Forén og find

- Introduktion
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter

Philip Bille

Forén og find

- Introduktion
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter

Forén og find

- Forén og find (*union-find*). Vedligehold en dynamisk familie af mængder under operationer:
  - INIT(n): opret mængder {0}, {1},…, {n-1}
  - UNION(i,j): forener de to mængder der indeholder i og j. Hvis i og j er i samme mængde skal der ingenting ske.
  - FIND(i): returnerer en repræsentant for mængden der indeholder i.

INIT(9)

{0} {1} {2} {3} {4} {5} {6} {7} {8}

UNION(5,0)

{1, 0, 6} {8, 3, 2, 7} {4, 5} → {1, 0, 6, 4, 5} {8, 3, 2, 7}

- Repræsentant kan være et hvilket som helst element i mængden.
- FIND(i) == FIND(j) hvis og kun hvis i og j er i samme mængde.

Forén og find

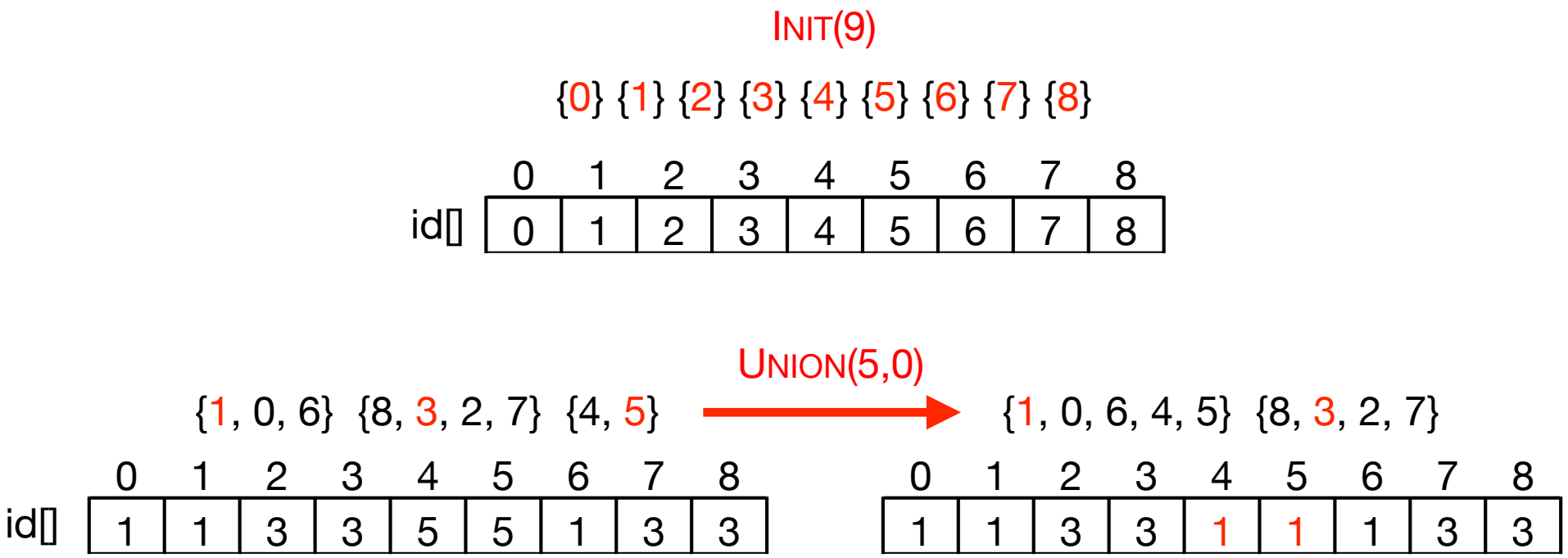
- Anvendelser.
  - Dynamiske sammenhængskomponenter.
  - Mindste udspændende træ.
  - Unificering i logik og oversættere (afgør om udtryk er ens).
  - Nærmeste fælles forfader i træer.
  - Hoshen-Kopelman algoritme i fysik
  - Spil (Hex og Go)
  - Illustration af snedige teknikker til design af datastrukturer.

Forén og find

- Introduktion
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter

Hurtig find

- Hurtig find (*quick-find*). Vedligehold en tabel id[0..n-1] så id[i] er repræsentant for i.
  - INIT(n): sæt alle elementer til at være deres egen repræsentant
  - UNION(i,j): opdater repræsentant for alle elementer i den ene mængde.
  - FIND(i): returner repræsentant.

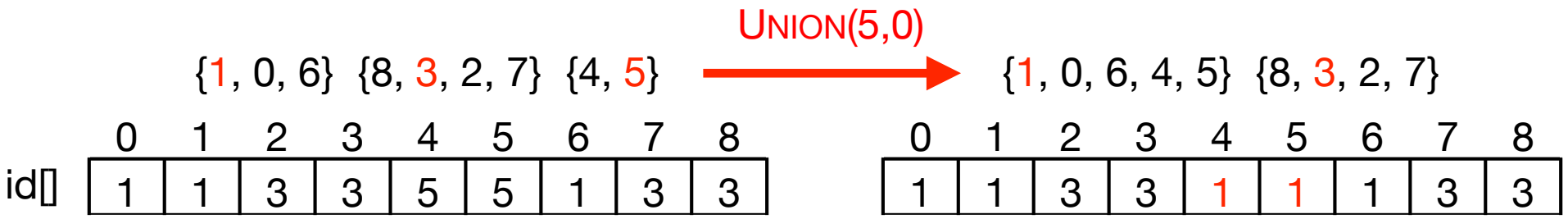


Hurtig find

```
INIT(n):
    for k = 0 to n-1
        id[k] = k
```

```
FIND(i):
    return id[i]
```

```
UNION(i,j):
    iID = FIND(i)
    jID = FIND(j)
    if (iID ≠ jID)
        for k = 0 to n-1
            if (id[k] == iID)
                id[k] = jID
```



- Tid.
  - O(n) tid for INIT, O(n) tid for UNION og O(1) tid for FIND.

Hurtig find

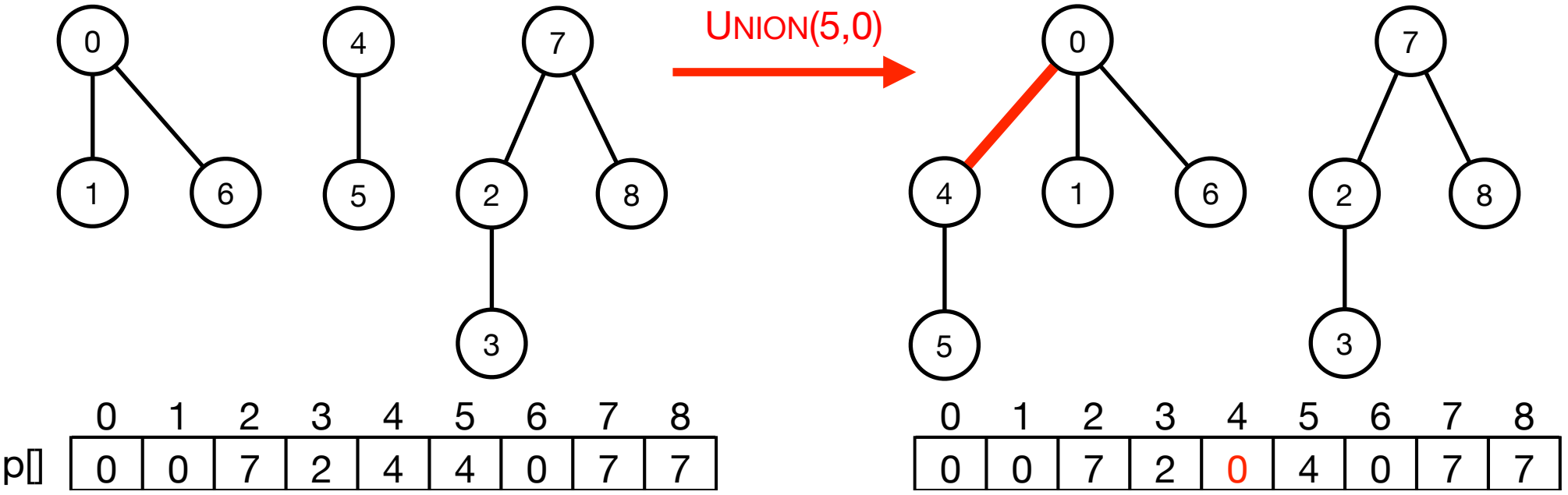
- Theorem. Vi kan løse forén og find med n elementer i
  - O(n) tid for INIT
  - O(1) tid for FIND
  - O(n) tid for UNION

Forén og find

- Introduktion
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter

Hurtig forening

- Hurtig forening (*quick-union*). Vedligehold hver mængde som et rodfæstet træ repræsenteret ved tabel p[0..n-1] af forældrepegere. Roden af træ er repræsentant for mængde og p[rod] = rod.
  - INIT(n): lav n træer med et element hver.
  - UNION(i,j): hvis FIND(i) ≠ FIND(j), gør rod af det ene træ til barn af roden af det andet træ.
  - FIND(i): følg sti til rod og returner rod.



Hurtig forening

- INIT(n): lav n træer med et element hver.
- UNION(i,j): hvis FIND(i) ≠ FIND(j), gør rod af det ene træ til barn af roden af det andet træ.
- FIND(i): følg sti til rod og returner rod.

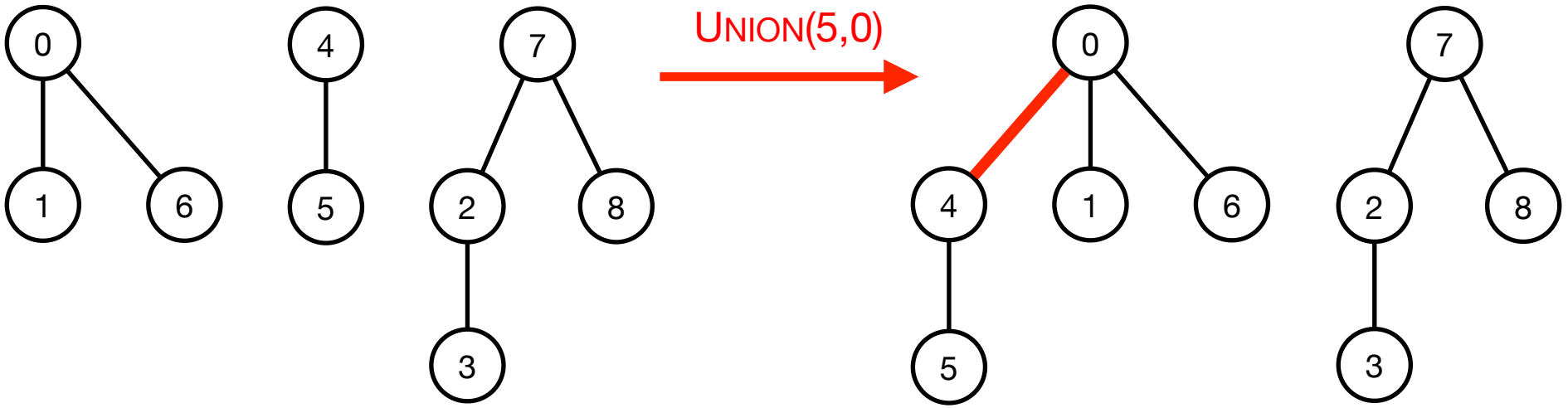
- Opgave. Vis datastruktur efter hver operation i følgende sekvens.
  - INIT(7), UNION(0,1), UNION(2,3), UNION(5,1), UNION(5,0), UNION(0,3), UNION(5,2), UNION(4,3), UNION(4,6).

Hurtig forening

```
INIT(n):
    for k = 0 to n-1
        p[k] = k
```

```
FIND(i):
    while (i != p[i])
        i = p[i]
    return i
```

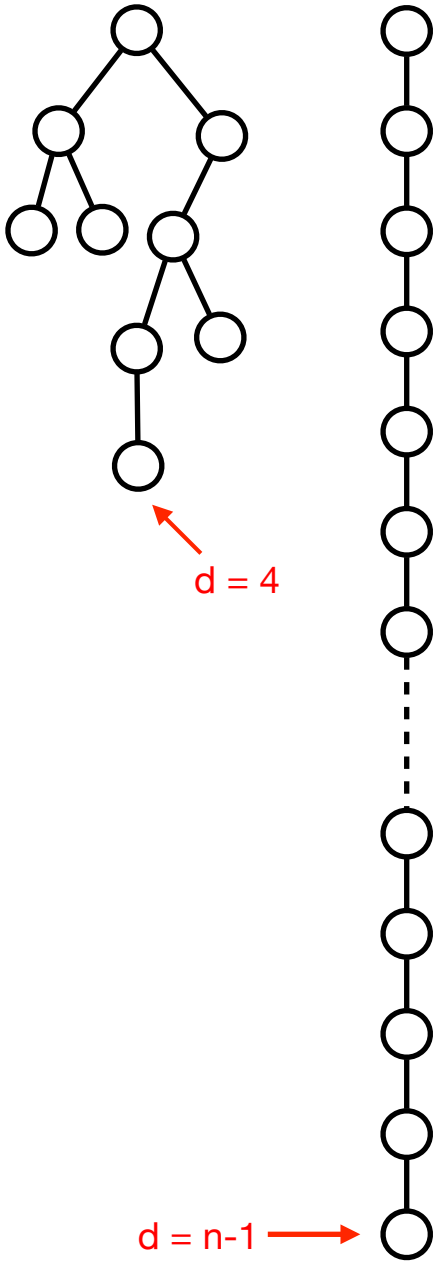
```
UNION(i,j):
    ri = FIND(i)
    rj = FIND(j)
    if (ri ≠ rj)
        p[ri] = rj
```



- Tid.
  - O(n) tid for INIT, O(d) tid for UNION og O(d) tid for FIND.

Hurtig forening

- Theorem. Vi kan løse forén og find med n elementer i
  - O(n) tid for INIT
  - O(d) tid for FIND
  - O(d) tid for UNION
- Dybden d af T er den maksimale længde af en sti fra rod til blad.
- Dårlig nyhed. Dybden kan være n-1.
- Udfordring. Kan vi sætte sammen træerne snedigt sammen så vi begrænser dybden?

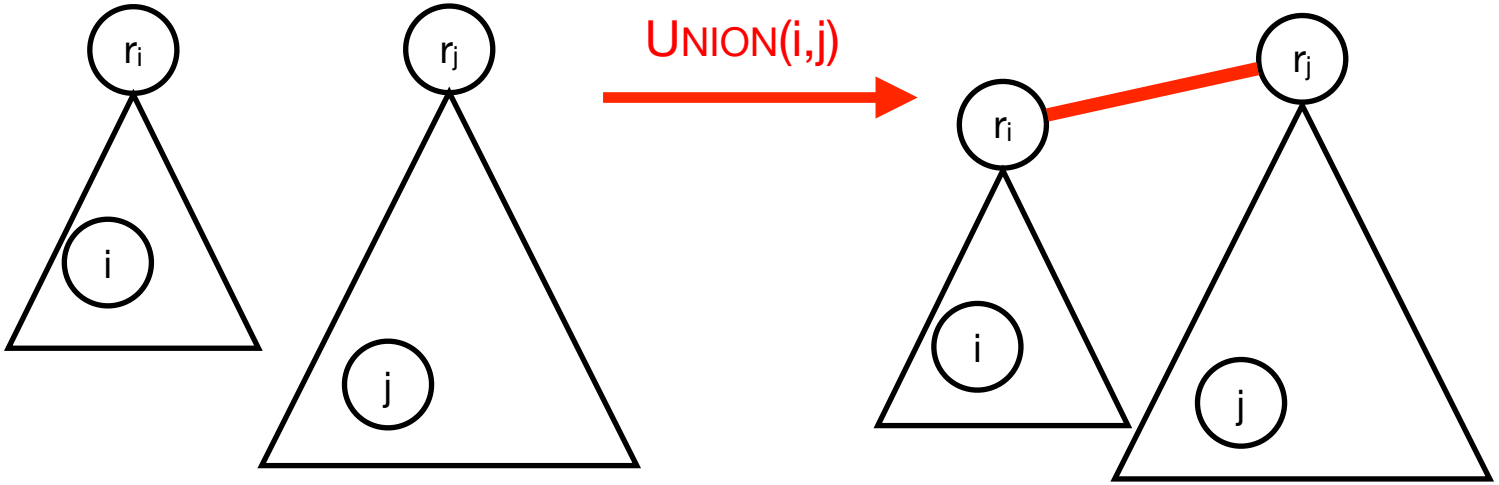


Forén og find

- Introduktion
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter

Vægtet forening

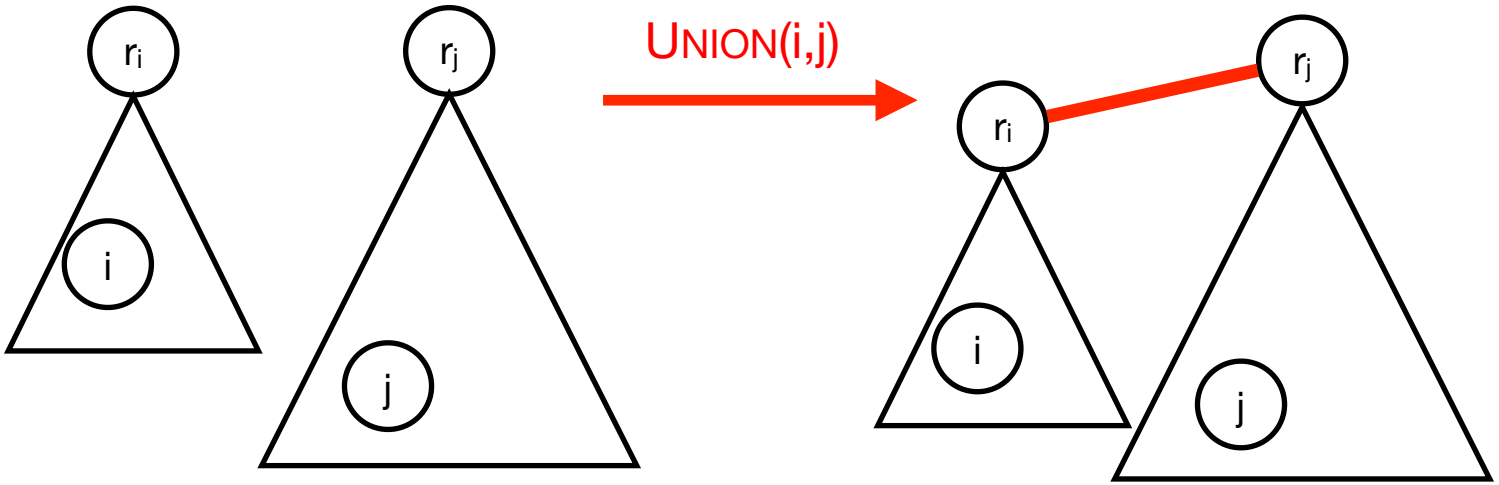
- Vægtet forening (*weighted quick-union*). Udvidelse af hurtig forening. Vedligehold tabel sz[1..n] med størrelse af hver knude = antallet af knuder i deltræ.
  - INIT: (næsten) som før.
  - FIND: som før.
  - UNION(i,j): hvis FIND(i) ≠ FIND(j), gør rod af det mindste træ til barn af roden af det største træ.
- Intuition. Vægtet forening balancerer træerne.



$sz[r_j] = sz[r_i] + sz[r_j]$

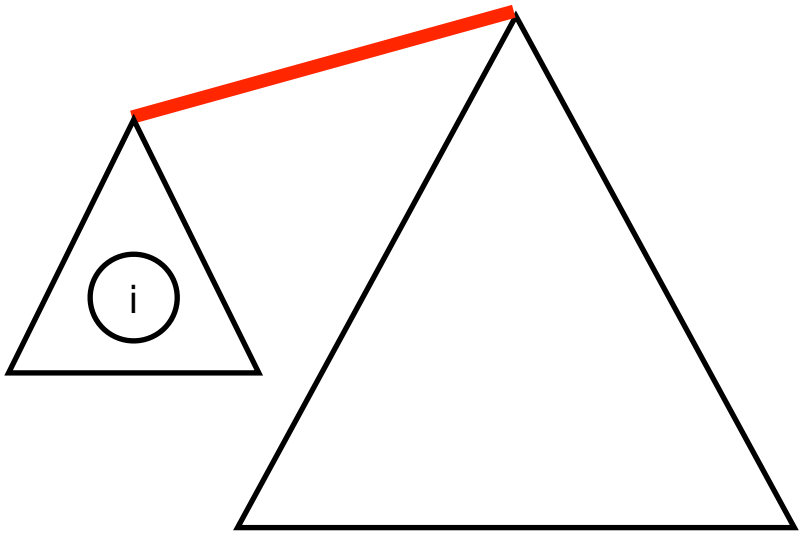
Vægtet forening

```
UNION(i,j):
    r_i = FIND(i)
    r_j = FIND(j)
    if (r_i ≠ r_j)
        if (sz[r_i] < sz[r_j])
            p[r_i] = r_j
            sz[r_j] = sz[r_i] + sz[r_j]
        else
            p[r_j] = r_i
            sz[r_i] = sz[r_i] + sz[r_j]
```



Vægtet forening

- Lemma. Med vægtet forening er dybden af en knude højst $\log_2 n$.
- Bevis.
  - Kig på en knude i med dybde d_i.
  - Initielt er $d_i = 0$.
  - d_i øges med 1 når træet med i forenes med et større træ.
  - Det forenede træ er mindst dobbelt så stort.
  - Vi kan fordoble størrelse af træer højst $\log_2 n$ gange.
  - $\Longrightarrow d_i \leq \log_2 n$.



Vægtet forening

- Theorem. Vi kan løse forén og find med n elementer i
  - O(n) tid for INIT
  - O(log n) tid for FIND
  - O(log n) tid for UNION

Forén og find

Datastruktur	UNION	FIND
hurtig find	O(n)	O(1)
hurtig forening	O(n)	O(n)
vægtet forening	O(log n)	O(log n)

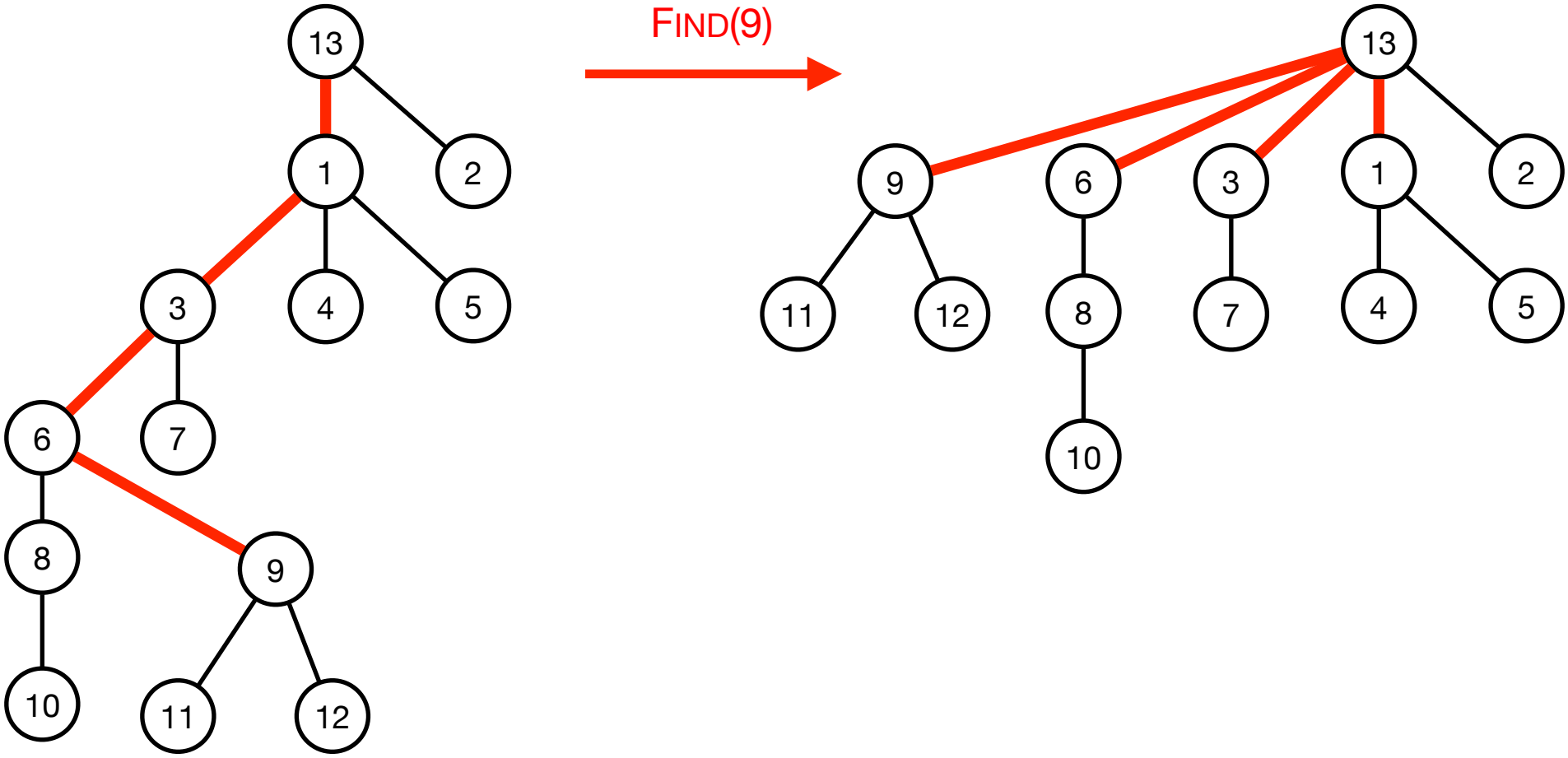
- Udfordring. Kan vi gøre det endnu bedre? Hvad er den bedste man kan håbe på?

Forén og find

- Introduktion
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter

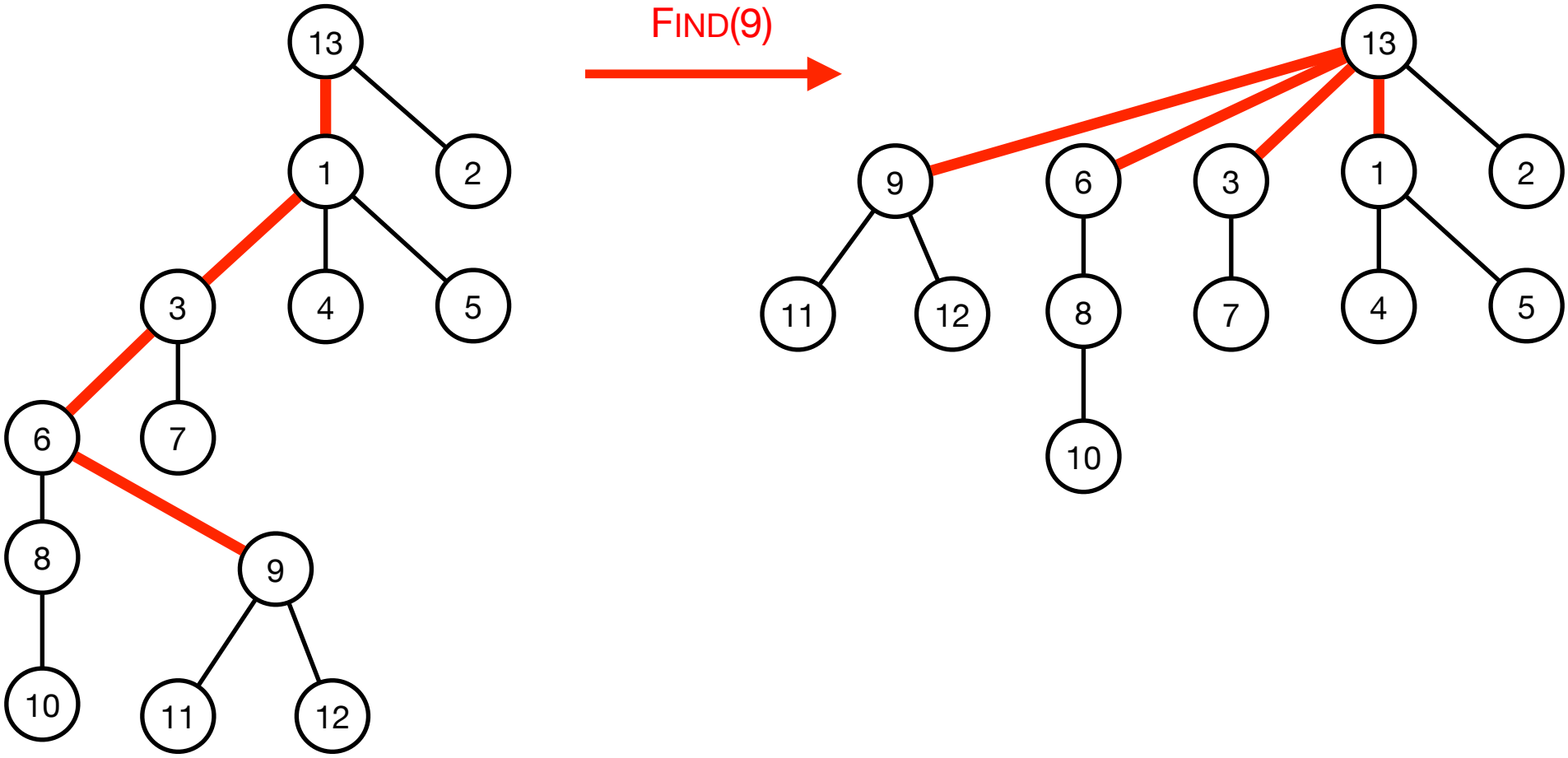
Stikompression

- Stikompression. Komprimer stien ved FIND = alle knuder på sti bliver barn af rod.
- Ændrer ikke på tid for en FIND operation. Efterfølgende FIND operationer bliver hurtigere.
- Virker både med hurtig forening og vægtet forening.



Stikompression

- Theorem [Tarjan 1975]. Med stikompression tager enhver sekvens af m FIND og UNION operationer over n elementer O(n + m α(m,n)) tid.
- α(m,n) er den inverse til Ackermanns funktion. α(m,n) ≤ 5 for ethvert praktisk input.
- Theorem [Fredman-Saks 1985]. Det er ikke muligt at understøtte m FIND og UNION operationer O(n + m) tid.



Forén og find

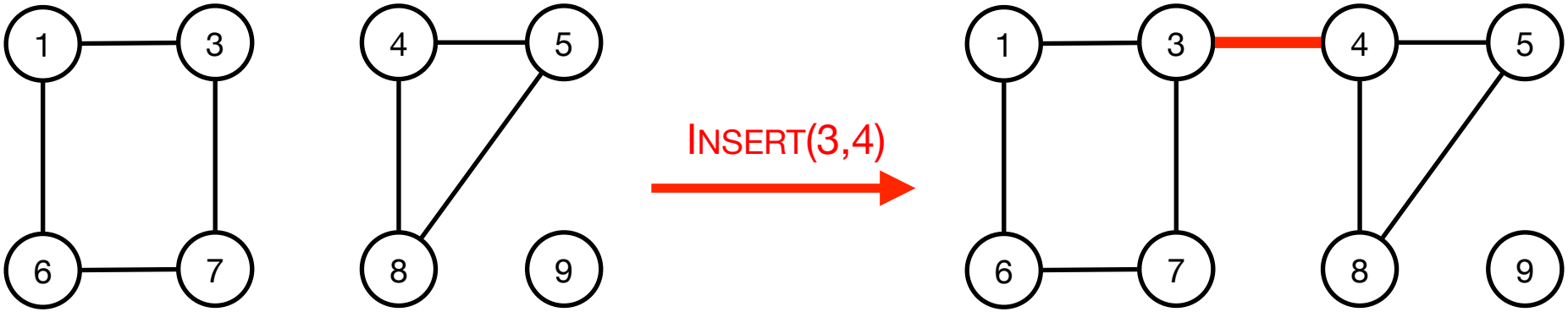
Datastruktur	m UNION og FIND
hurtig find	O(mn)
hurtig forening	O(mn)
vægtet forening	O(n + mlog n)
vægtet forening + stikompression	O(n + m α(m,n))
umuligt	O(n + m)

Forén og find

- Introduktion
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter

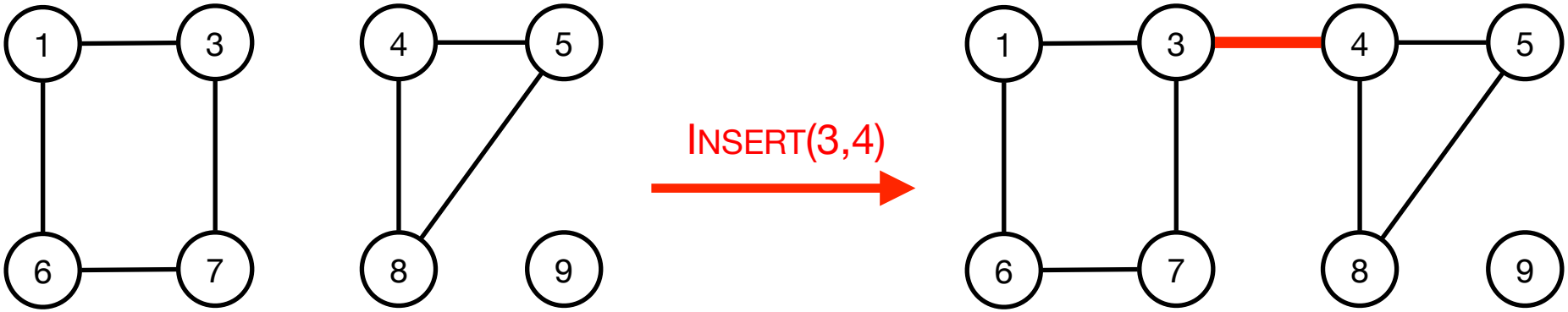
Dynamiske sammenhængskomponenter

- Dynamiske sammenhængskomponenter. Vedligehold en dynamisk graf under operationer.
  - INIT(n): opret en graf G med n knuder og ingen kanter.
  - CONNECTED(u,v): afgør om u og v er sammenhængende.
  - INSERT(u, v): tilføj kant (u,v). Vi antager (u,v) ikke allerede findes.



Dynamiske sammenhængskomponenter

- Implementation med forén og find.
  - INIT(n): initialiser en forén og find datastruktur med n elementer.
  - CONNECTED(u,v): FIND(u) == FIND(v).
  - INSERT(u, v): UNION(u,v)



Dynamiske sammenhængskomponenter

- Theorem. Vi kan løse dynamiske sammenhængskomponenter i en graf med n knuder i
  - O(n) tid for INIT
  - O(log n) tid for CONNECTED
  - O(log n) tid for INSERT

Forén og find

- Introduktion
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter