CS 225

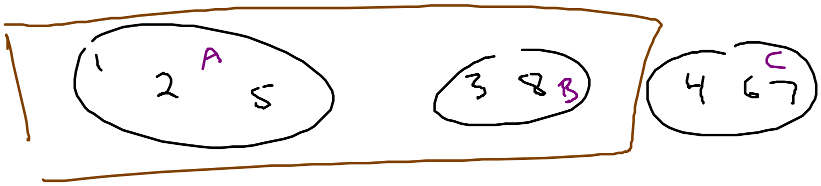
Disjoint Sets: Implementation

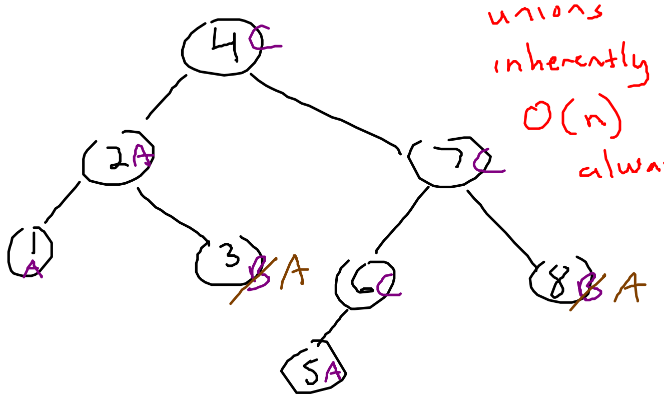
void Union (Set A, Set B)

Set Find (element x)

1) find actual element x] other structure

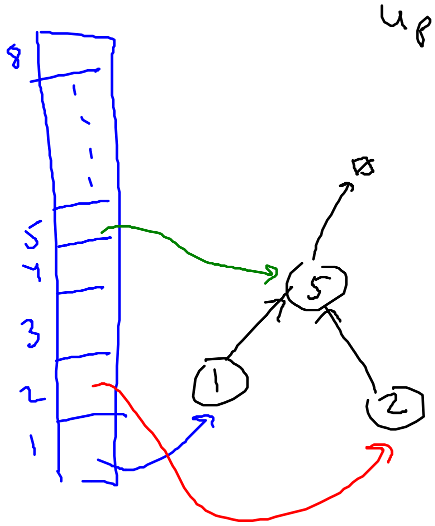
2) find set x is in

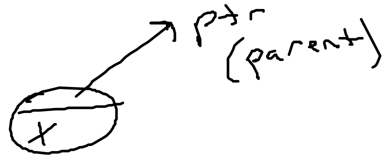




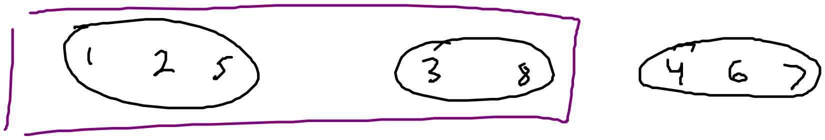
unions inherently $O(n)$ always

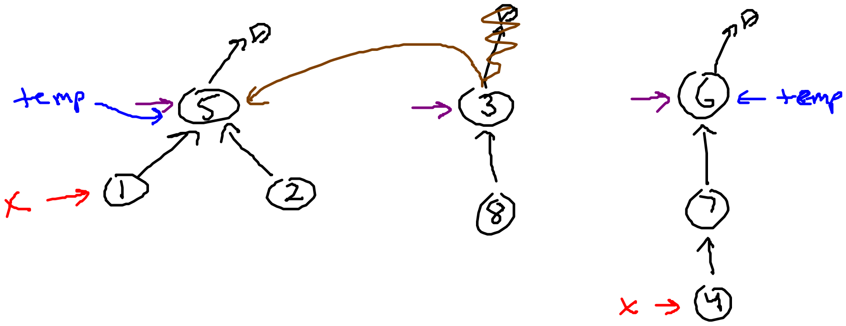
up-tree





<u>not</u> BST


1 2 5
3 8
4 6 7


temp
x
5
1
2
3
8
6
7
4
temp
x

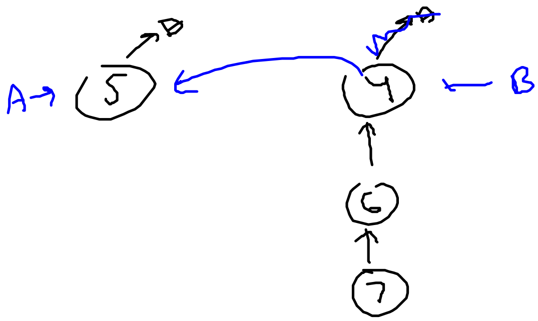
```
Node * Find ( Node * x)
{   Node * temp = X;
    while(temp -> parent != NULL)
        temp = temp -> parent;
    return temp;
}
```

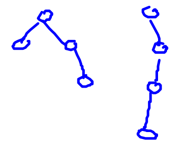
bc: O(1)
ac: O(lg n)
wc: O(n)

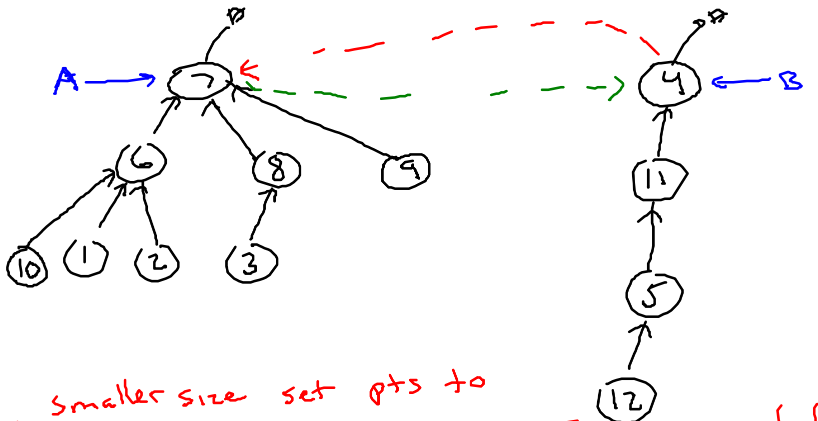
assume set roots

```
void Union (Node * A, Node * B)
{
    B -> parent = A;
}
```

wc: O(1)

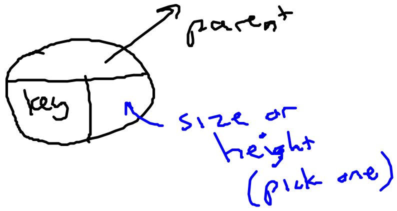

A
5
D
4
A
B
6
7



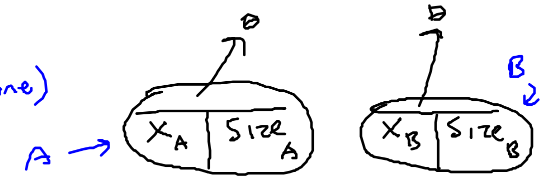


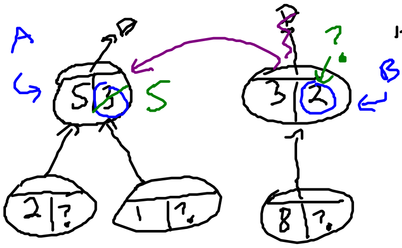
smaller size set pts to larger size set; increase Find time of fewer nodes

smaller height set pts to larger height set; kept deepest node from getting too deep



union by size



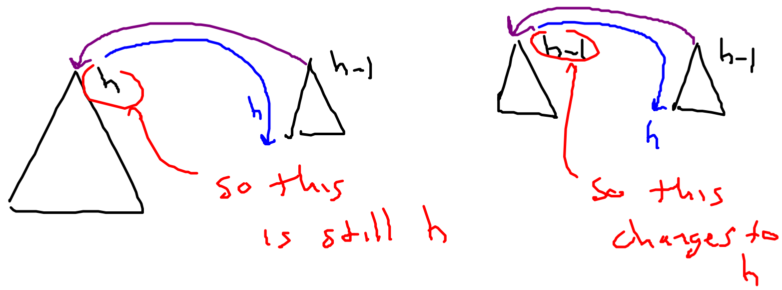



```
if (size_A ≥ size_B) {
    size_A = size_A + size_B
    B->parent = A;
}
else
  // other way around
```

union - by - height

same idea, but point smaller height tree to larger height tree

height only changes if tie



Smart union algorithms

1. union by size
2. union by height bc tree

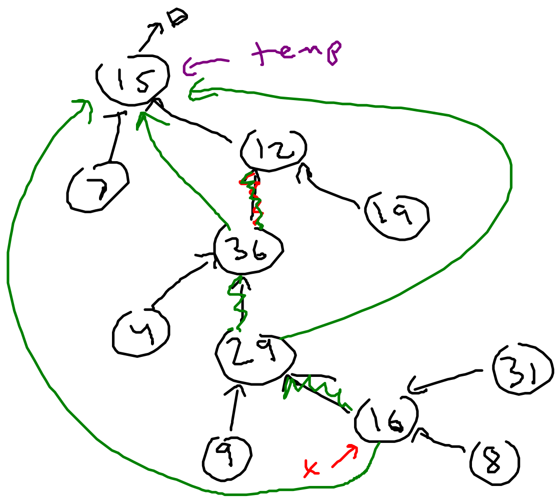
wc : you always get a tie

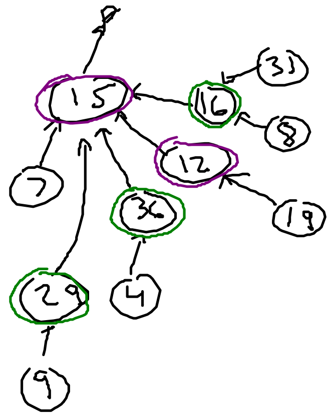
$O(\lg n)$ height trees

↳ wc Find $O(\lg n)$

union still $O(1)$

Path Compression





union-by-size & path compression ok

union-by-height & path compression

union by estimated height

(union by rank)

Amortized analysis

m operations (union, find, $O(1)$ inserting of new values into universe)

$O(m \log^* n)$ universe size n

$\Downarrow$

expected (amortized) cost per operation is $O(\log^* n)$

$$\underbrace{\lg \dots \lg \lg \lg}_{i \,=\, \log^* n} n <= 1$$

$$2^{65536} \to 65536 \to 16 \to 4 \to 2 \to 1$$

$$\log^* 2^{65536} = 5$$

almost constant