Ex: Implement BFS w/o Q, using O(n) space.

CSE525 Lec15

Disjoint-set

•••

Debajyoti Bera (M21)

{3,-2,10} {43,Q} {12}

Disjoint-Set data structure

Maintain a collection of disjoint-sets. Every set has a "leader" element in the set that is used to identify it.

{ 3 , Table , 10 , -2 }   { 12 , 5 , AB , 7 } { 43, 12, 51, Q }

- MakeSet(x): Create a new set {x}. Leader is x.
- Find(x): Return the leader of the set containing x.
- Union(x,y): Replaces two Find(x) and Find(y) by their union. Return leader of the new set.
- No duplicates. (Assumption for now. How to implement?)

MakeSet(-2)
Find(-2) → -2
MakeSet(43)
MakeSet(Q)
Union(43,Q)
MakeSet(10)
Union(10,-2)
Find(Q)
MakeSet(12)
Find(-2)
MakeSet(3)
Union(-2,3)
Find(3)
...

Exercise: Implement DS using different types of Linked Lists.

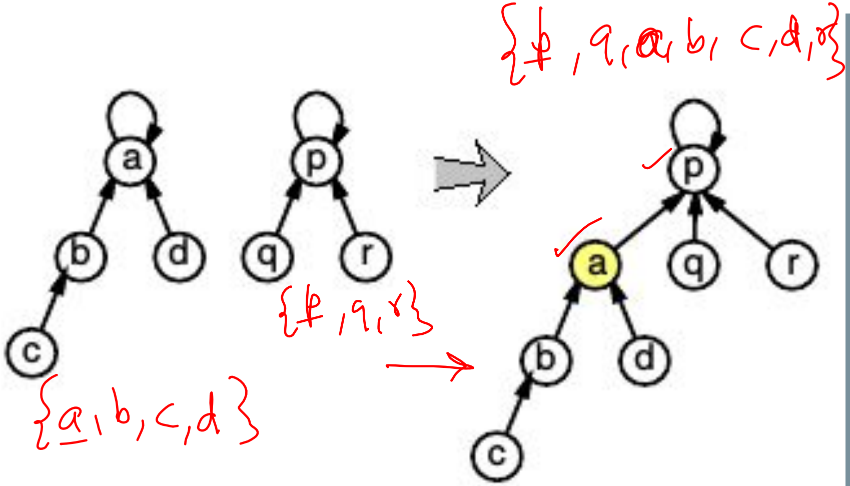
Disjoint-Set using reversed trees

```
MakeSet(x):
    parent(x) ← x
```

```
Find(x):
    while x ≠ parent(x)
        x ← parent(x)
    return x
```

```
Union(x, y):
    x̄ ← Find(x)
    ȳ ← Find(y)
    parent(ȳ) ← x̄
    return x̄
```

x



Tight complexity of ...

- Makeset $O(1)$
- Find $\Theta(n)$
- Union " $O(\text{find})$ " $\approx O(n)$ $\therefore \Theta(n)$

Give a technique to **generate** a sequence of ops. on **n** elements so that Find(x) takes Θ(n) time.

def GenerateTightExample(n):

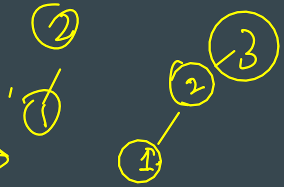
MakeSet(1)

⋮

MakeSet(n)

Union(2,1) →

Union(3,1) →



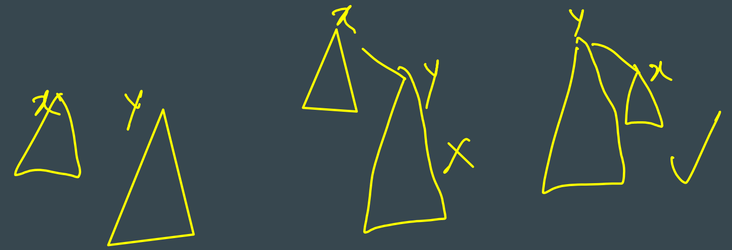
Union(4,1)

⋮

Union(n,1)

Find(1) ← will require $\Theta(n)$ steps.

Reversed-Trees + Union-by-depth



During union, join the shorter tree to the root of the deeper tree (not the other way).

Q: How to know "shorter" and "deeper" ? With element x store height(x).

When x is a leader, ht(x) stores the height of the tree.

Lemma: For any set A, $|A| >= 2^{depth(A)}$. [⇒ depth(A) <= log(#elements of A)]

$|A| = 1024$, depth(A) ≤ 10

Proof: Let x = leader of A. Induction on the depth of the set with x as leader.

{x} {x, .} {x, ...} . . . . {x . .} → x becomes nonleader

- size(x's set) = 1: $|A| = 1$, depth $= 0$, $1 \geq 2^0$ ✓
- Assume that $|x\text{'s set}| >= 2^{depth(x\text{'s set})}$ for depth(x's set) <= d
- Consider the case that depth(x's set) = d+1. Prove that $|x\text{'s set}| >= 2^{d+1}$. *Analyse the operation which changed the depth of x's set to d+1*

Consider a set with depth d+1. Let x be its leader.

Q: How are the complexities affected?

- ... of MakeSet O(1)
- ... of Find O(lg n) Θ(log n)
- ... of Union Θ(log n)

Update depth?

Ex. generate a tight example for which Find take Θ(lg n)

∴ depth of any tree ≤ lg(n)

Union(x, y):

l_x = Find(x)

l_y = Find(y)

if ht(l_x) > ht(l_y):

  parent(l_y) = l_x

  return l_x

else if ht(l_x) ≤ ht(l_y)

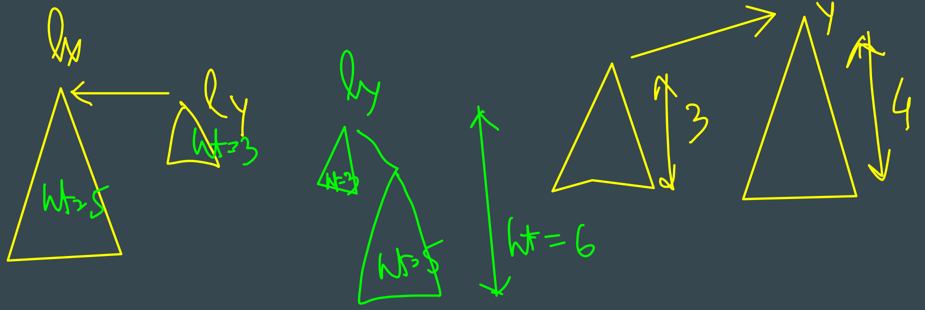
  if ht(l_x) = ht(l_y):

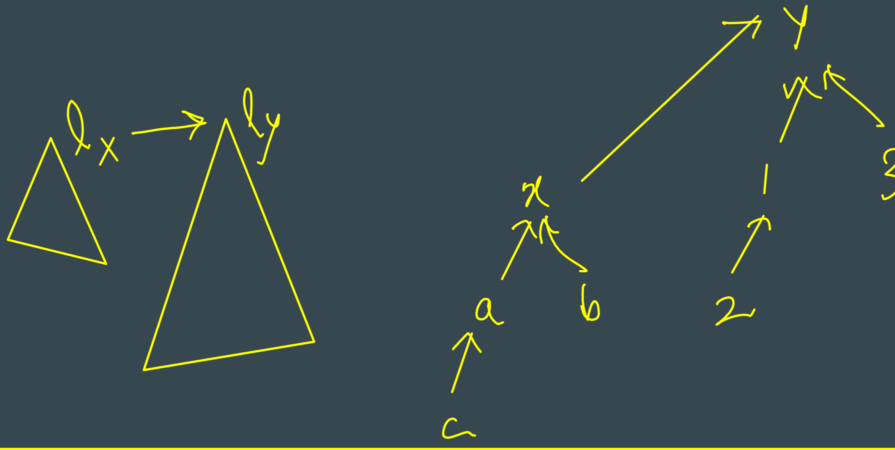
    ht(l_y) = ht(l_y) + 1

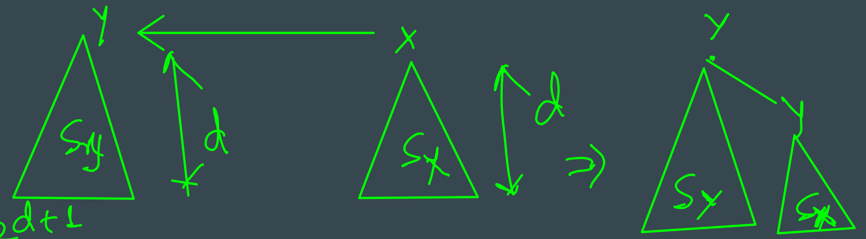
  parent(l_x) = l_y

  return l_y

Constant overhead







Set with x has depth d, by IH, $|S_x| \geq 2^d$

Set " y " " d, by IH, $|S_y| \geq 2^d$

$A = S_x \cup S_y$ $|A| = |S_x| + |S_y| \geq 2^{d+1}$