Chapter 4
Data Structures
Union Find

Algorithm Theory
WS 2012/13

Fabian Kuhn

Union-Find Data Structure

Also known as **Disjoint-Set Data Structure**…

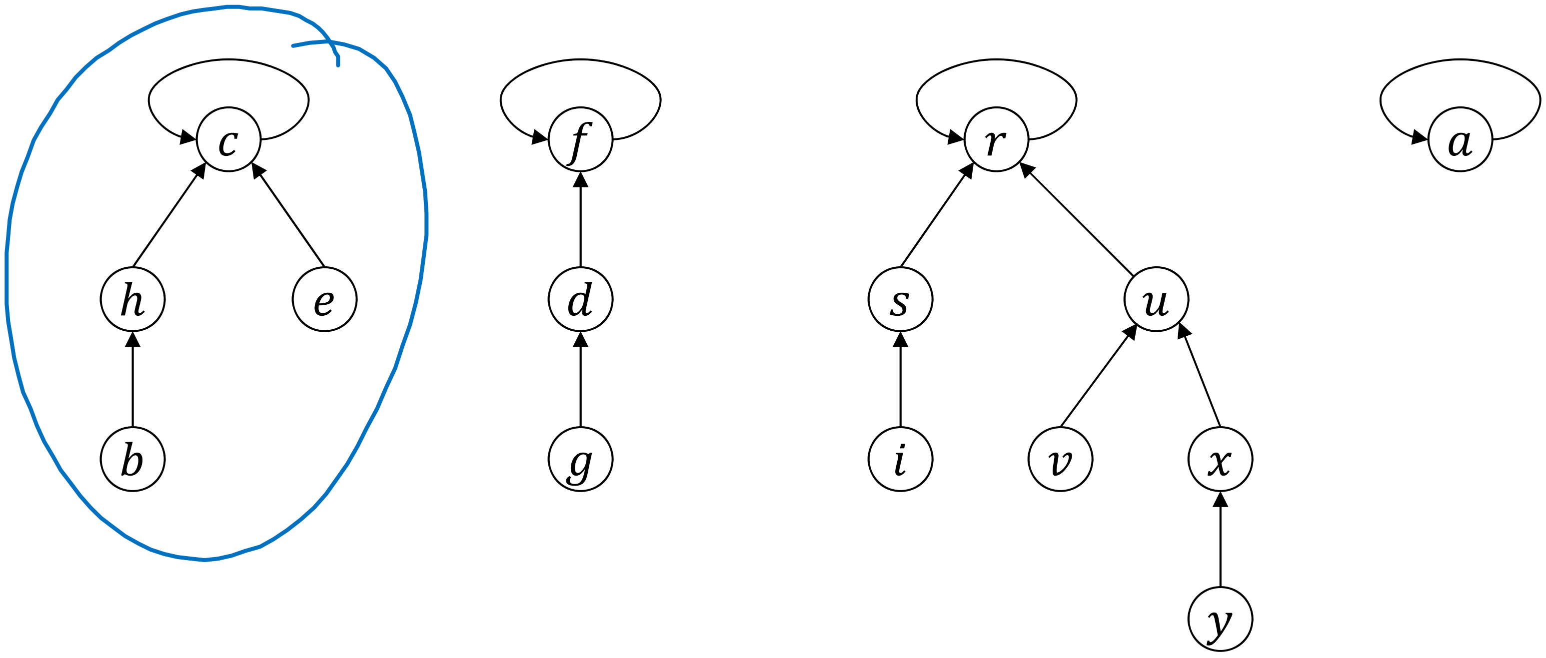
Manages partition of a set of elements

- set of disjoint sets

Operations:

- $\boxed{\textbf{make\_set}(\boldsymbol{x}):}$ create a new set that only contains element x
- $\textbf{find}(\boldsymbol{x})$**:** return the set containing x
- $\textbf{union}(\boldsymbol{x}, \boldsymbol{y})$**:** merge the two sets containing x and y

Disjoint-Set Forests

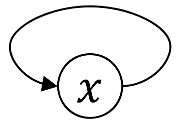


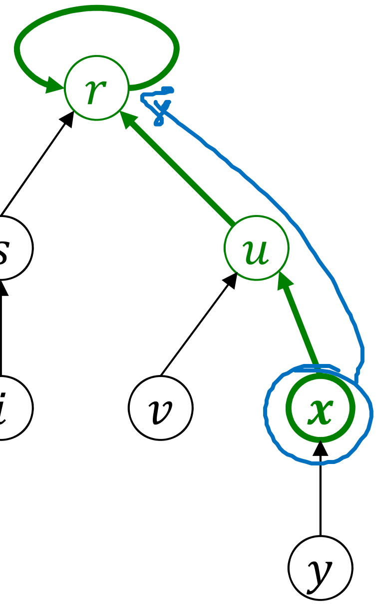
- Represent each set by a tree

- Representative of a set is the root of the tree

Disjoint-Set Forests

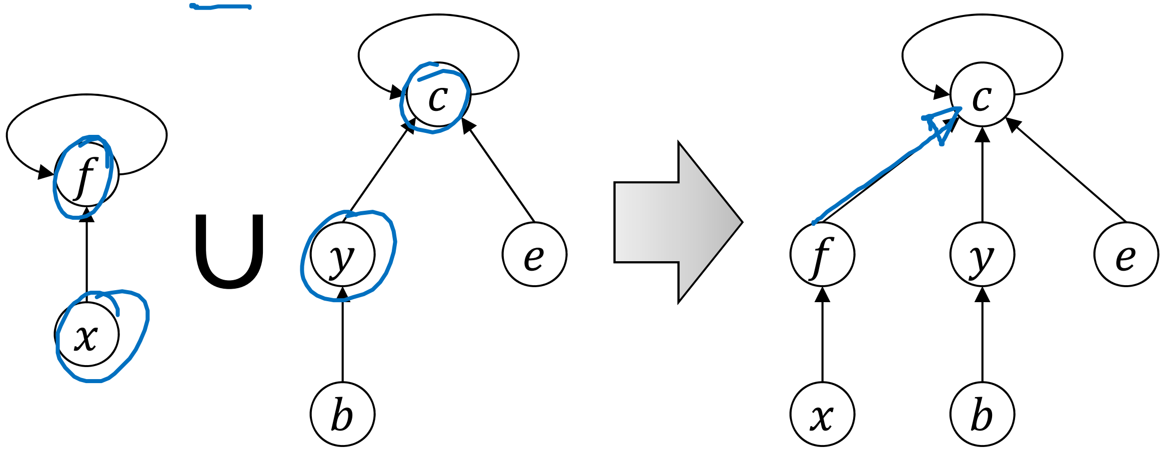
make_set(x): create new one-node tree





find(x): follow parent point to root
(parent pointer to itself)

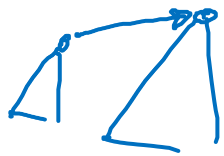
union(x, y): attach tree of x to tree of y



Union-By-Size Heuristic

Union of sets S_1 and S_2:

- Root of trees representing S_1 and S_2: r_1 and r_2
- W.l.o.g., assume that $|S_1| \geq |S_2|$
- Root of $S_1 \cup S_2$: r_1 (r_2 is attached to r_1 as a new child)



Theorem: If the union-by-rank heuristic is used, the **worst-case cost of a find-operation is $O(\log n)$**

Proof:

Union-Find Algorithms

Recall: m operations, n of the operations are make_set-operations

Linked List with Weighted Union Heuristic:

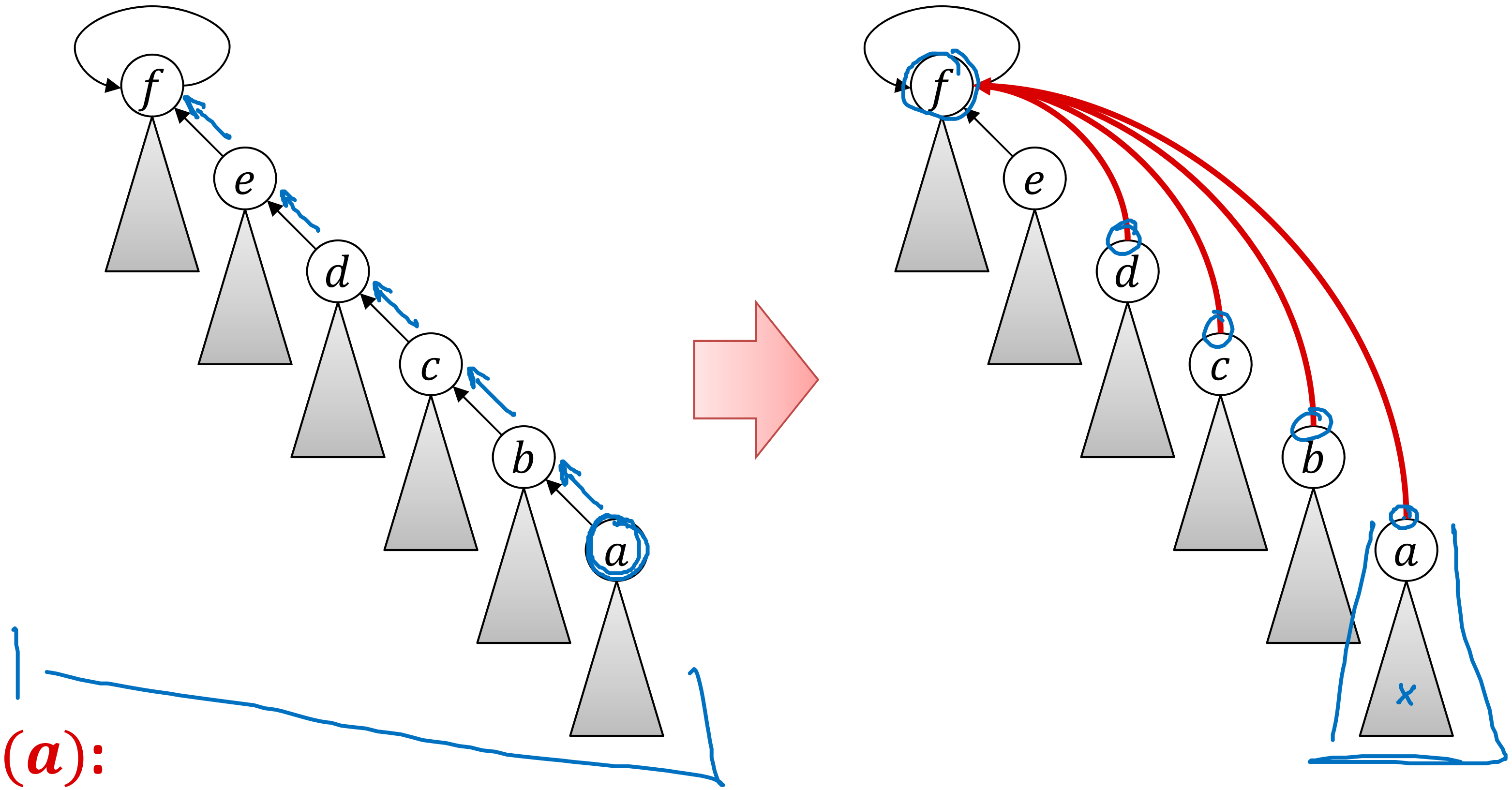
- make_set: worst-case cost $O(1)$
- find : worst-case cost $O(1)$
- union : amortized worst-case cost $O(\log n)$

Disjoint-Set Forest with Union-By-Size Heuristic:

- make_set: worst-case cost $O(1)$
- find : worst-case cost $O(\log n)$
- union : worst-case cost $O(\log n)$

Can we make this faster?

Path Compression During Find Operation



find(a):

1. **if** $a \neq a.parent$ **then**
2. $\quad a.parent := \text{find}(a.parent)$
3. **return** $a.parent$

Complexity With Path Compression

When using only path compression (without union-by-~~rank~~ size):

m: total number of operations

- f of which are find-operations
- n of which are make_set-operations
  → at most $n-1$ are union-operations

Total cost: $O\left(n + f \cdot \left\lceil \log_{2+f/n} n \right\rceil\right) = O\left(m + f \cdot \log_{2+m/n} n\right)$

make-set
union

depends on f

$f = n^{3/2} \rightarrow \log_{2+f/n}(n) = O(1)$

Union-By-Size and Path Compression

Theorem:

Using the combined union-by-size and path compression heuristic, the running time of m disjoint-set (union-find) operations on n elements (at most n make_set-operations) is

$$\boldsymbol{\Theta}\left(\boldsymbol{m} \cdot \boldsymbol{\alpha}(\boldsymbol{m}, \boldsymbol{n})\right),$$

Where $\alpha(m, n)$ is the inverse of the Ackermann function.

incr. in m & n
grows extremely slowly

in practice:
$\alpha(m,n) \leq 4$

Ackermann Function and its Inverse

Ackermann Function:

grows extremely fast

For $k, \ell \geq 1$,

$$A(k,\ell) := \begin{cases} 2^{\ell}, & \text{if } k = 1, \ell \geq 1 \\ A(\underline{k-1}, \underline{2}), & \text{if } k > 1, \ell = 1 \\ A(\underline{k-1}, A(k, \ell-1)), & \text{if } k > 1, \ell > 1 \end{cases}$$

Inverse of Ackermann Function:

$m \geq n$ $\qquad\qquad\qquad\qquad \geq 1$

$$\underline{\alpha(m,n)} := \min\{k \geq 1 \mid A(k, \lfloor m/n \rfloor) > \log_2 n\}$$

Inverse of Ackermann Function

- $\alpha(m,n) := \min\{k \geq 1 \mid A(k, \lfloor m/n \rfloor) > \log_2 n\}$

  $m \geq n \Longrightarrow A(k, \lfloor m/n \rfloor) \geq A(k,1) \Longrightarrow \alpha(m,n) \leq \min\{k \geq 1 \mid A(k,1) > \log n\}$

  $\lfloor m/n \rfloor \geq 1$

- $A(1,\ell) = 2^\ell, \quad A(k,1) = A(k-1,2),$
  $A(k,\ell) = A\big(k-1, A(k,\ell-1)\big)$

$A(1,1) = 2, \quad A(2,1) = A(1,2) = 4$

$A(3,1) = A(2,2) = A(1, A(2,1)) = A(1,4) = 2^4 = 16$

$A(4,1) = A(3,2) = A(2, A(3,1)) = A(2,16) = A(1, A(2,15)) = 2^{A(2,15)}$

$A(2,15) = A(1, A(2,14)) = 2^{A(2,14)}, \quad A(2,14) = 2^{A(2,13)}, \ldots$

$A(4,1) = \left.2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}\right\}{\geq 15} \geq 2^{2^{2^{2^2}}} = 2^{65536}$

atoms in obs. universe $\approx 10^{80} < 2^{300}$