Disjoint sets with union

- a fixed set U is partitioned into disjoint subsets

- maintain these subsets under operations

  Create-Set(x)

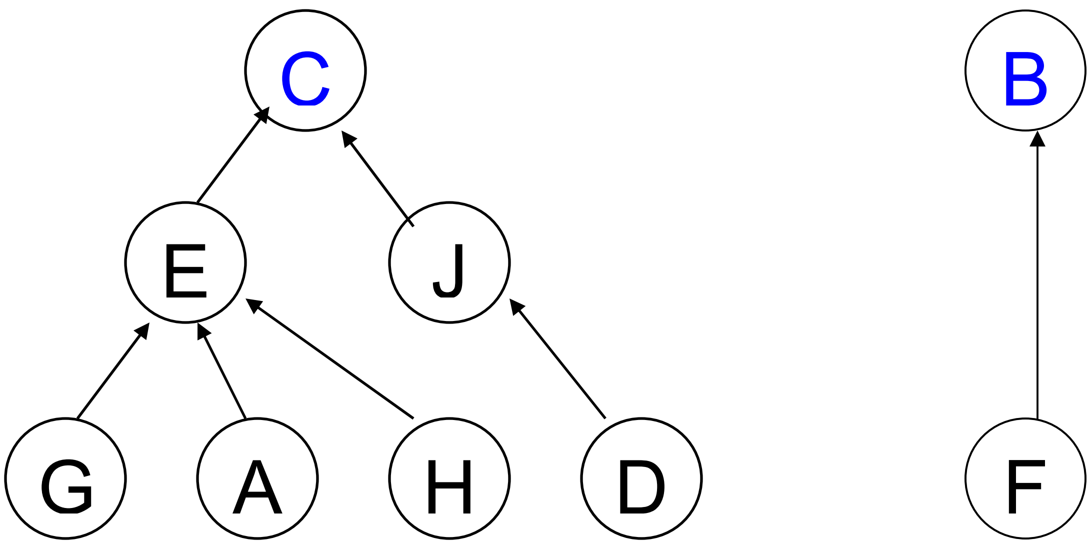
  Union(S, T)

  Find-set(x)

S,T sets. x an element

N.B. No Insert, Delete,DeleteMin, FindMin

Up-trees



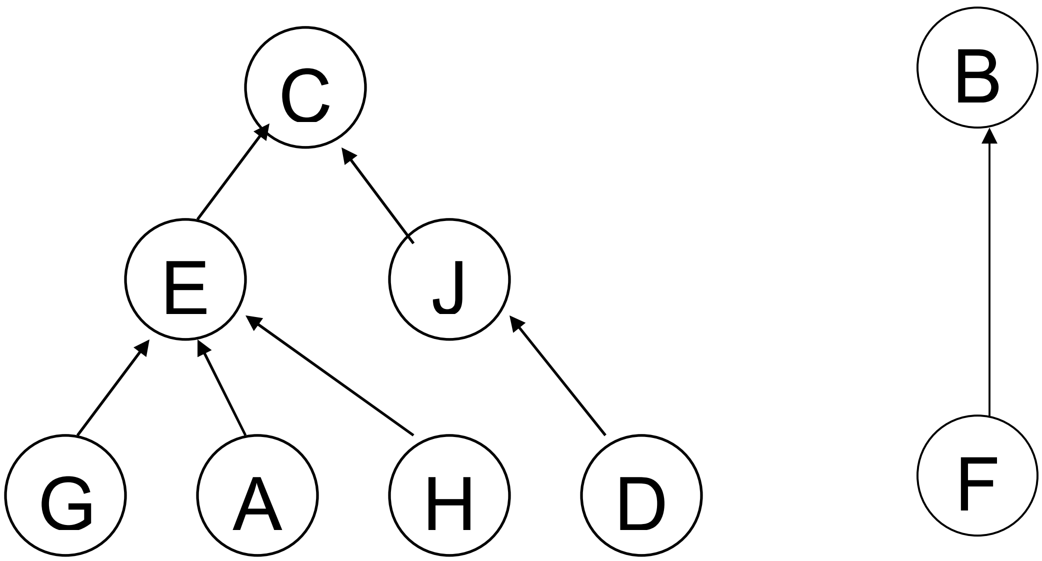
{ A, C, D, E, G, H, J }

Use C to denote this set

{ B, F }

Use B to denote this set

No limit on number of children



Find-set(x)

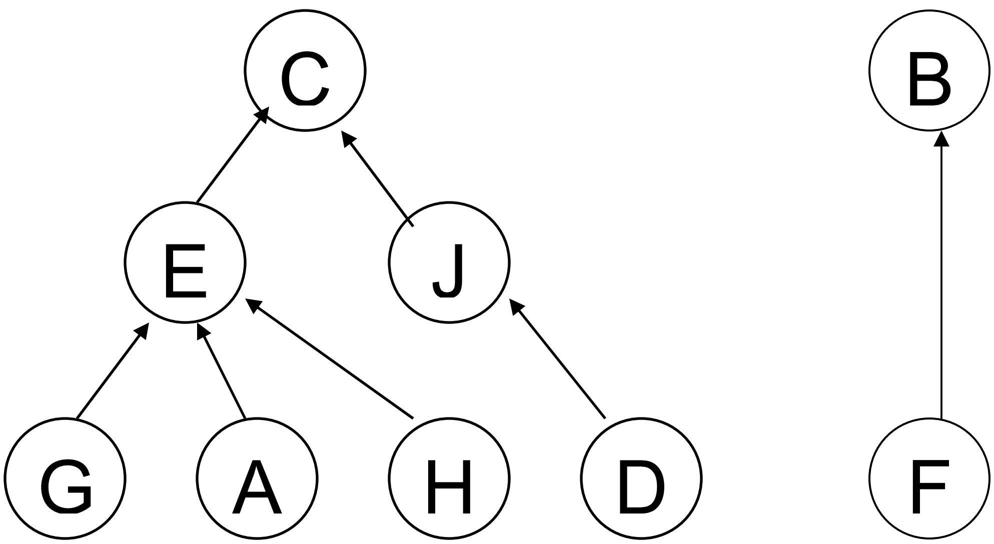
$z \leftarrow x$

loop if $z = parent[z]$

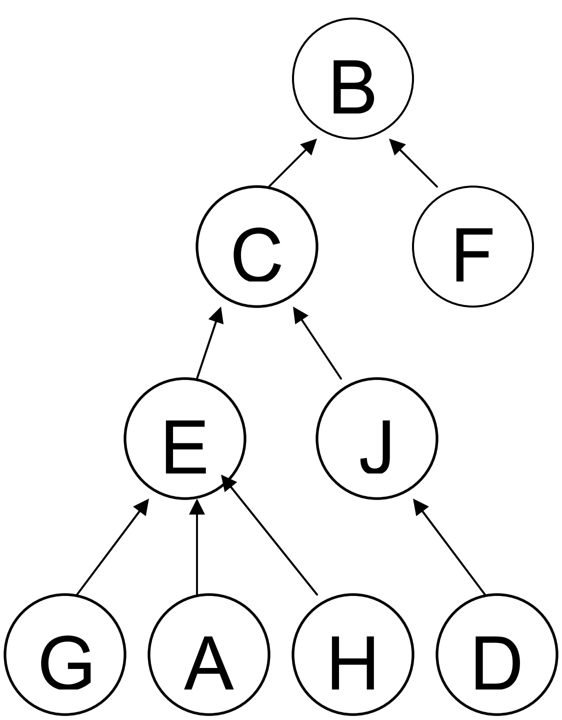
then return z

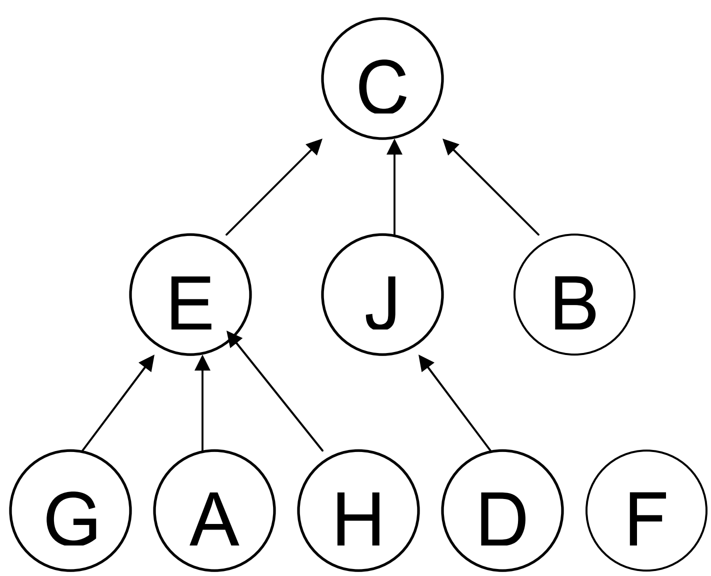
else $z \leftarrow parent[z]$

Find-Set (A)

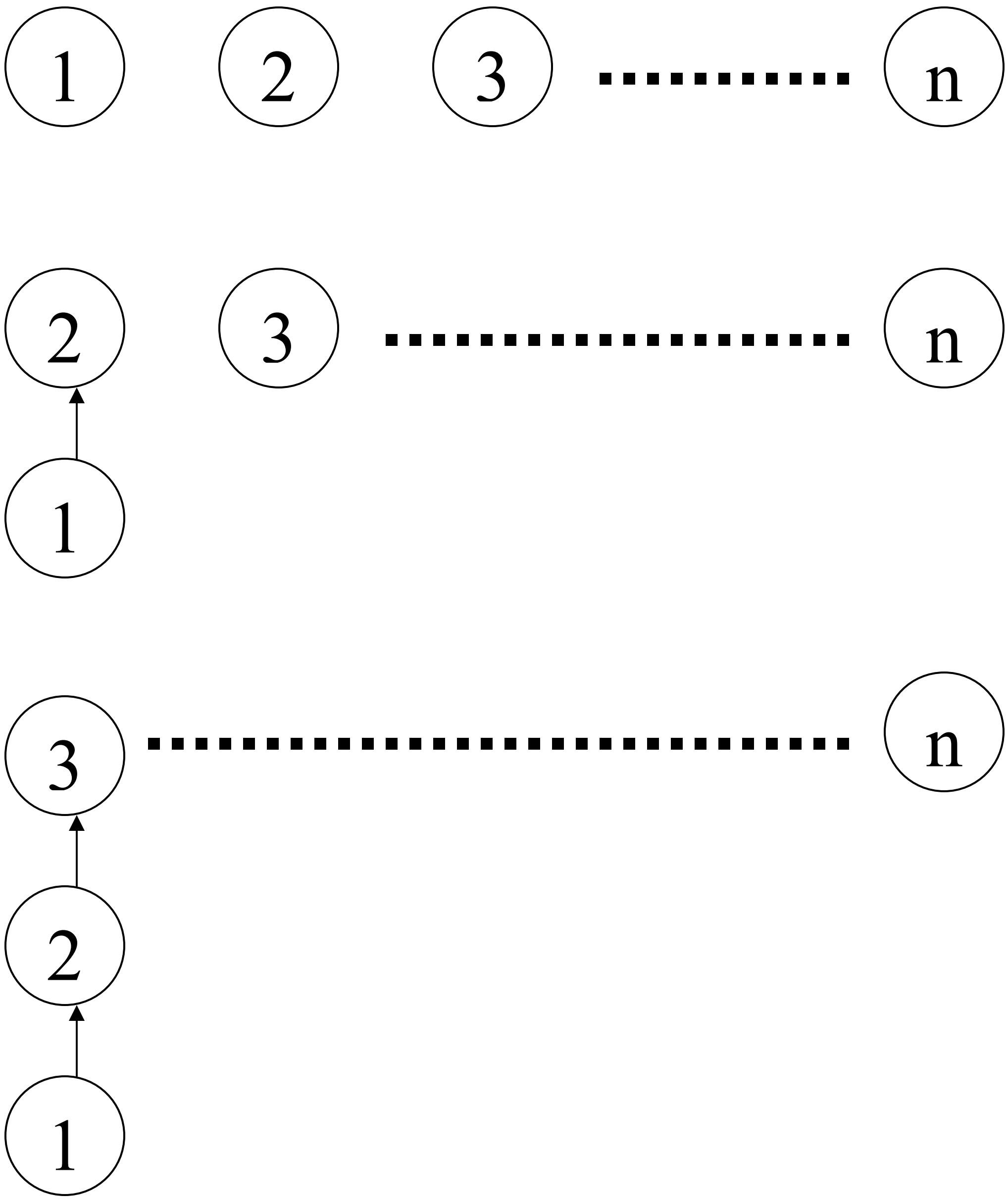


Union (C, B) *two possibilities*





Efficiency concern:



Possible to become a long single linked list.

Union by height

- the root of every tree holds the height of the tree.

- merge the shorter tree into the taller.

(make root of taller tree the parent of root of shorter tree)

(in case of ties, make root of first tree point to root of second)

CREATE-SET (x)

```
parent [x] ← x
height [x] ← 0
```

UNION (x, y)

```
if height [x] > height [y]
    then parent [y] ← x
    else parent [x] ← y
        If height [x] = height [y]
            Then height [y]++
```

LEMMA 1. For any root x,
$size(x) \geq 2^{height(x)}$

size(x): # descendants of x, including x

PROOF (by induction)
BASE CASE: At beginning, all heights are 0 and each tree has size 1.

INDUCTIVE STEP: Assume true just before a union(x, y).
DEF: size'(x) and height'(x) after union

CASE 1. height(x) < height(y)

Then size'(y) = size(x) + size(y)

$$\geq 2^{\text{height}(x)} + 2^{\text{height}(y)}$$

$$\geq 2^{\text{height}(y)}$$

$$= 2^{\text{height}'}$$

CASE 2. height(x) = height(y)

Then size'(y) = size(x) + size(y)

$$\geq 2^{\text{height}(x)} + 2^{\text{height}(y)}$$

$$= 2^{\text{height}(y)+1}$$

$$= 2^{\text{height}'(y)}$$

CASE 3. height(x) >height(y)

same as Case 1

COROLLARY

Every node has

height <= $\lfloor \lg n \rfloor$.

PROOF

Let h' > lg n.

There are at most

$n/2^{h'} < 1$ nodes of height h'.

$\Rightarrow$ There are zero nodes with height > lg n.

THM.

Create-Set(x) uses O(1) time

Union(x,y) uses O(1) time when
  x,y are roots of respective trees

Find-Set(x) uses O(log n) time

PROOF

Create-Set and Union are obviously O(1) time.

Find operation is O(h) where h is the max height of any tree.

By corollary, $h = \lfloor \lg n \rfloor$.

$\Rightarrow$ Find-Set(x) uses O(log n) time

Note:
Union(x,y) used by Kruskal's algorithm is actually the combination of three commands:

A = Find-Set(x)
B = Find-Set(y)
Union(A,B)

And therefore requires O(log n) time.

Note:
It is possible to improve the Union-Find data-structure so that it works even faster but that is beyond scope of this course. See CLRS for details.

Recall that running time of Kruskal's algorithm is dominated by the

$$O(|E| \log |E|)$$

sorting first stage so improving Union-Find won't speed up Kruskal's algorithm.