

# Algoritmer og Datastrukturer 1

Gerth Stølting Brodal

**Union-Find [CLRS, kapitel 21.2-21.3]**



AARHUS UNIVERSITET

# Union-Find

**MakeSet( $x$ )**

Opret en ny mængde  $S = \{ x \}$

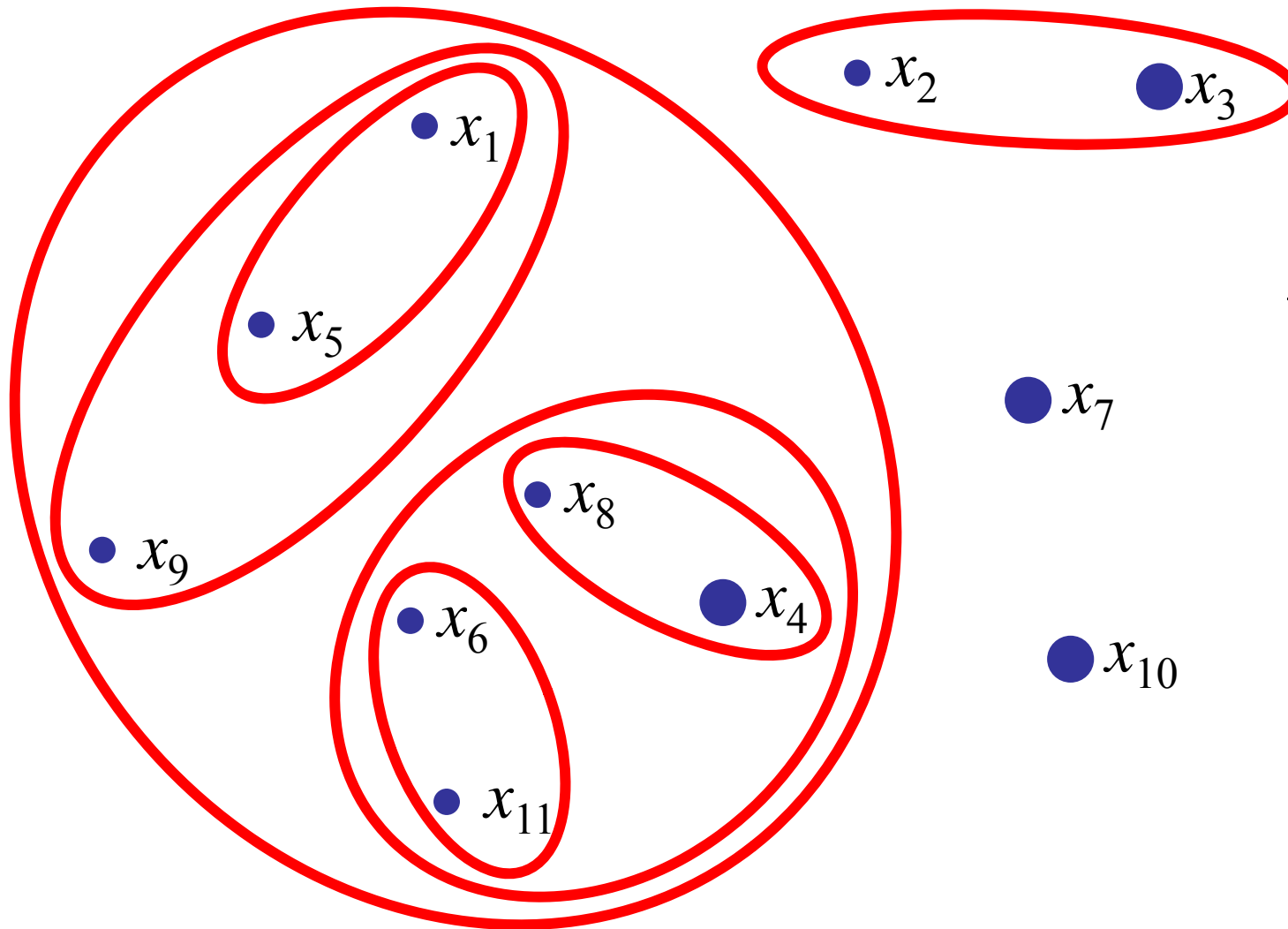
**Union( $x, y$ )**

Erstat  $S_x = \{ \dots, x, \dots \}$   $S_y = \{ \dots, x, \dots \}$   
med  $S_x \cup S_y = \{ \dots, x, \dots, y, \dots \}$

**FindSet( $x$ )**

Retuner en *repræsentant* for  $S_x = \{ \dots, x, \dots \}$   
FindSet( $x$ ) = FindSet( $y$ ) hvis og kun hvis  $x$   
og  $y$  er i samme mængde

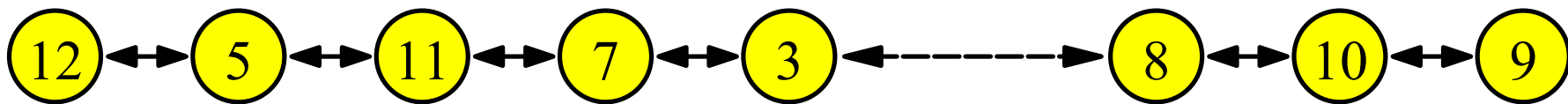
# Eksempel : Union-Find



$x_i$	FindSet( $x_i$ )
$x_1$	$x_4$
$x_2$	$x_3$
$x_3$	$x_3$
$x_4$	$x_4$
$x_5$	$x_4$
$x_6$	$x_4$
$x_7$	$x_7$
$x_8$	$x_4$
$x_9$	$x_4$
$x_{10}$	$x_{10}$
$x_{11}$	$x_4$

# Trærepræsentation (I)

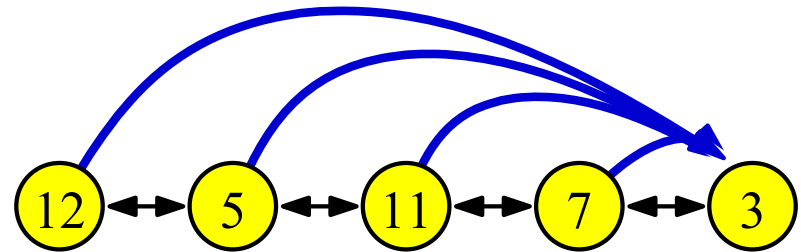
- Mængde = enkeltkædet liste
- MakeSet = lav en ny knude
- FindSet = returner sidste knude
- Union = konkatener listerne



<b>MakeSet(<math>S, x</math>)</b>	$O(1)$
<b>Union(<math>x, y</math>)</b>	$O( S_x  +  S_y )$
<b>FindSet(<math>x</math>)</b>	$O( S_x )$

# Trærepræsentation (II)

- Mængde = enkelt kædet liste
- MakeSet = lav en ny knude
- FindSet = returner sidste knude
- Union = konkatener listerne og opdater pointerne



<b>MakeSet(<math>S, x</math>)</b>	$O(1)$
<b>Union(<math>x, y</math>)</b>	$O(\min( S_x ,  S_y ))$
<b>FindSet(<math>x</math>)</b>	$O(1)$

# Trærepræsentation (II)

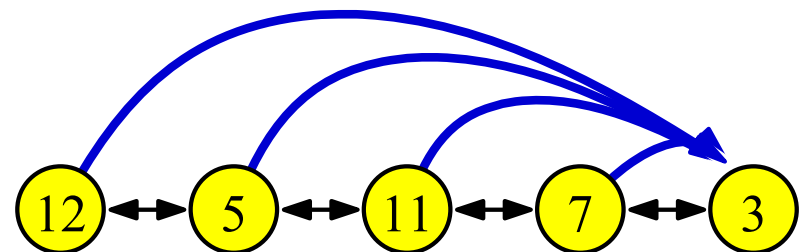
## Sekvens af Union

### Sætning

Et sekvens af  $n$  union operationer tager tid  $O(n \cdot \log n)$

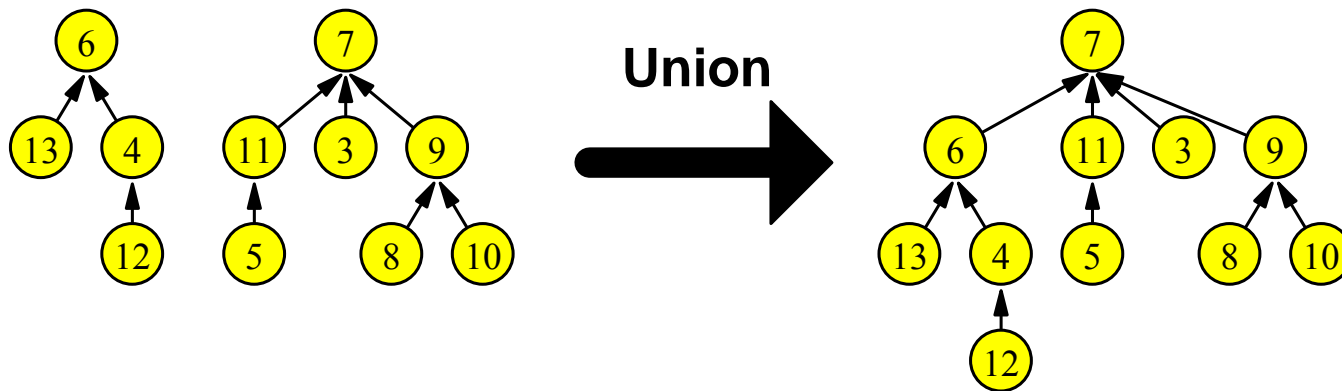
### Bevis

Hver pointer flyttes højest  $\log n$  gange - hver gang til en liste der mindst er dobbelt så stor



# Trærepræsentation (III)

- Mængde = træ
- MakeSet = lav en ny knude
- FindSet = returner roden
- Union = Sæt det "lille" træ under roden af det "store" træ



# Stikomprimering

MAKE-SET( $x$ )

- 1  $x.p = x$
- 2  $x.rank = 0$

UNION( $x, y$ )

- 1 LINK(FIND-SET( $x$ ), FIND-SET( $y$ ))

LINK( $x, y$ )

- 1 **if**  $x.rank > y.rank$
- 2      $y.p = x$
- 3 **else**  $x.p = y$
- 4     **if**  $x.rank == y.rank$
- 5          $y.rank = y.rank + 1$

linking ved rank

FIND-SET( $x$ )

- 1 **if**  $x \neq x.p$
- 2      $x.p = \text{FIND-SET}(x.p)$
- 3 **return**  $x.p$

stikomprimering



# Analyse af Trærepræsentation med Rank-Linkning

## Lemma

$$height[rod] \leq rank[rod]$$

$$size[rod] \geq 2^{rank[rod]}$$

## Bevis

Induktion.

<b>MakeSet</b> ( $S, x$ )	$O(1)$
<b>Union</b> ( $x, y$ )	$O((\log  S_x ) + (\log  S_y ))$
<b>FindSet</b> ( $x$ )	$O(\log  S_x )$

(Ovenstående udnytter kun linking by rank)

# Analyse af Trærepræsentation med Stikomprimering

## Sætning

En sekvens af  $m$  Union-Find operationer på  $n$  elementer tager tid  $O(m \cdot \alpha(n))$  hvor  $\alpha(n)$  er den Inverse til Ackerman funktionen ([CLRS], kapitel 21.4) hvor for alle praktiske formål  $\alpha(n) \leq 4$ .