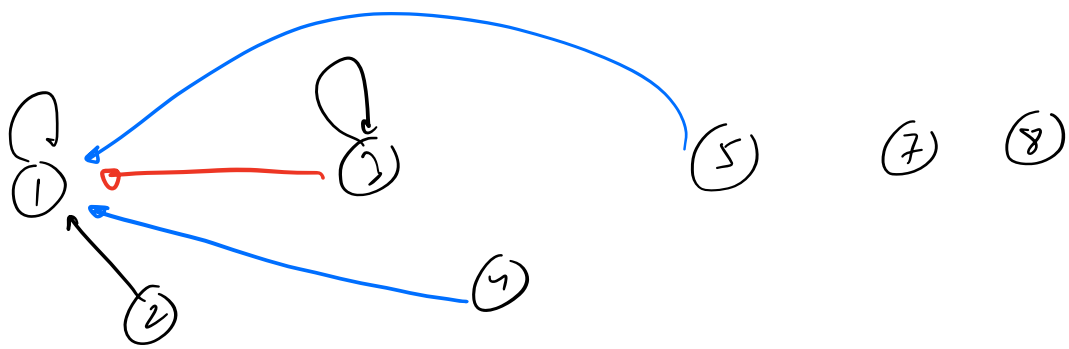
$1 \longrightarrow x$

$x \longrightarrow 1$

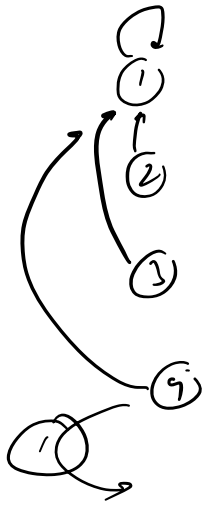
$1 + x \longrightarrow x + 1$

Avg TC per op $\rightarrow O(1)$

Union find



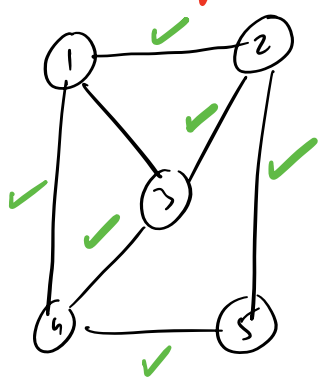
```
int find (n) {
  if (n == p[n]) ret n;
  p[n] = find(p[n]);
  ret p[n];
}
```

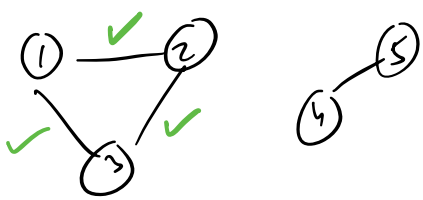


TC Amortized per q
= O(1)

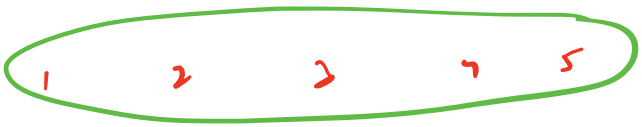
Q Given an undirected graph. Check if it is 1 cc!

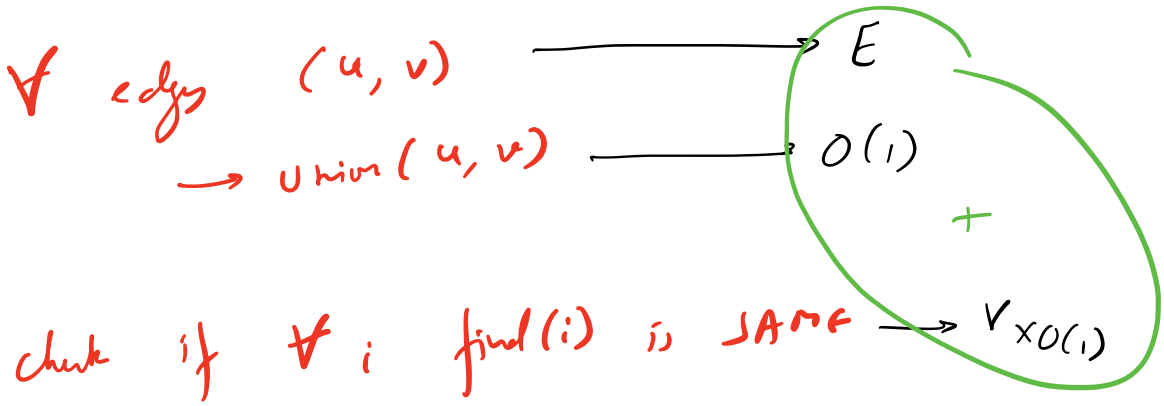
Graph





DSU





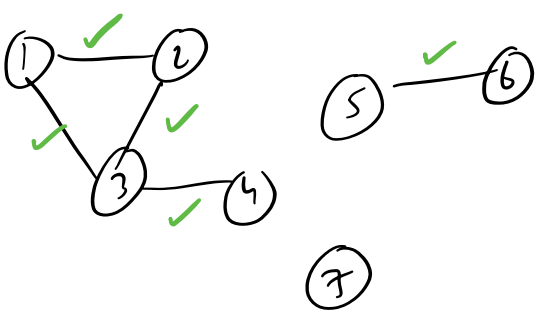
TRUE : 1 CC

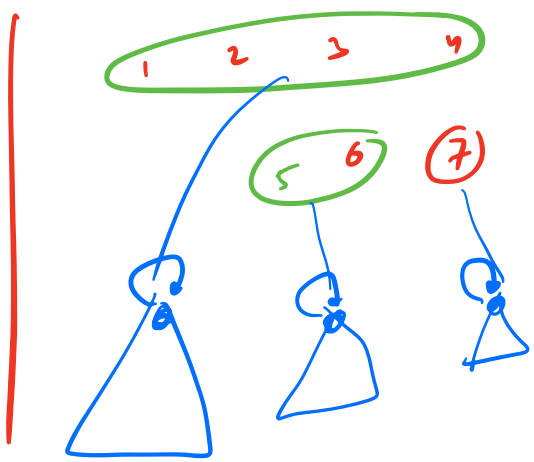
FALSE : > 1 CC

TC = O(V+E)

SC = O(V) p[]

Q Given a graph. find the # of CC's!





$\forall$ edge (u, v) $\longrightarrow$ E

$\longrightarrow$ union (u, v) $\longrightarrow$ $O(1)$

$\times$

$+$

$cc = 0$

for $(i = 1 \longrightarrow V)$ $\longrightarrow$ V

if $(p[i] == i)$

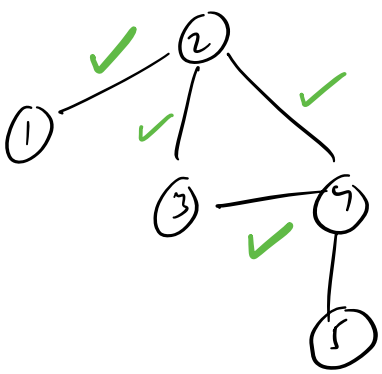
$cc++;$ $\longrightarrow O(1)$

$\times$

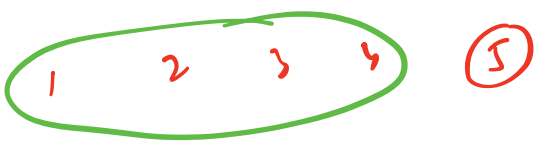
$TC = O(V+E)$

$SC = O(V)$

Q Given an undirected graph. Check if there is a cycle!



DSU



∀ edge (u, v) → E

if (find(u) == find(v))

→ CYCLE ✔

else

Union(u, v)

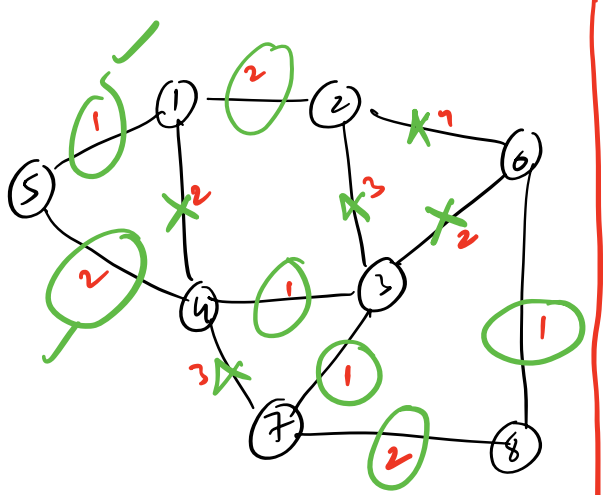
→ O(1)

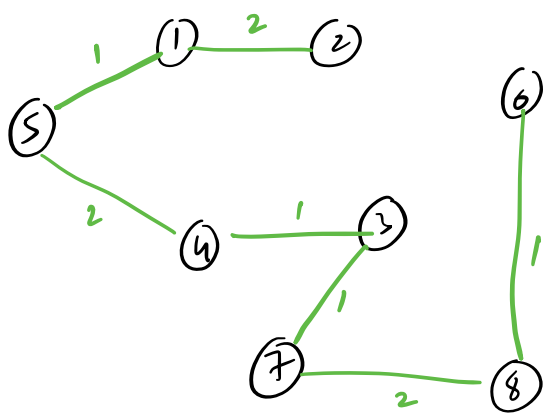
→ NO CYCLE

$TC = O(V+E)$

$SC = O(V)$

KRUSTKAL'S ALGO →





Algo

1) Initialize DSU. → V

2) Sort All edge in INC order of weight → ElgE

3) Iterate over the edge list → E

(w, (u, v))

```
if ( find(u) == find(v))
        skip this edge

else
        // sum += w
        // print( u, v)
        // Edge u,v is in MST!
```

→ O(1)

$TC = O(V + E \lg E)$

$SC = O(V + E)$

dsu, edge list