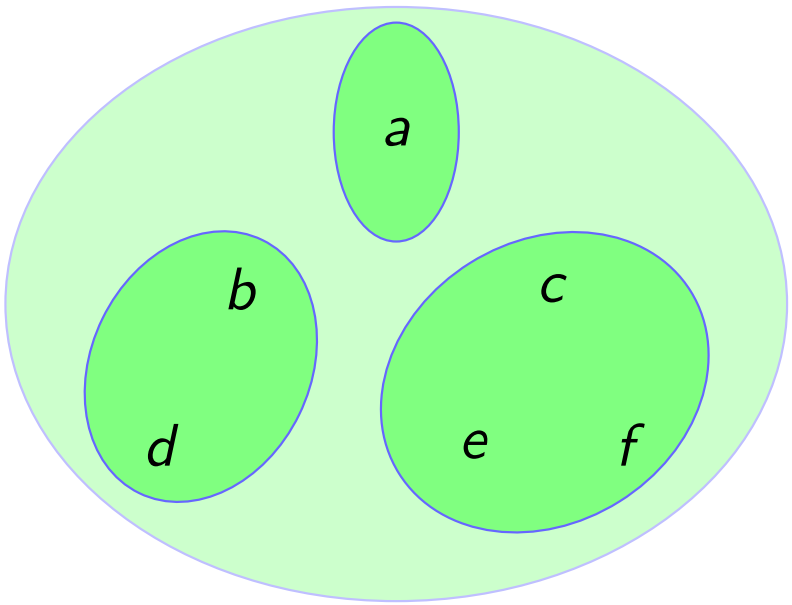
Disjoint Sets



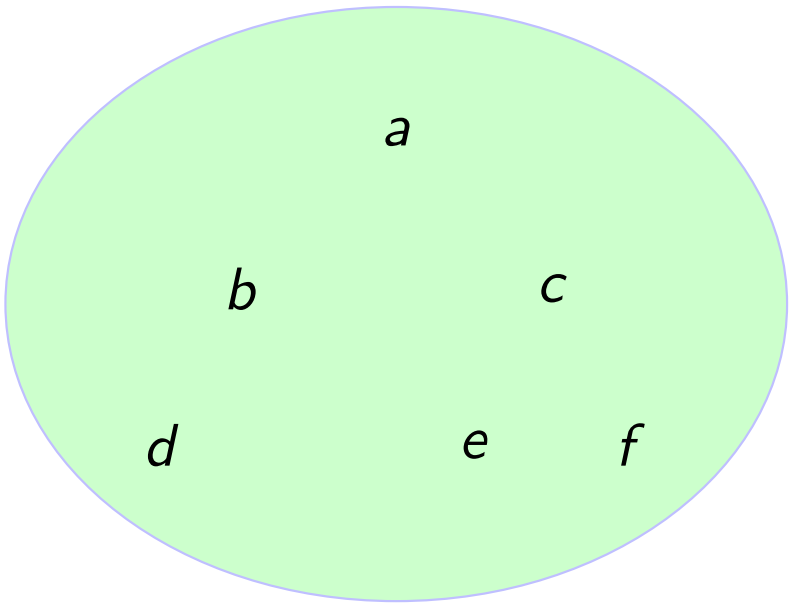
set of sets

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

equivalence relation

$$\begin{array}{c} \\ a \\ b \\ d \\ c \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & d & c & e & f \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

Disjoint Set Operations



$\longrightarrow$

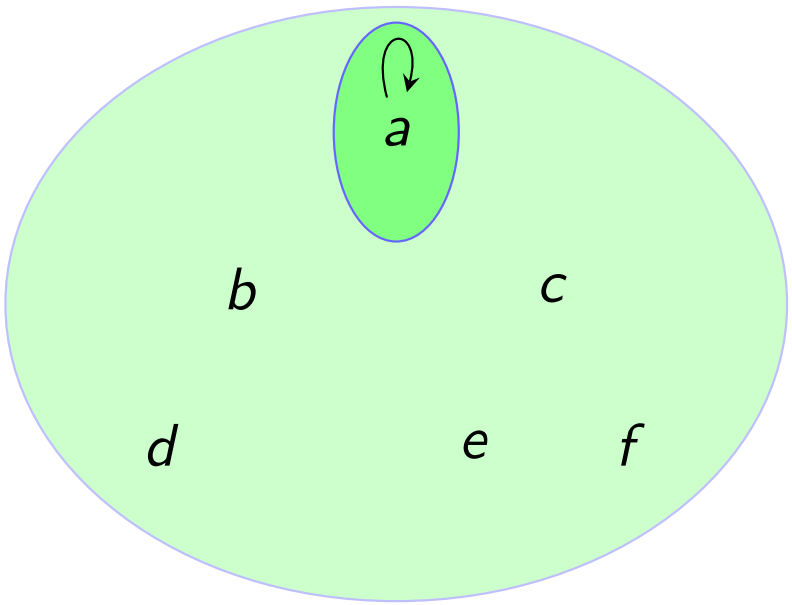
make-set(a)
make-set(b)
make-set(c)
make-set(d)
make-set(e)
make-set(f)

find-set(f) == f

union-sets(f, c)
find-set(f) == c

union-sets(d, b)
union-sets(f, e)
find-set(f) == e

Disjoint Set Operations



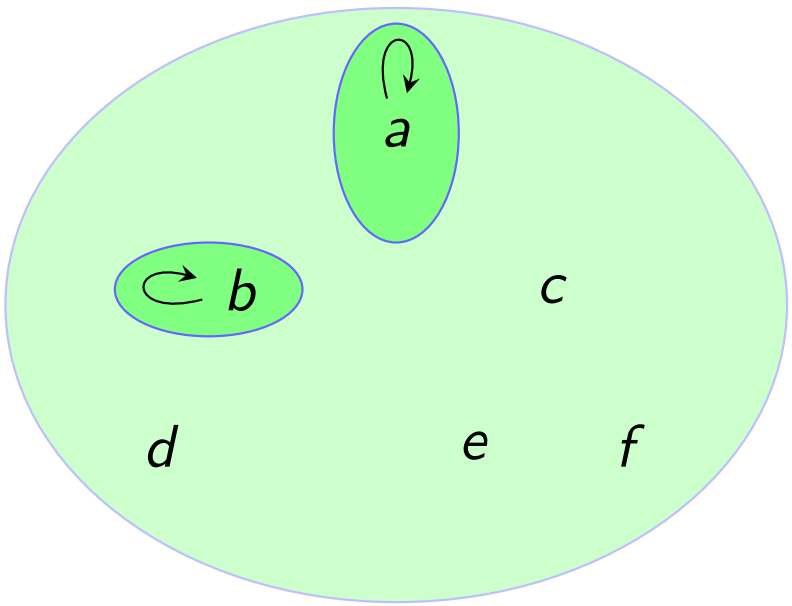
→ make-set(a)
make-set(b)
make-set(c)
make-set(d)
make-set(e)
make-set(f)

find-set(f) == f

union-sets(f, c)
find-set(f) == c

union-sets(d, b)
union-sets(f, e)
find-set(f) == e

Disjoint Set Operations



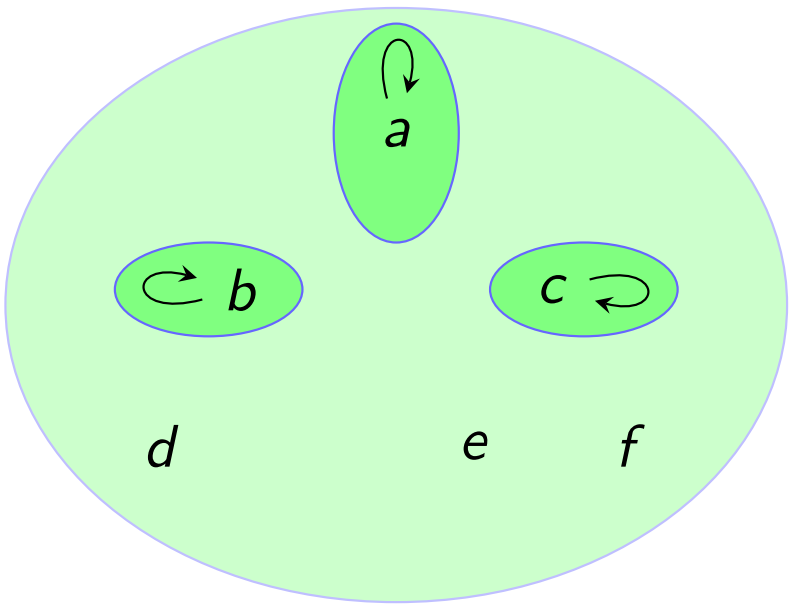
make-set(a)
make-set(b)
make-set(c)
make-set(d)
make-set(e)
make-set(f)

find-set(f) == f

union-sets(f, c)
find-set(f) == c

union-sets(d, b)
union-sets(f, e)
find-set(f) == e

Disjoint Set Operations



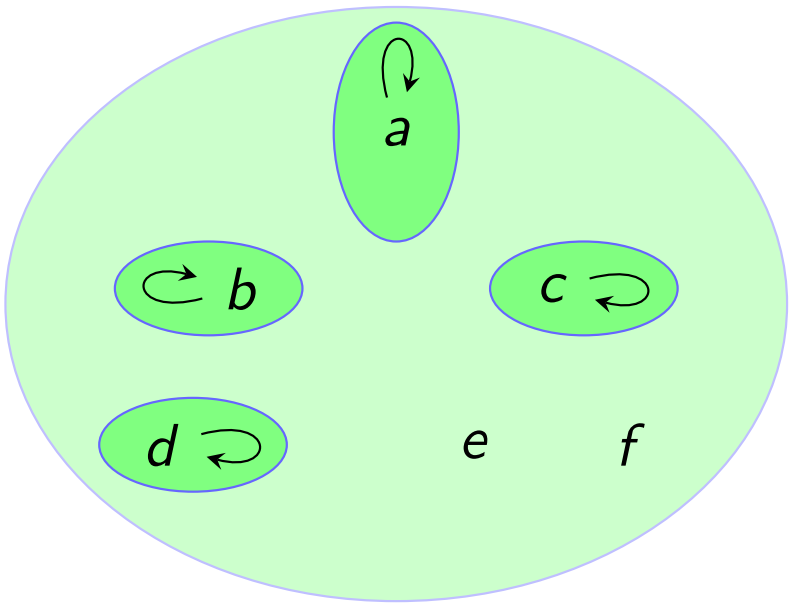
make-set(a)
make-set(b)
make-set(c)
make-set(d)
make-set(e)
make-set(f)

find-set(f) == f

union-sets(f, c)
find-set(f) == c

union-sets(d, b)
union-sets(f, e)
find-set(f) == e

Disjoint Set Operations



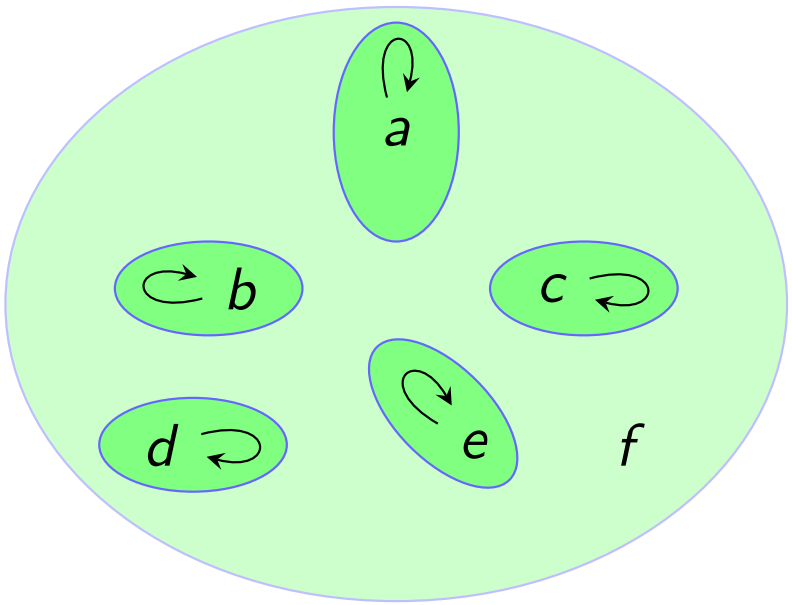
make-set(a)
make-set(b)
make-set(c)
make-set(d)
make-set(e)
make-set(f)

find-set(f) == f

union-sets(f, c)
find-set(f) == c

union-sets(d, b)
union-sets(f, e)
find-set(f) == e

Disjoint Set Operations



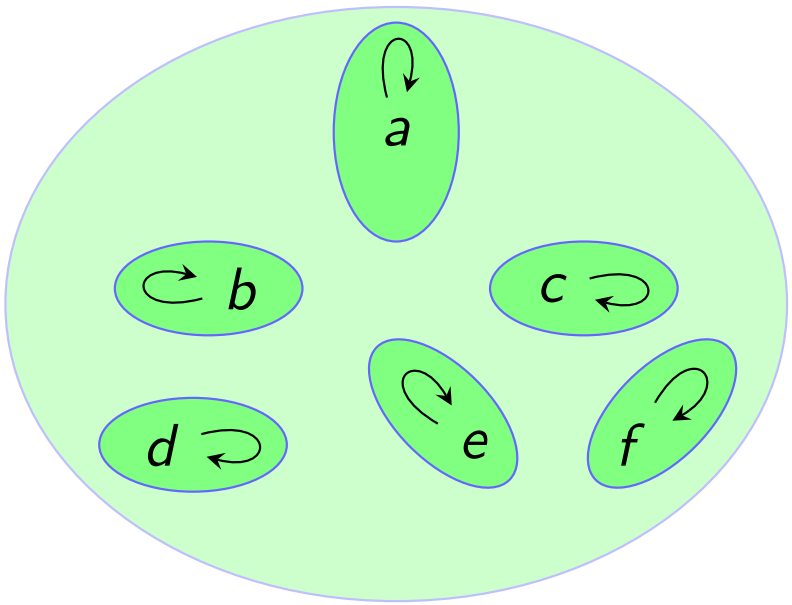
make-set(a)
make-set(b)
make-set(c)
make-set(d)
make-set(e)
make-set(f)

find-set(f) == f

union-sets(f, c)
find-set(f) == c

union-sets(d, b)
union-sets(f, e)
find-set(f) == e

Disjoint Set Operations



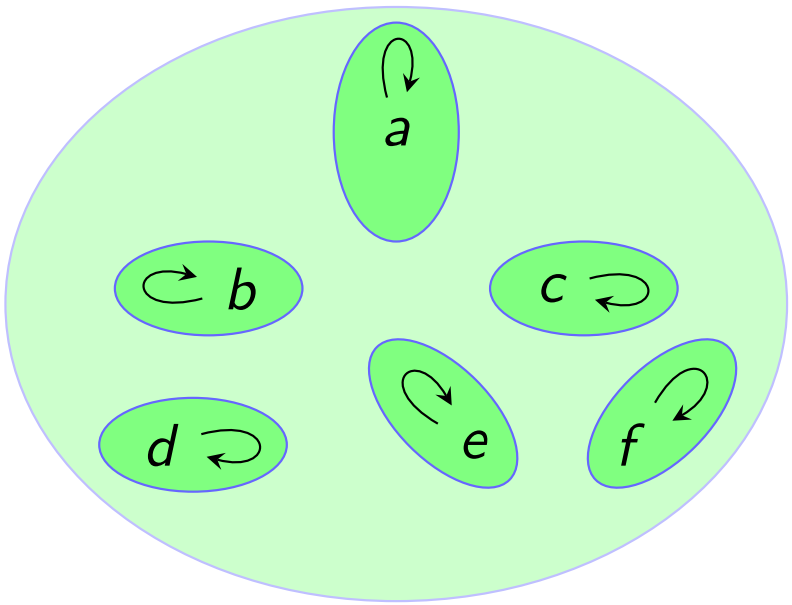
make-set(a)
make-set(b)
make-set(c)
make-set(d)
make-set(e)
make-set(f)

find-set(f) == f

union-sets(f, c)
find-set(f) == c

union-sets(d, b)
union-sets(f, e)
find-set(f) == e

Disjoint Set Operations



make-set(a)
make-set(b)
make-set(c)
make-set(d)
make-set(e)
make-set(f)

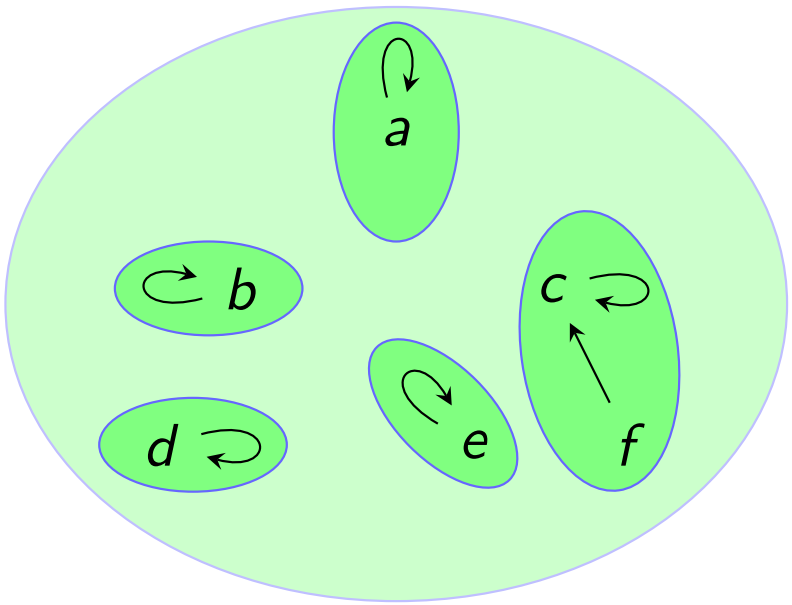
find-set(f) == f

→

union-sets(f, c)
find-set(f) == c

union-sets(d, b)
union-sets(f, e)
find-set(f) == e

Disjoint Set Operations



make-set(a)
make-set(b)
make-set(c)
make-set(d)
make-set(e)
make-set(f)

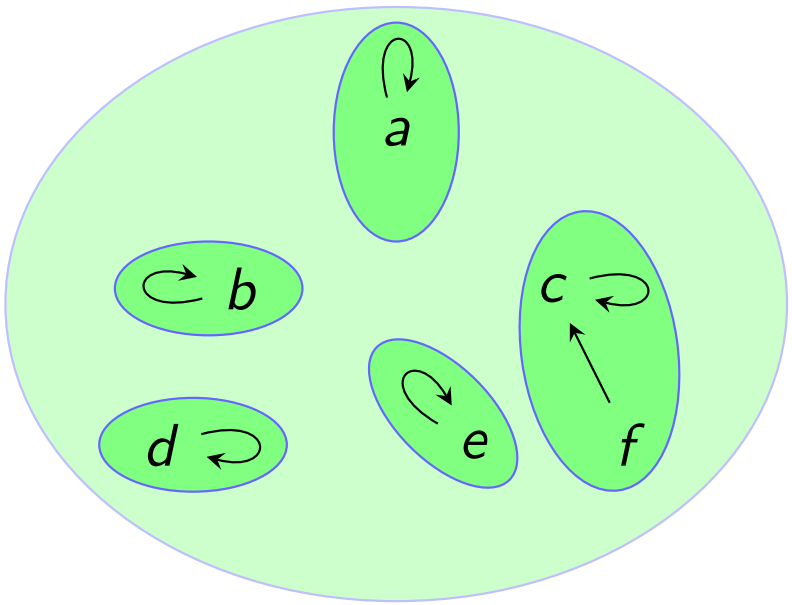
find-set(f) == f

union-sets(f, c)

→ find-set(f) == c

union-sets(d, b)
union-sets(f, e)
find-set(f) == e

Disjoint Set Operations



make-set(a)
make-set(b)
make-set(c)
make-set(d)
make-set(e)
make-set(f)

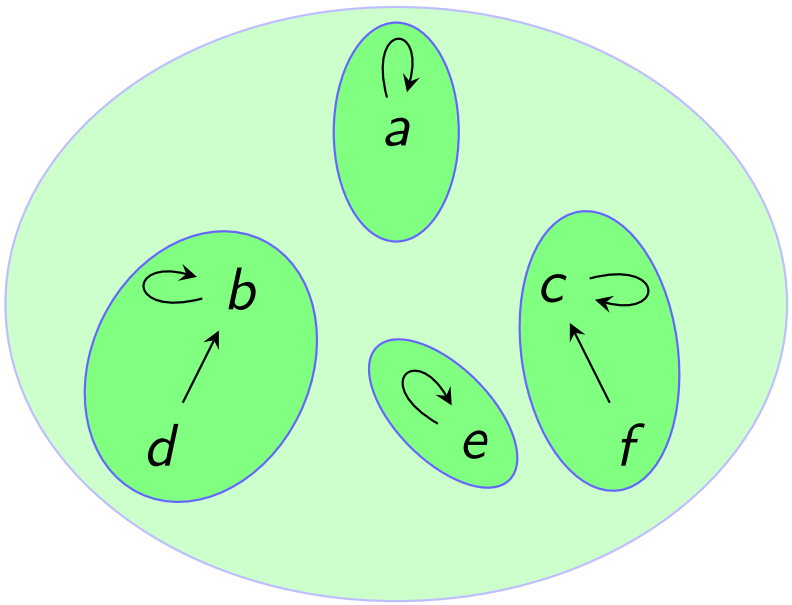
find-set(f) == f

union-sets(f, c)
find-set(f) == c

→

union-sets(d, b)
union-sets(f, e)
find-set(f) == e

Disjoint Set Operations



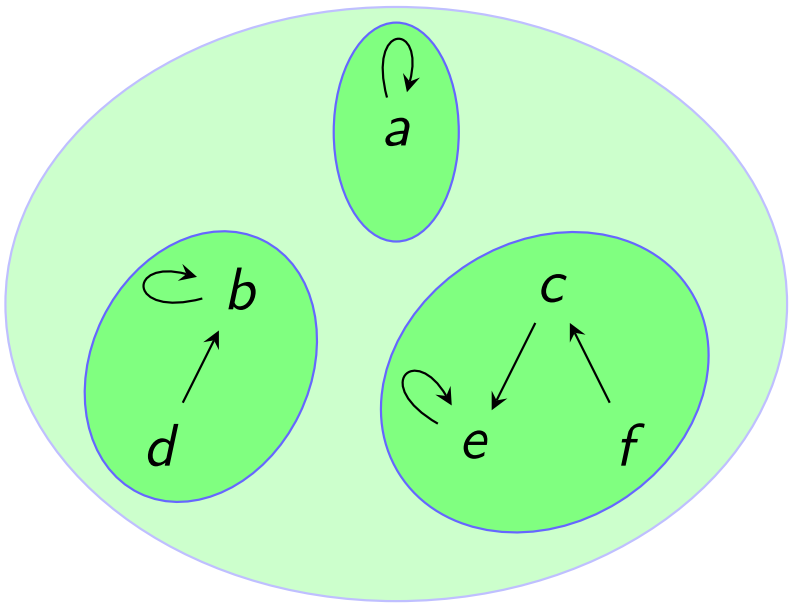
make-set(a)
make-set(b)
make-set(c)
make-set(d)
make-set(e)
make-set(f)

find-set(f) == f

union-sets(f, c)
find-set(f) == c

union-sets(d, b)
union-sets(f, e)
find-set(f) == e

Disjoint Set Operations



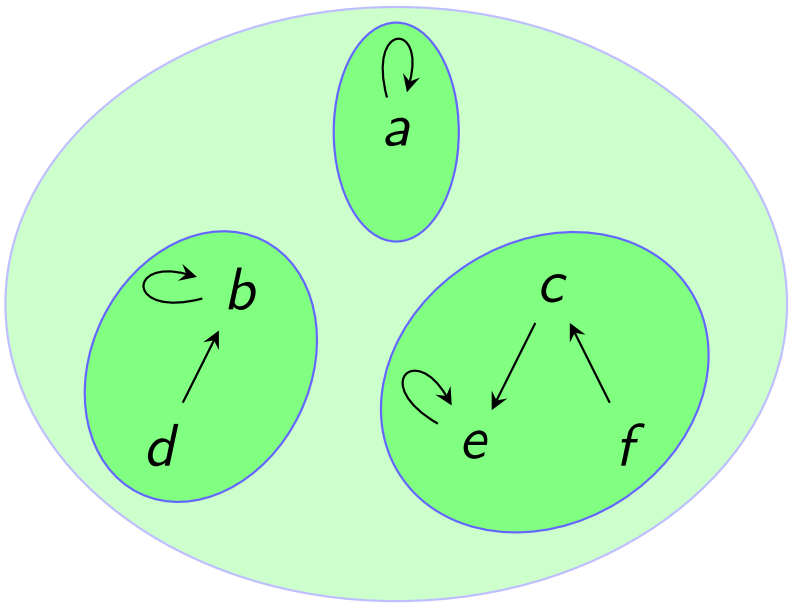
make-set(a)
make-set(b)
make-set(c)
make-set(d)
make-set(e)
make-set(f)

find-set(f) == f

union-sets(f, c)
find-set(f) == c

union-sets(d, b)
union-sets(f, e)
→ find-set(f) == e

Disjoint Set Operations



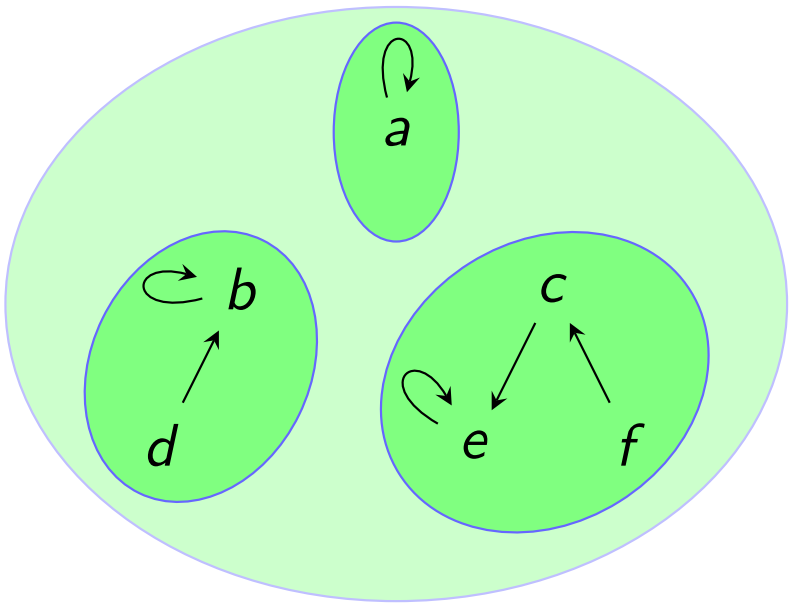
make-set(a)
make-set(b)
make-set(c)
make-set(d)
make-set(e)
make-set(f)

find-set(f) == f

union-sets(f, c)
find-set(f) == c

union-sets(d, b)
union-sets(f, e)
find-set(f) == e

Disjoint Set Forests



tree = equivalence class

root = representative

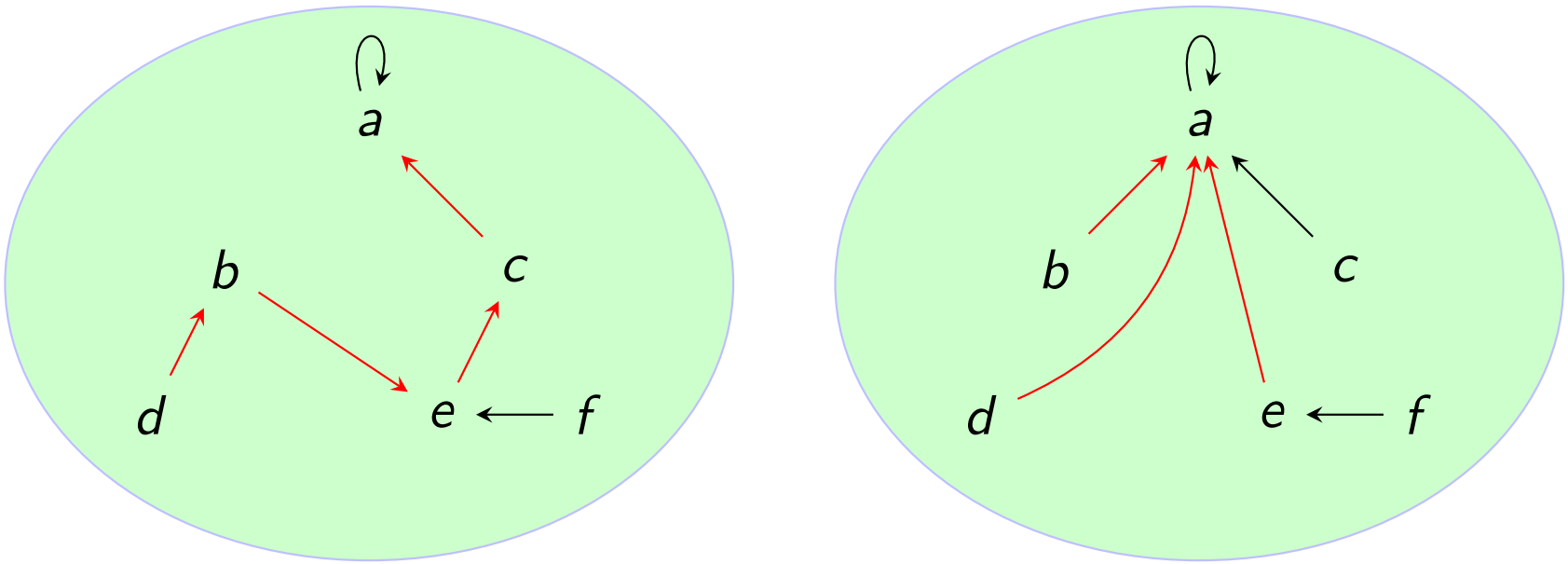
parent array

$$\begin{bmatrix} a & b & c & d & e & f \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ a & b & e & b & e & c \end{bmatrix}$$

parent relation

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Path Compression



find-set(d) == a

Implementation

make-set(x):
    $p[x] = x$
    $rank[x] = 0$

find-set(x):
    if $x \neq p[x]$:
        $p[x] =$ find-set($p[x]$)
    return $p[x]$

union-sets(x, y):
    $r =$ find-set(x)
    $s =$ find-set(y)
    if $rank[r] > rank[s]$:
        $p[s] = r$
    else:
        $p[r] = s$
        if $rank[r] == rank[s]$:
            $rank[s] \mathrel{+}= 1$

– union by rank makes smaller tree a subtree of larger tree

– $rank[x]$ is an upper bound on the depth of tree with root x

– time complexity almost linear in the total number of operations