

Disjoint Sets (Union-Find Problem)

A disjoint set data structure supports the following operations. x and y are elements.

MAKE-SET(x) Creates a new set $\{x\}$. x must not be in any other set.

UNION(x, y) Combine the set that contains x with the set that contains y .

FIND-SET(x) **FIND-SET**(x) = **FIND-SET**(y) when x and y are in the same set.

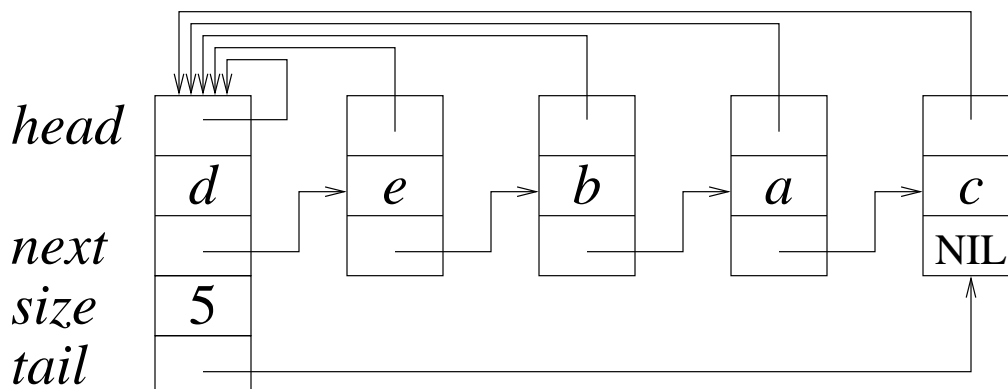
Example: Connected components of an undirected graph

```

CONNECTED-COMPONENTS( $G$ )
  for each vertex  $v$  in graph  $G$ 
    do MAKE-SET( $v$ )
  for each edge  $(u, v)$  in graph  $G$ 
    do if FIND-SET( $u$ )  $\neq$  FIND-SET( $v$ )
      then UNION( $u, v$ )
  
```

Linked List Representation

Assume x has fields $head$, $tail$, $next$, and $size$.



MAKE-SET(x)

$head[x] \leftarrow x$

$tail[x] \leftarrow x$

$next[x] \leftarrow NIL$

$size[x] \leftarrow 1$

FIND-SET(x)

return $head[x]$

UNION(x, y)

$x \leftarrow \text{FIND-SET}(x)$

$y \leftarrow \text{FIND-SET}(y)$

if $size[x] < size[y]$

then exchange $x \leftrightarrow y$

$next[tail[x]] \leftarrow y$

$tail[x] \leftarrow tail[y]$

$size[x] \leftarrow size[x] + size[y]$

while $y \neq NIL$

do $head[y] \leftarrow x$

$y \leftarrow next[y]$

Amortized Analysis of Linked List Rep.

Assume m operations including n MAKE-SETS.

Analyze number of assignments to *head* field.

Using accounting method of amortized analysis:

Use amortized cost $1 + \lg n$ units/MAKE-SET.

Consider an arbitrary element x .

Use one unit immediately for MAKE-SET(x).

Use one unit each time UNION modifies $head[x]$.

UNION changes *head* fields of smaller set, so a change to $head[x]$ at least doubles x 's set size.

The size of a set cannot exceed $n = 2^{\lg n}$, so the cost $\lg n + 1$ covers all changes to $head[x]$.

Over n elements, the total cost is $n(\lg n + 1)$.

There can be $O(m)$ FIND-SET operations, so total time is $O(m + n \lg n)$.

It is possible that $n - 1$ UNIONS make $(n/2) \lg n$ assignments to *head* fields so $O(m + n \lg n)$ is tight.

Disjoint-Set Forests

Assume x has fields p and r (parent and rank).

MAKE-SET(x)

$p[x] \leftarrow x$

$r[x] \leftarrow 0$

FIND-SET(x)

if $x \neq p[x]$

then

$p[x] \leftarrow$

FIND-SET($p[x]$)

return $p[x]$

UNION(x, y)

$x \leftarrow$ **FIND-SET**(x)

$y \leftarrow$ **FIND-SET**(y)

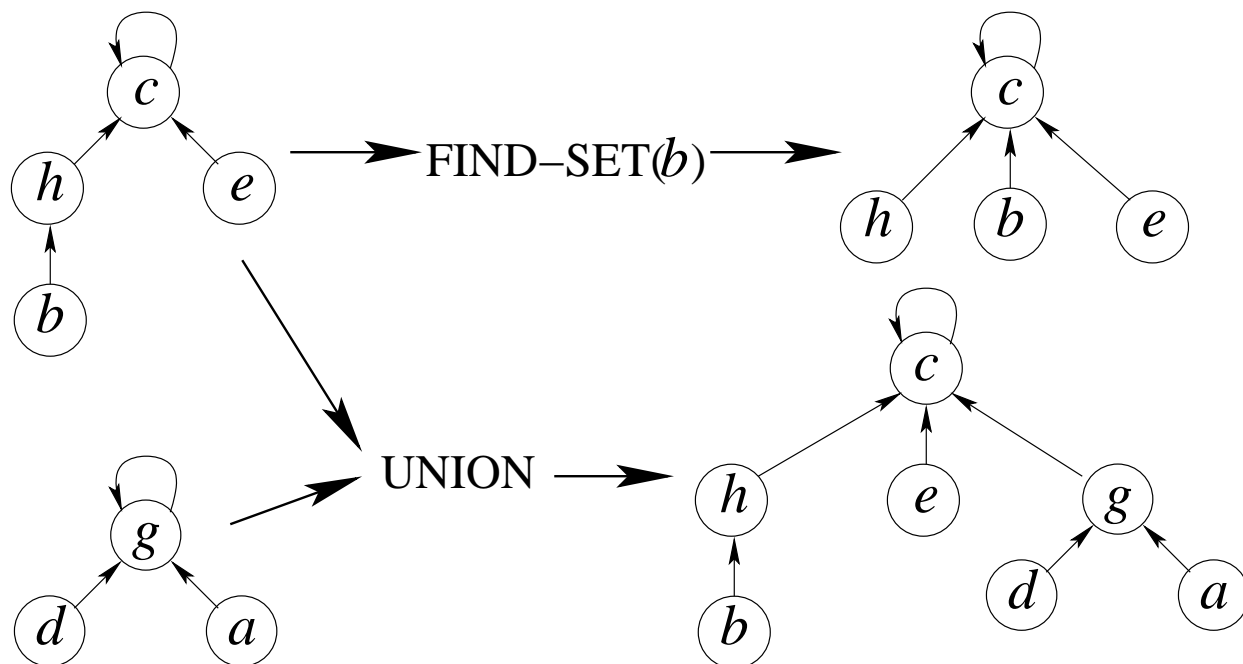
if $r[x] > r[y]$

then $p[y] \leftarrow x$

else $p[x] \leftarrow y$

if $r[x] = r[y]$

then $r[y] \leftarrow r[y] + 1$



The rank bounds the height of the tree:

Basis: When $r = 0$, then $h = 0$.

Assume: When $r = k$, then $h \leq k$.

Show: When $r = k + 1$, $h \leq k + 1$.

Induction: h can increase only when
combining two trees with same r .

A tree with rank r has $\geq 2^r$ nodes.

Basis: When $r = 0$, there is $2^0 = 1$ node.

Assume: When $r = k$, there are $\geq 2^k$ nodes.

Show: When $r = k + 1$, there are $\geq 2^{k+1}$ nodes.

Induction: r increases only when combining
two trees with same r . Their union
must have $\geq 2(2^k) = 2^{k+1}$ nodes.

Without considering compression, this implies
that **FIND-SET** will traverse $\leq \lg n$ links/call.

Amortized analysis for $O(\lg \lg n)$ (not in book)

Let f be the number of **FIND-SETS** excluding recursion.

Let l_1, \dots, l_f be the number of links compressed by calls to **FIND-SET**. Want to bound $\sum_{i=1}^f l_i$. Why?

If a call to **FIND-SET** compresses l links, then $\geq 2^{l-1}$ nodes are closer to the root.

Proof: A subtree of rank $\geq l - 1$ now points to the root. This subtree has $\geq 2^{l-1}$ nodes.

Let l be the average of l_1, \dots, l_f .

The number of recursive calls is fl .

It can be shown that $\sum_{i=1}^f 2^{l_i-1} \geq f2^{l-1}$.

For example, note that $2^{l-1} + 2^{l+1} > 2^l + 2^l$

There is $\leq n \lg n$ “distance” to compress because each node is $\leq \lg n$ away from the root.

$\leq n \lg n$ distance to compress $\rightarrow f2^{l-1} \leq n \lg n$.

Case 1: $f \leq n/(\lg n)$

$f \leq n/(\lg n)$ and $l \leq \lg n$ imply $fl \leq n$.

Case 2: $f \geq n/(\lg n)$

Start with $f2^{l-1} \leq n \lg n$

Taking logarithms

$$(\lg f) + l - 1 \leq (\lg n) + (\lg \lg n)$$

$$\begin{aligned} l - 1 &\leq (\lg n) + (\lg \lg n) - (\lg f) \\ &\leq (\lg n) + (\lg \lg n) - \lg(n/\lg n) \\ &= (\lg n) + (\lg \lg n) - (\lg n) + (\lg \lg n) \\ &= 2 \lg \lg n. \end{aligned}$$

so l is $O(\lg \lg n)$, which implies that the total cost of f FIND-SETS is $O(n + f \lg \lg n)$.