Disjoint set data structure

(S: a collection of disjoint sets)

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

Make-Set(x)

{a, b}, {c, d}, {e, f, g}, {h}, {x}

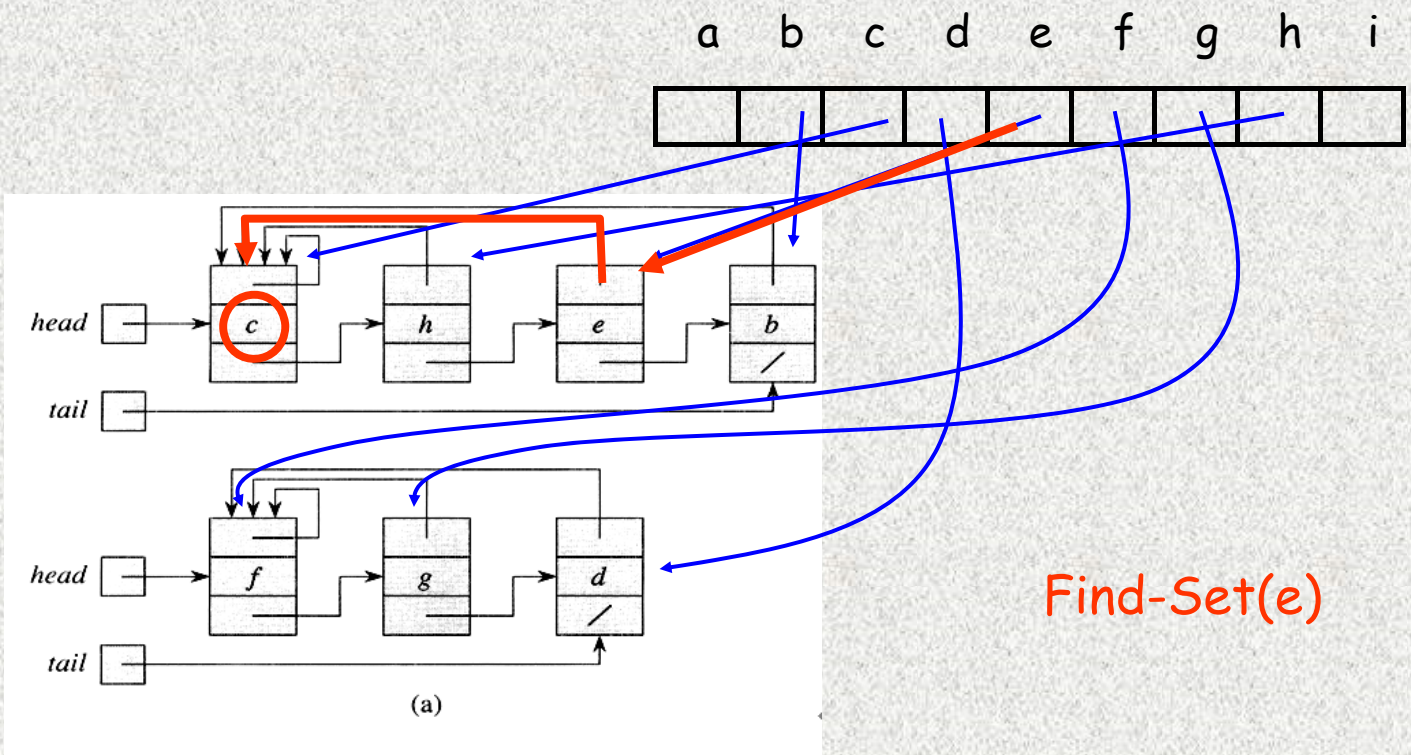
Union(c, h)

{a, b}, {c, d, h}, {e, f, g}, {x}

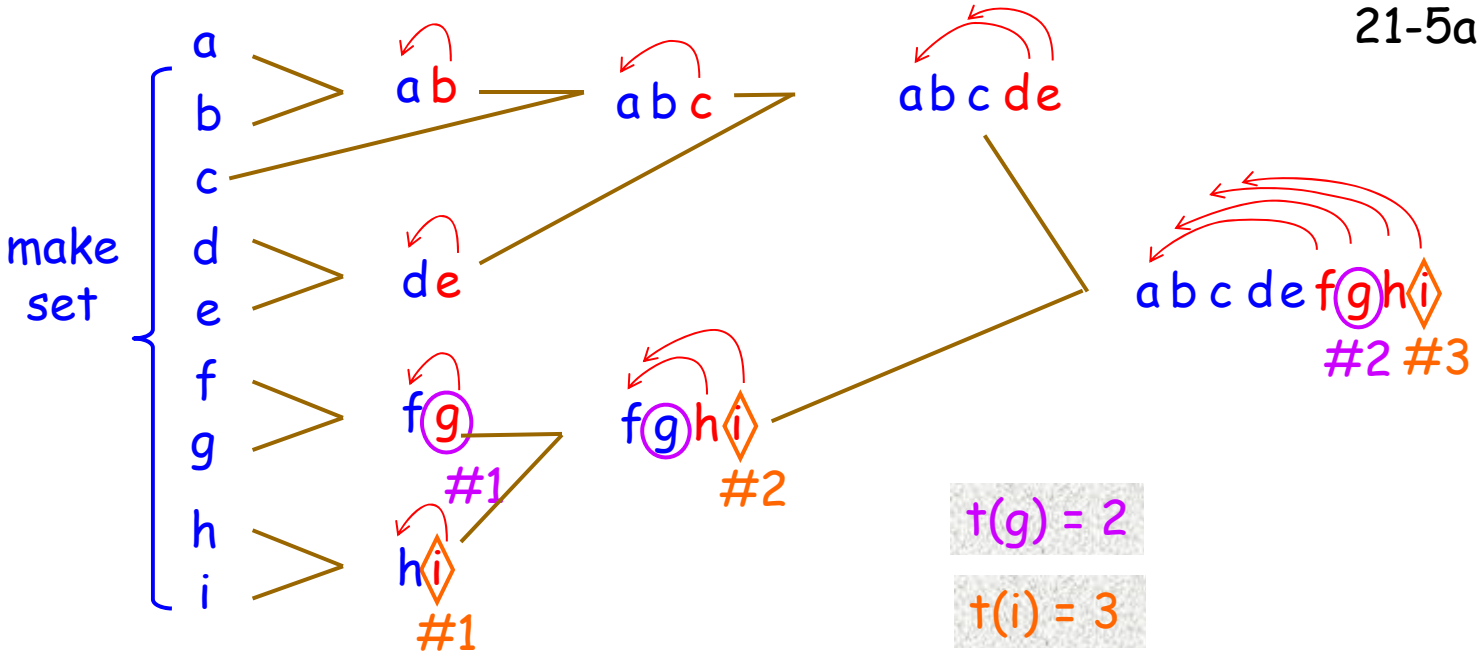
Find-Set(h)

return d (representative)

handle: pointers to the objects in a data structure



Find-Set(e)



t(g) = 2

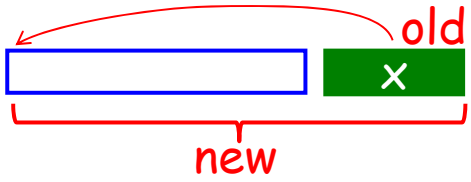
t(i) = 3

t(d) = ?

Time for all Union = $\sum_x t(x)$

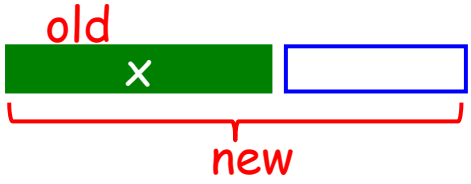
Perform Union on the list containing x

Case 1:



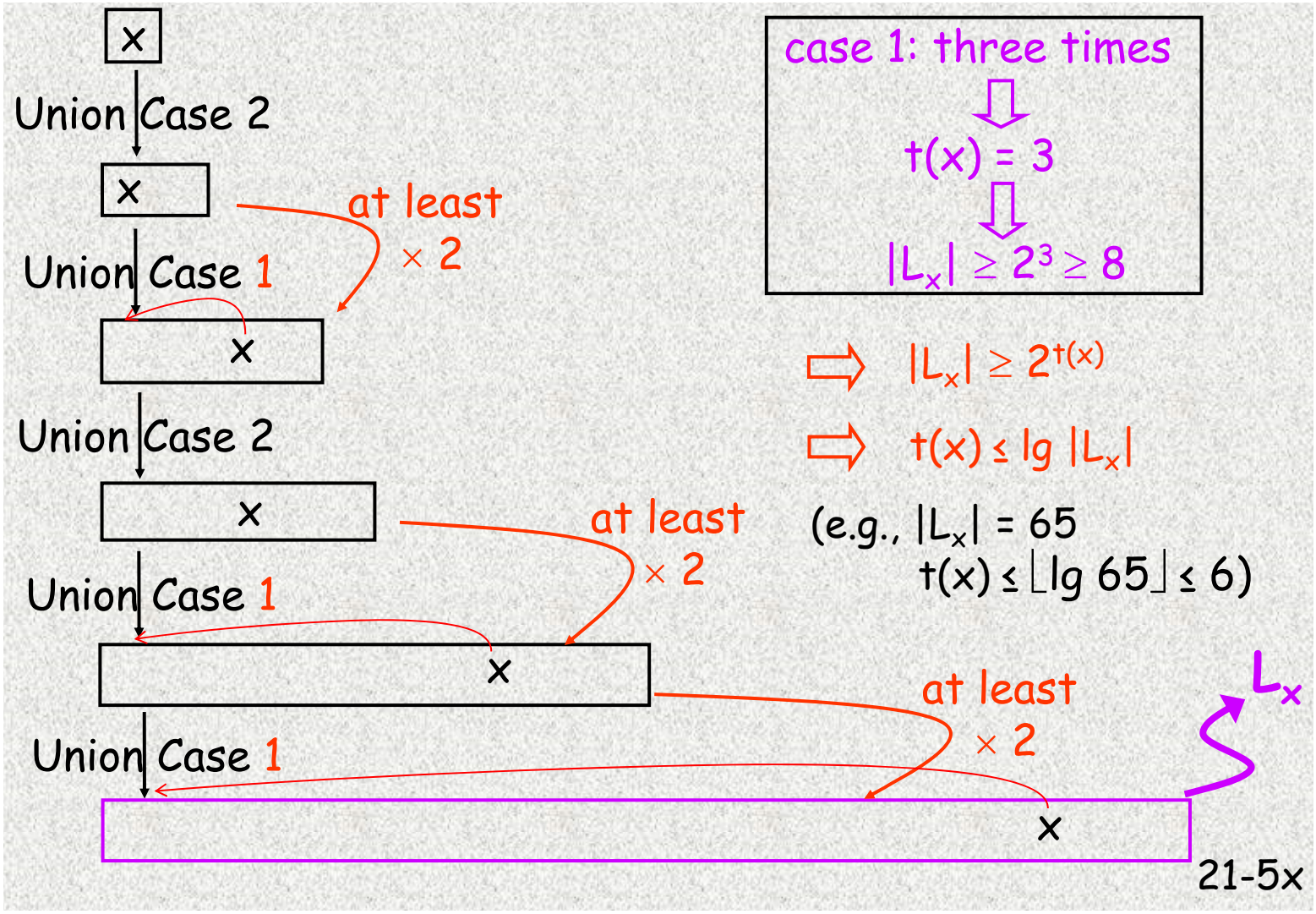
update x ⇨ $L_{NEW} \geq 2\, L_{OLD}$

Case 2:



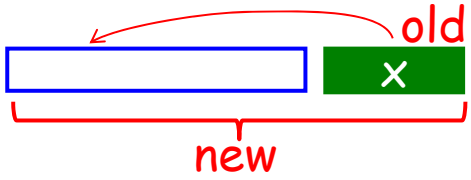
not update x ⇨ $L_{NEW} \geq L_{OLD} + 1$

⇨ IF x has been update k times, $|L_x| \geq 2^k$

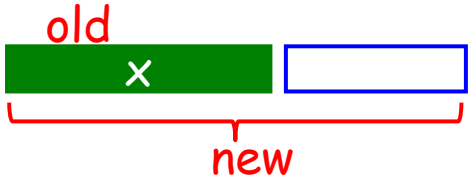


Perform Union on the list containing x

21-5b

Case 1: 

update x ⇨ $L_{NEW} \geq 2\, L_{OLD}$

Case 2: 

not update x ⇨ $L_{NEW} \geq L_{OLD} + 1$

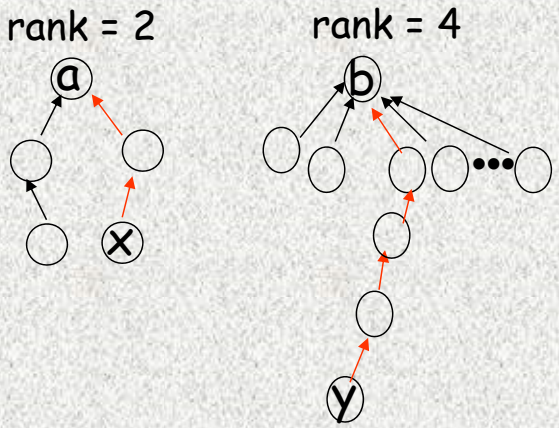
⇨ IF x has been update k times, $|L_x| \geq 2^k$

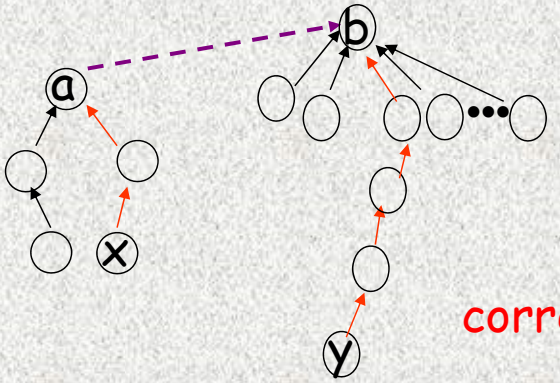
⇨ $t(x) = k \leq \lg |L_x| \leq \lg n$

⇨ Time all Union = $\sum_x t(x) \leq \sum_x \lg n \leq n \lg n$

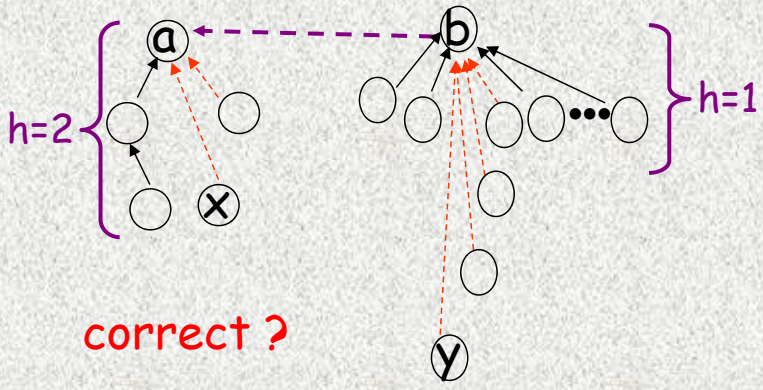
Union by rank + Path compression

Union(x, y)





correct ?

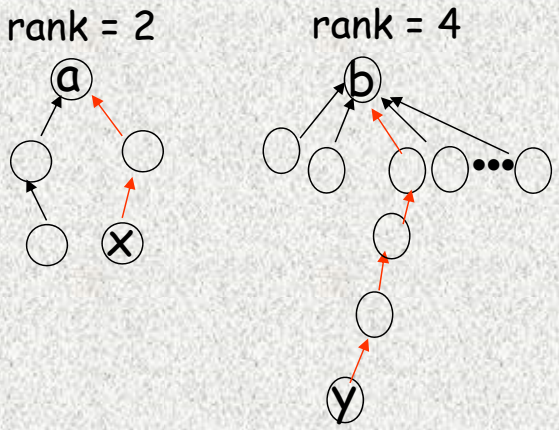


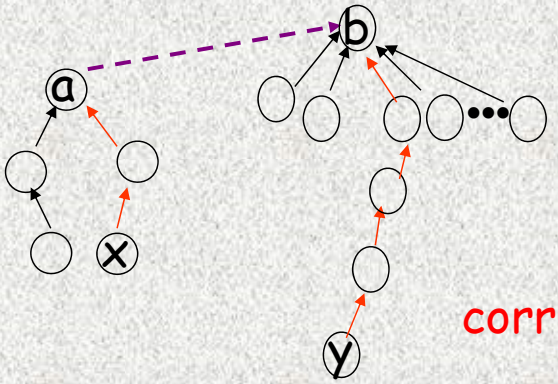
correct ?

一個 union 藏 3 兩個 find

Union by rank + Path compression

Union(x, y)

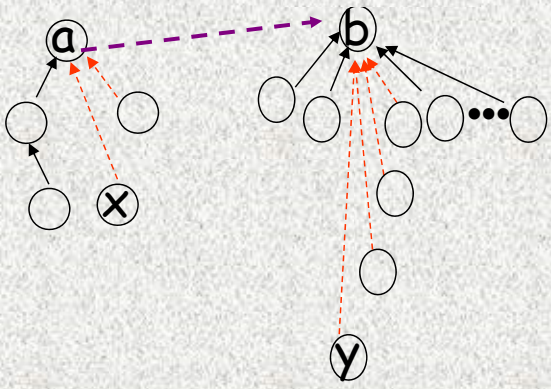




correct ?

an upper bound

rank = ?

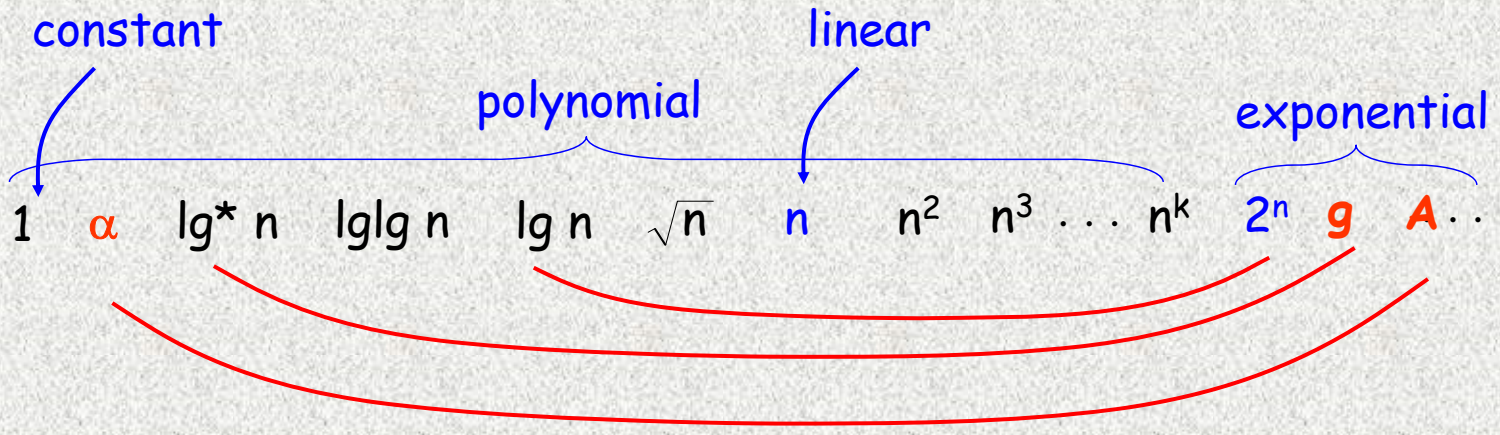


一個 union 藏 3 兩個 find

21-8x

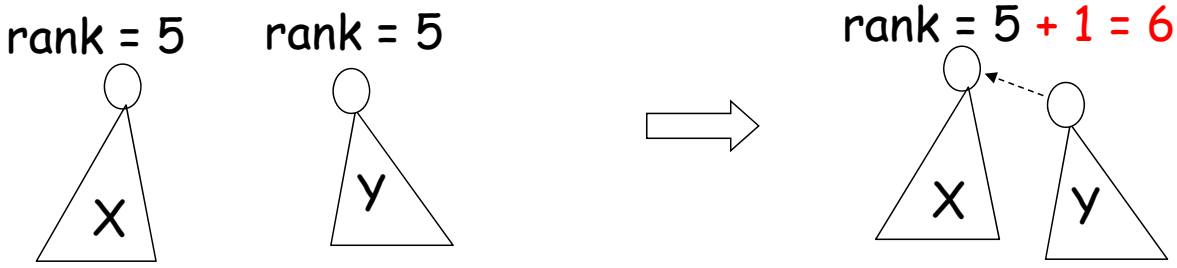
time complexities

α: almost constant
$n\alpha$: almost linear

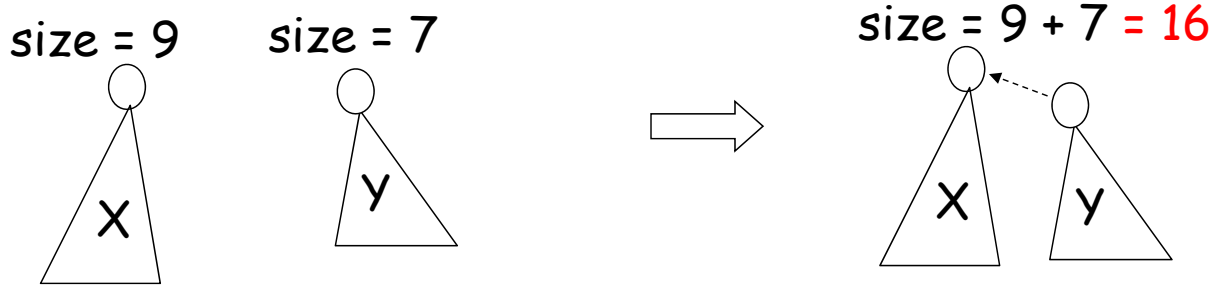


Union by Rank and Union by Size

UR:



US:



$3 \times 2 = 6$ possibilities

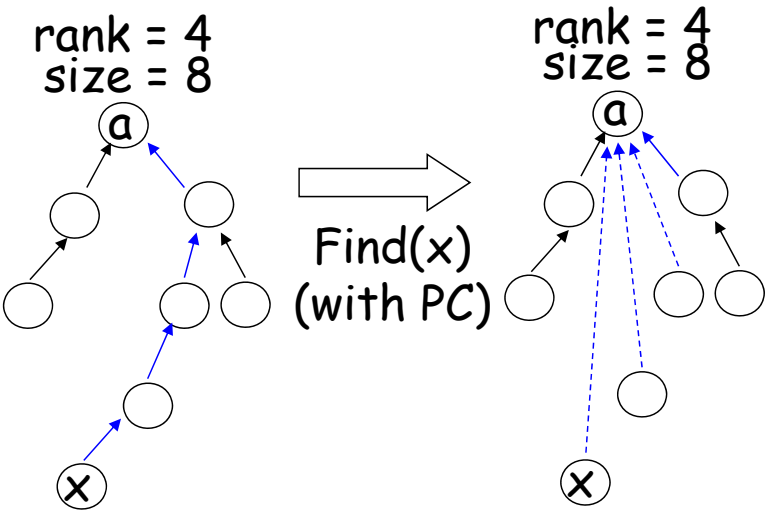
may be approximate

UR (union by rank)

US (union by size)

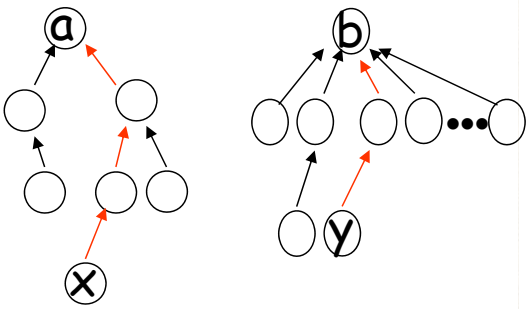
exact

$\times$ PC (path compression)

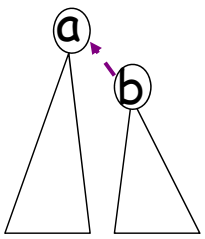


* US is as good as UR

Union(x, y)

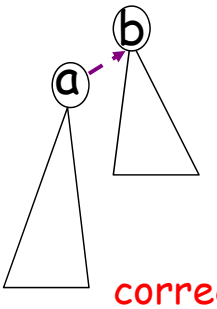


UR only



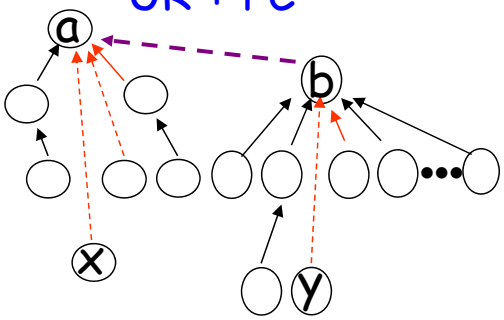
correct?

US only



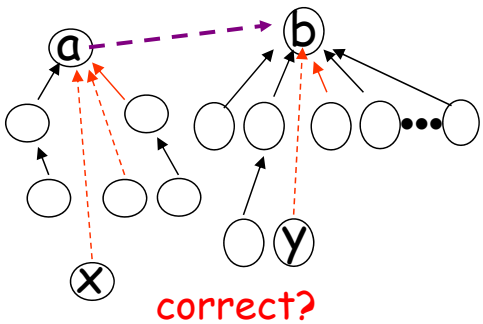
correct?

UR + PC



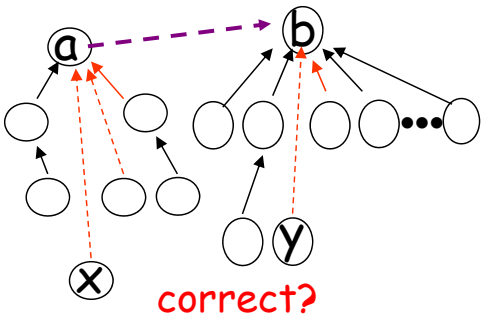
correct?

US + PC



correct?

PC only



correct?

Data Structures for Disjoint Sets

Traditional: try to reduce single OP worst-case

Amortized: try to reduce overall time

Worst-case

Procedure	✓ 2-3 trees	✗ Linked Lists	✓ Forests
MAKE-SET	$O(1)$	$O(1)$	$O(1)$
UNION	$O(\lg n)$	$O(n)$	$O(\lg n)$
FIND-SET	$O(\lg n)$	$O(1)$	$O(\lg n)$

Amortized

Procedure	✗ 2-3 trees	✓ Linked Lists	☆ Forests
MAKE-SET	$O(1)$	$O(1)$	$O(1)$
UNION	$O(\lg n)$	$O(\lg n)$	$O(\alpha(n))$
FIND-SET	$O(\lg n)$	$O(1)$	$O(\alpha(n)))$

much simpler

better & simpler