Let's begin at 9:02 PM

L84
Disjoint Set Union

Join Discord - https://bit.ly/ly-discord

RECAP

What is the problem?

The Problem

1. Imagine we have N nodes, and all nodes are a separate set initially.
2. Now, we get queries of 2 kinds:
    a. Combine the corresponding sets of 2 given nodes.
    b. Given 2 nodes, check whether they are in the same set or not.

N = 6 ⇒ {1}, {2}, {3}, {4}, {5}, {6}

combine(1, 4) ⇒ {1, 4}, {2}, {3}, {5}, {6}

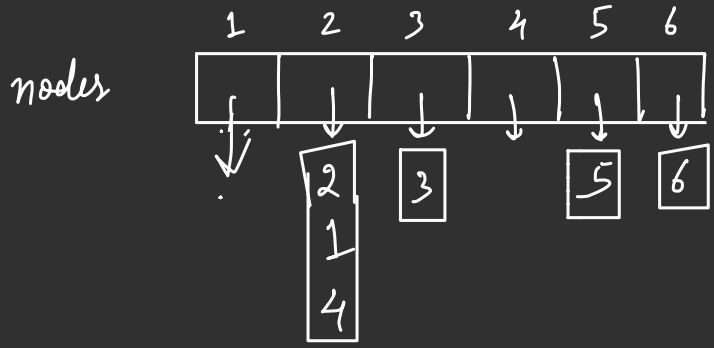
check(1, 2) ⇒ false

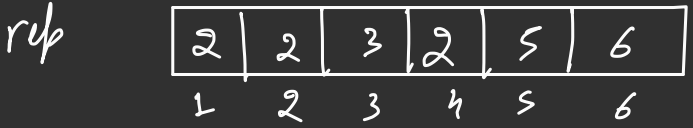
combine(4, 2) ⇒ {1, 2, 4}, {3}, {5}, {6}

check(1, 2) ⇒ true

Come on, come up with some ideas

A Possible Way





union(1,4)

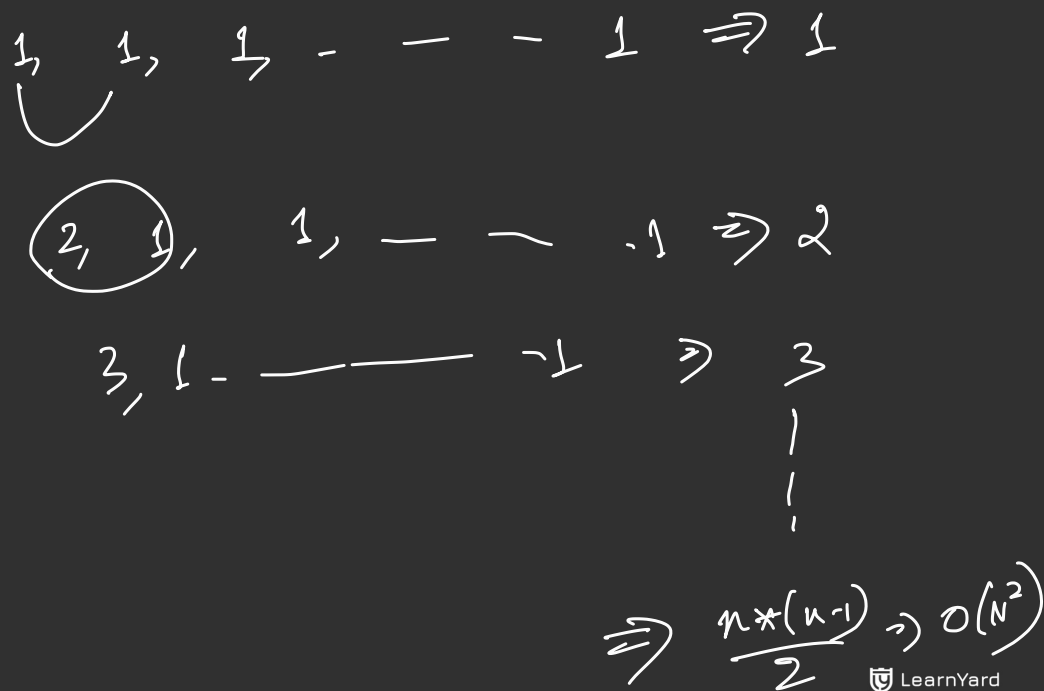
find(1,2) ⇒ f

union(2,4)

find(1,2) ⇒ t

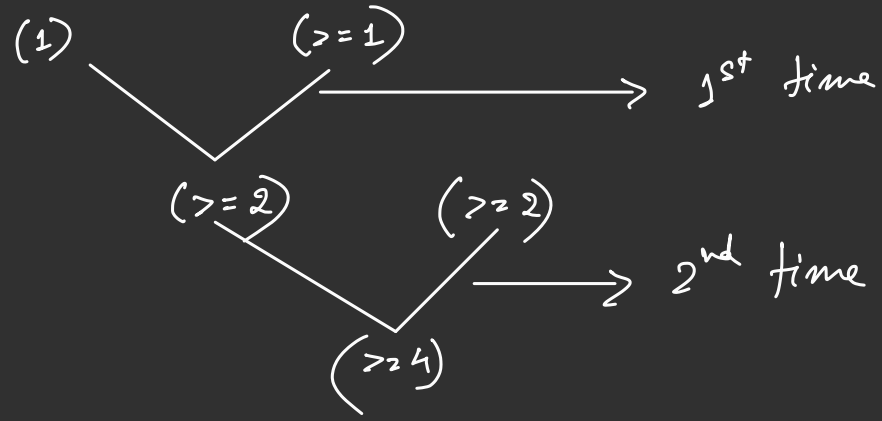
union ⇒ O(N)

find ⇒ O(1)



Small to Large Merging

Consider a node x



$$2^t \leq N$$

$$t \leq \log_2 N$$

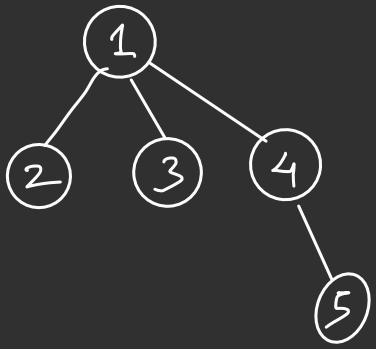
1 element $\Longrightarrow$ $\log_2 N$

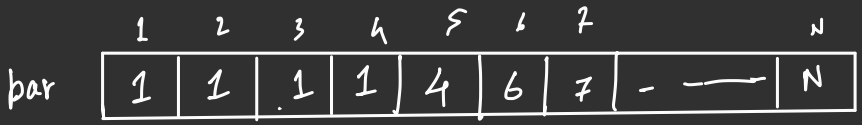
If all elements $\Rightarrow$ not more than $N * \log_2 N$ transfers

Let’s implement this

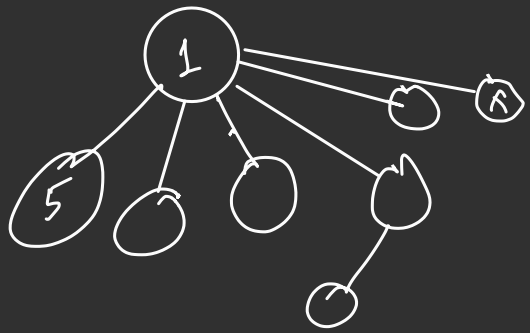
Can we do better?
Can trees help?

un(1, 3)
un(3, 2)





Optimisation 1 - Path Compression



→ Avg. = O(log N)

Optimisation 2 - Union by Size

If 1 & 2 both are done

↳ Time = inverse ackermann func $\Rightarrow \alpha(N)$

almost $O(1)$

(amortized)

$\alpha(N) \leq 4 \quad \forall N \leq 10^{600}$

Let’s implement

Drawback?

The connections, once made, can't be reversed.

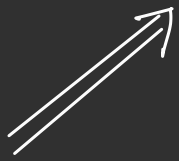
Practice

Find num. of components efficiently

0. Warm Up

Possible way :

```
int cut = 0;
for(i = 1 to n){
    if (par[i] == i)
        cut++;
}
return cut;
```



O(N) time

Thank You!

Reminder: Going to the gym & observing the trainer work out can help you know the right technique, but you'll muscle up only if you lift some weights yourself.

So, PRACTICE, PRACTICE, PRACTICE!