Disjoint Set Union

Maintain a collection of disjoint sets under find and union.

Initially set $i = \{i\}$ for $1 \le i \le n$

find (x): Return the name of the set containing element x

union (A, B) Replace set A by the union of sets A and B, destroying the old sets.

Name each set by some (arbitrary) element in it.

Applications:

FORTRAN EQUIVALENCE and COMMON statements

computing minimum spanning trees

many other graph algorithms

unification

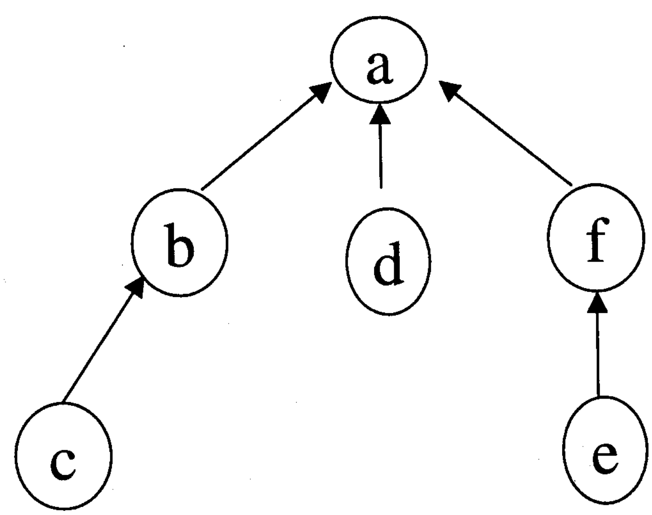
Tree Data Structure

Represent each set by a tree whose nodes are the elements of the set.

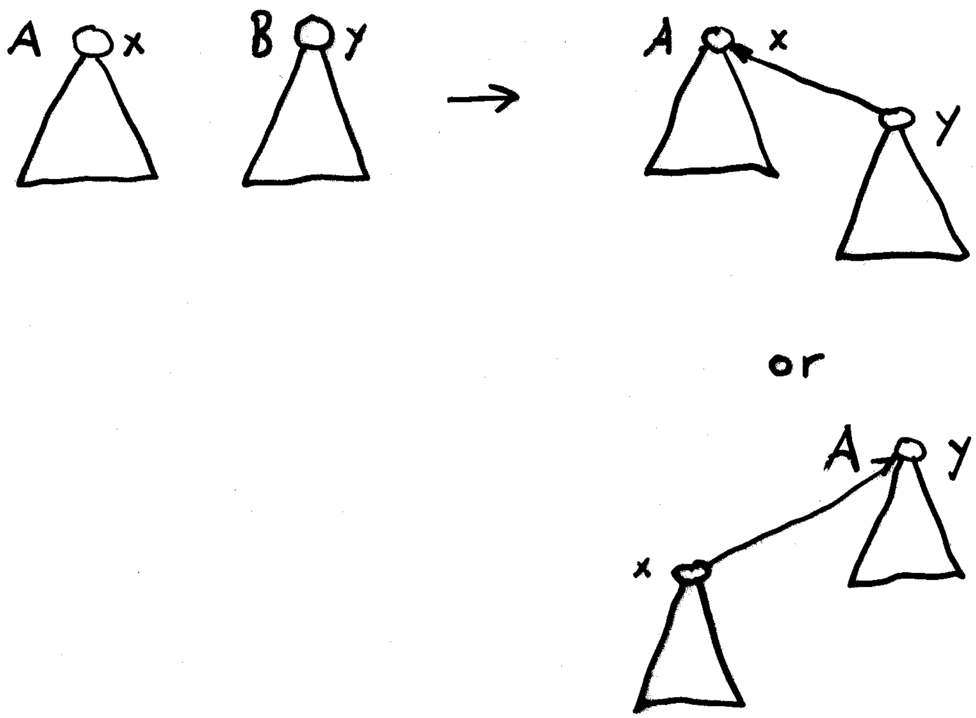
The root is the set name (can store any info at the root)

Each node points to its parent. (M. Fischer and Galler)

{a,b,c,d,e,f}



Union



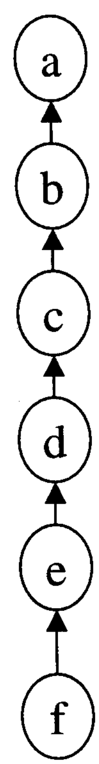
find(x): Follow parent pointers from x to root,
return root.

cost = # nodes on find path

unite(x,y): Make root x the parent of root y (x is the name of the new set)

cost = 1

Total cost = Θ (mn)



Basis for improvement: the structure of each tree is arbitrary; only the partition defined by the node sets matters.

Union by size: maintain at each root the tree size (# nodes).

unite (x,y): if size (x) ≥ size (y) make x the parent of y
if size (x) ≤ size (y) make y the parent of x
(McIlroy)

Union by rank: maintain a <u>rank</u> at each root, initially 0.

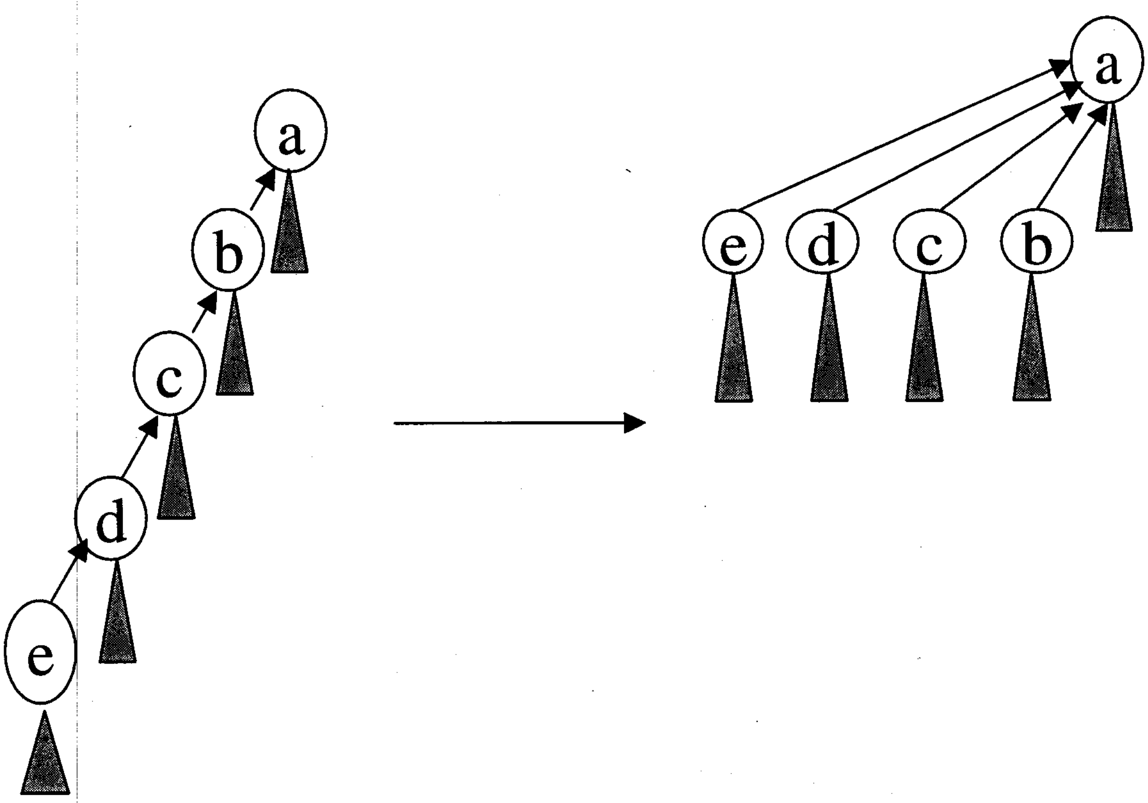
unite (x,y): if rank (x) > rank (y) make x the parent of y
if rank (x) < rank (y) make y the parent of x
if rank (x) = rank (y) make x the parent of y and increase the rank of x by 1

rank = tree height

With either union by size or union by rank, the height of a k-node tree is $\leq \lg k$

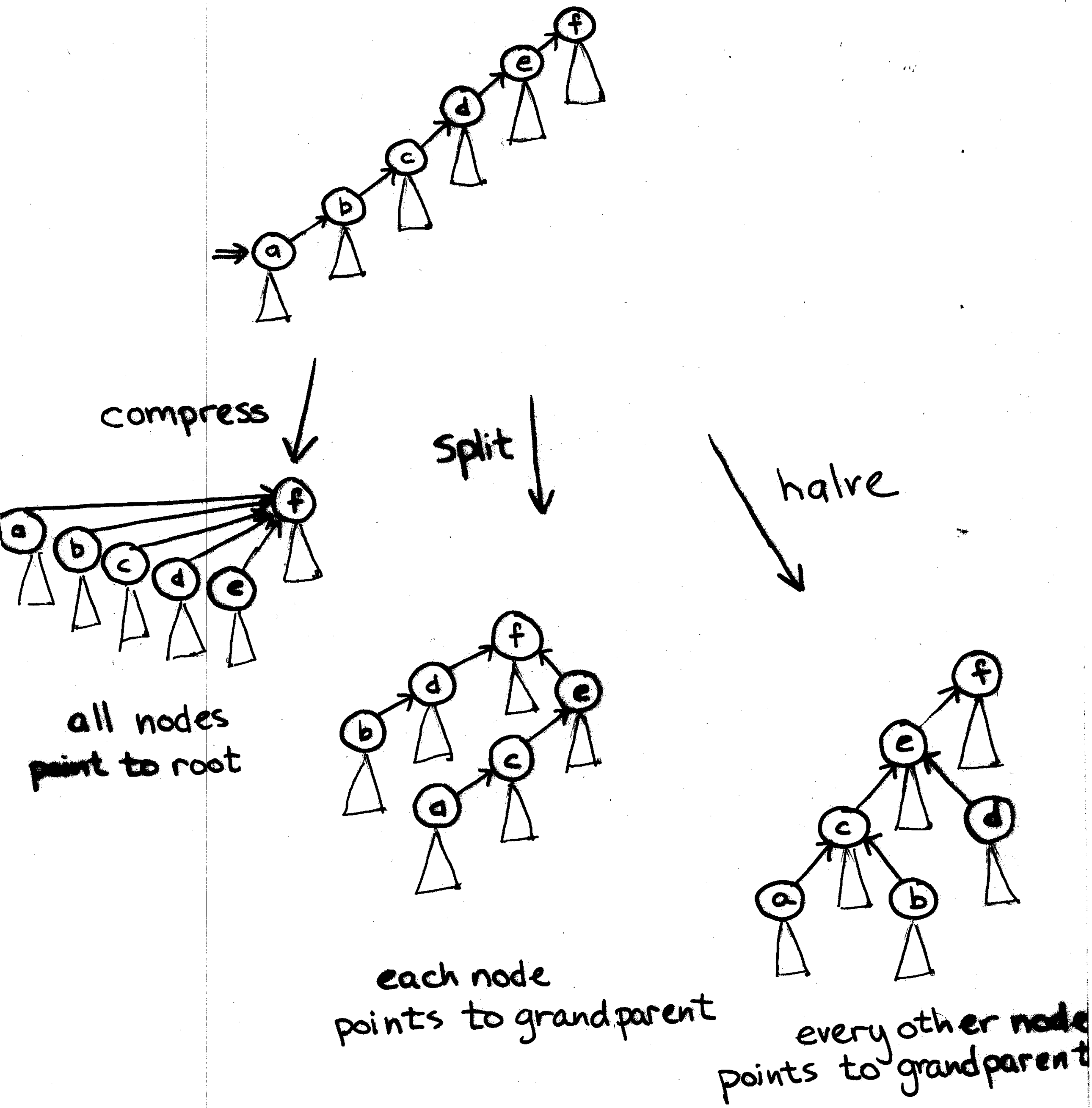
$\Rightarrow$ total cost = $\Theta(m \log n)$

Path Compression: after a find, make each node along the find path a child of the root (Tritter)



Compressions shorten paths and thus make later finds cheaper (but cost a constant factor)

Find with Compression



History of Bounds (early 70's)

$O(m)$ (bogus)

$O(m \log \log n)$ M. Fischer

$O(m \log^* n)$ Hopcroft and Ullman

$\Omega(n \log \log n)$(bogus)

$\Omega(n\alpha(n))$Tarjan

$O(m\alpha(n))$Tarjan

How efficient is path compression?

Without union by size or rank, O(mlogn), tight for m=n

With union by size or rank, O(m α (n)), tight for m=n,

where α is an inverse of Ackerman's function:

for k≥0, j≥1

$$A_0(j) = j+1, A_k(j) = A_{k-1}^{(j+1)}(j) \text{ if } k\geq 1$$

where $f^0(x) = x, f^{(i+1)}(x) = f(f^{(i)}(x))$

$$\alpha(n) = \min\left\{k : A(1) \geq n\right\}$$

$A_k(x)$ strictly increases in both k and x

$A_1(x) = 2x+1$

$A_2(x) > 2^x$

$A_3(x) > \left. 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \right\} x+1 \text{ 2's}$

$\alpha(n)$ grows very slowly

Union by rank

Fix x

$r(x)$ starts at 0, increases while x is a root,

then fixed

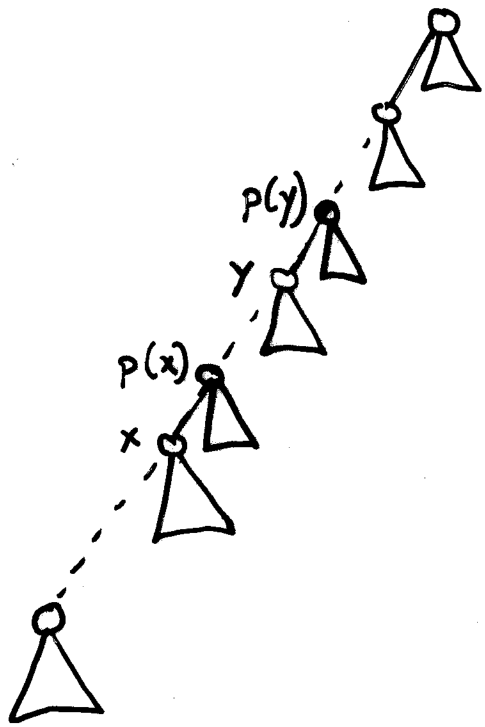
$r(p(x)) > r(x)$

$r(p(x))$ only increases: $p(x)$ changes by

a compression, or $r(p(x))$ changes

by a link

$r(x) \leq n-1$ (actually $\leq \lg n$, not used)



After compression, the new parent of x (root) has rank at least $r(p(y))$

x a nonroot with $r(x) \geq 1$

<u>level</u> of $x = k(x) = \max\{k : r(p(x)) \geq A_k(r(x))\}$

<u>index</u> of $x = i(x) = \max\{i : r(p(x)) \geq A_{k(x)}^{(i)}(r(x))\}$

$0 \leq k(x) < \alpha(n)$:

$$A_0(r(x)) = r(x) + 1 \leq r(p(x))$$

$$A_{\alpha(n)}(r(x)) \geq A_{\alpha(n)}(1) \geq n > n-1 \geq r(p(x))$$

$1 \leq i(x) \leq r(x)$:

$$A_{k(x)}^{1}(r(x)) = A_{k(x)}(r(x)) \leq r(p(x)) \qquad \text{dfn } k(x)$$

$$A_{k(x)}^{r(x)+1}(r(x)) = A_{k(x)+1}(r(x)) \qquad \text{dfn } A$$

$$> r(p(x)) \qquad \text{dfn } k(x)$$

$k(x)$ increases since $r(p(x))$ increases

$i(x)$ increases while $k(x)$ is fixed,

can decrease when $k(x)$ increases

$$\phi(x) = \begin{cases} \alpha(n)\, r(x) & \text{if } x \text{ a root or } r(x)=0 \\ (\alpha(n) - k(x))\, r(x) - i(x) & \text{otherwise} \end{cases}$$

$$\Phi = \sum_x \phi(x)$$

$0 \le \phi(x) \le \alpha(n)\, r(x)$ for all x:

true if x a root or $r(x) = 0$

otherwise:

$k(x) \le \alpha(n) - 1$ and $i(x) \le r(x)$

$$\Rightarrow \phi(x) \ge r(x) - i(x) \ge 0$$

$k(x) \ge 0$ and $i(x) \ge 1$

$$\Rightarrow \phi(x) \le \alpha(n)\, r(x) - 1$$

while x is a root, $\phi(x)$ only increases;

while x is a nonroot, $\phi(x)$ only decreases

Amortized time per operation is $O(\alpha(n))$:

link (x,y) with y the new root:

actual time $= O(1)$

$\Delta\phi(z) \le 0$ if $z \ne x, z \ne y$

$\Delta\phi(x) \le 0$

$\Delta\phi(y) \le \alpha(n)$: $r(y)$ increases by at most 1

$\Rightarrow$ amortized time $\le \alpha(n) + O(1)$

find with compression:

actual cost: #nodes on find path $= \ell$

x on find path $\Rightarrow \Delta\phi(x) \leq 0$

At least $\ell - (\alpha(n) + 2)$ nodes x on path

have $\Delta\phi(x) \leq -1$

$\Rightarrow$ amortized find cost $\leq \alpha(n) + 2$

$(= \ell - (\ell - (\alpha(n) + 2)))$

Let x be on path with $r(x) > 0$ and some

y after x on path has $k(y) = k(x)$

(all but $\alpha(n)+2$ nodes on path:

first, last (root), last in each level)

Let $k = k(x) = k(y)$

before compress:

$$r(y) \geq r(p(x)) \geq A_k^{(i(x))}(r(x))$$

$$r(p(y)) \geq A_k(r(y)) \geq A_k(r(p(x)))$$

$$\geq A_k(A_k^{(i(x))}(r(x))) = A_k^{i(x)+1}(r(x))$$

$\Rightarrow$ $i(x)$ or $k(x)$ increases due to compress

$\Rightarrow$ $\phi(x)$ decreases