Foren og find

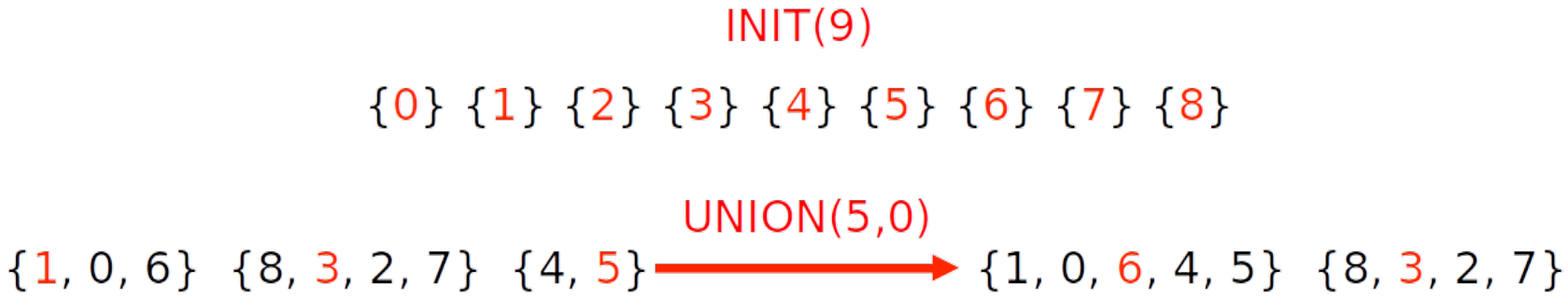
- Foren og find
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter

Foren og find

Forén og find: (union-find)

Vedligehold en dynamisk familie af mængder under følgende operationer:

- Init(n) opret mængder $\{0\}, \{1\}, \ldots, \{n-1\}$
- Union(i, j) forén de to mængder der indeholder hhv. i og j.
  Hvis i og j er i samme mængde, skal der ingenting ske.
- Find(i) returnér en repræsentant for mængden der indeholder i.



Repræsentant kan være hvilket som helst element i mængden.

Find(i) == Find(j) hvis og kun hvis i og j er i samme mængde.

Foren og find – anvendelser

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.

Foren og find

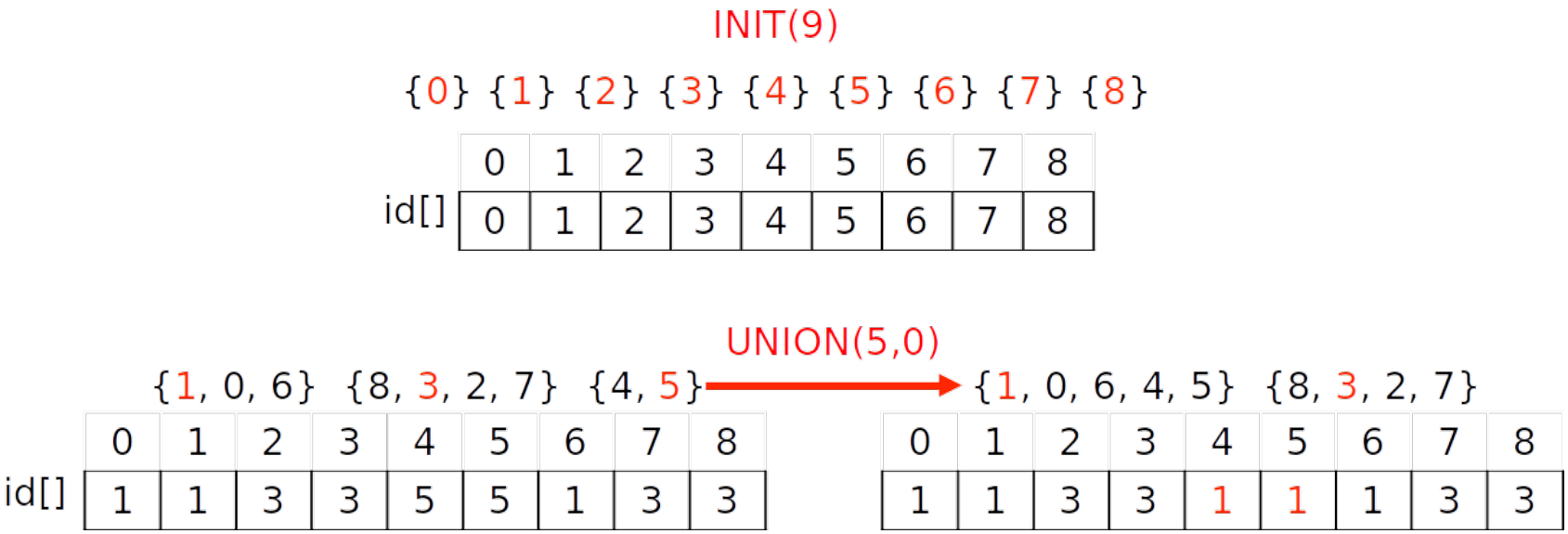
- Foren og find
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter

Hurtig find

Hurtig find: (quick find)

Vedligehold et array 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) hvis Find(i) $\neq$ Find(j), opdatér repræsentant for alle elementer i den ene mængde,
- Find(i) returnér repræsentant.

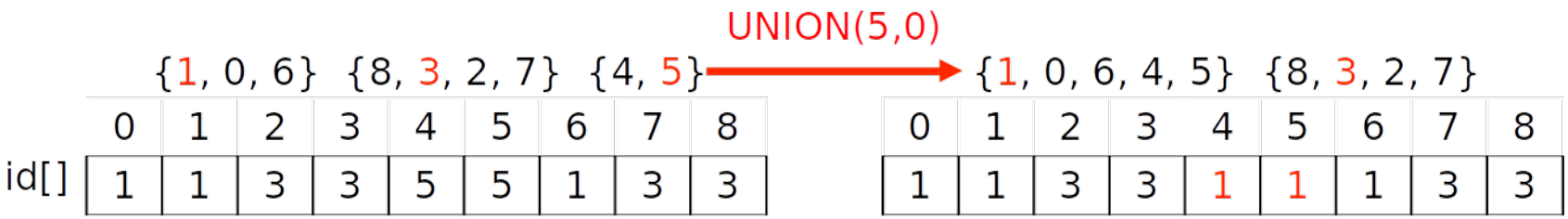


Hurtig find

```
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: Init $O(n)$ tid. Union $O(n)$ tid. Find $O(1)$ tid.

Foren og find

- Foren og find
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter

Hurtig forening

Hurtig forening: (quick union)

Vedligehold hver mængde som et rodfæstet træ.

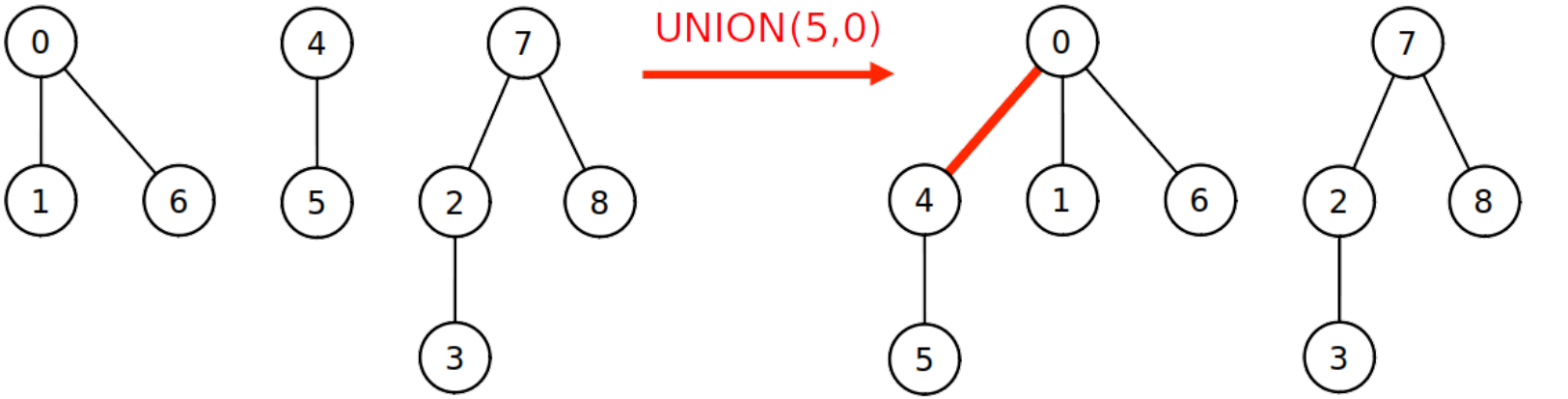
Implementér træ som array p[0..n-1] så p[i] er forælder af i, og p[root] = root.

Roden af træ er repræsentant for mængde.

Init(n) lav n træer med et element hver,

Union(i,j) hvis Find(i) $\neq$ Find(j), gør rod af det ene træ til barn af roden af det andet træ,

Find(i) følg sti til rod og returnér rod



	0	1	2	3	4	5	6	7	8
p[]	0	0	7	2	4	4	0	7	7

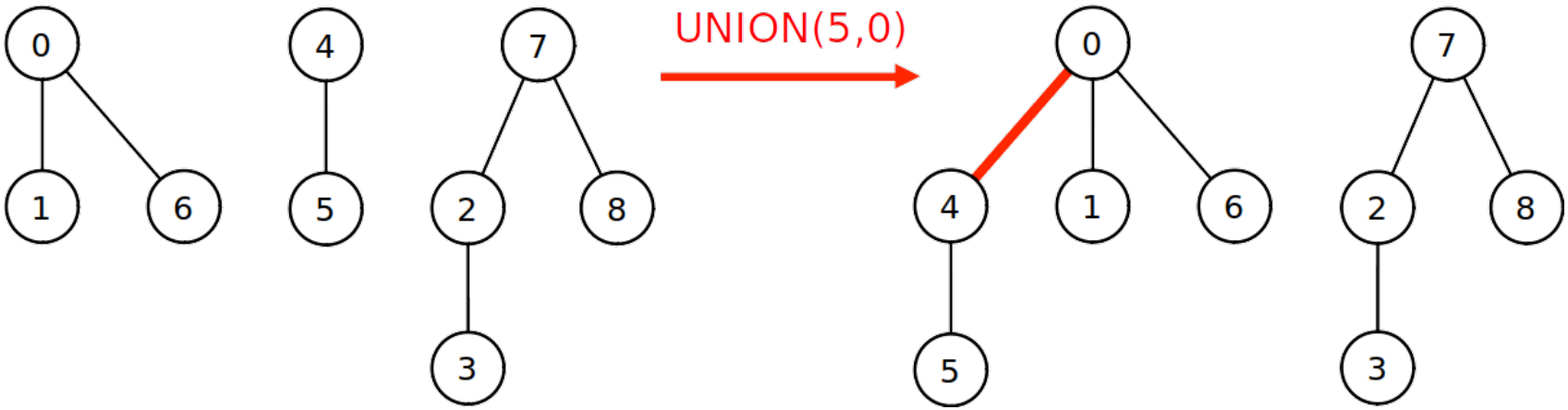
0	1	2	3	4	5	6	7	8
0	0	7	2	0	4	0	7	7

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):
    r_i = FIND(i)
    r_j = FIND(j)
    if (r_i ≠ r_j)
        p[r_i] = r_j
```



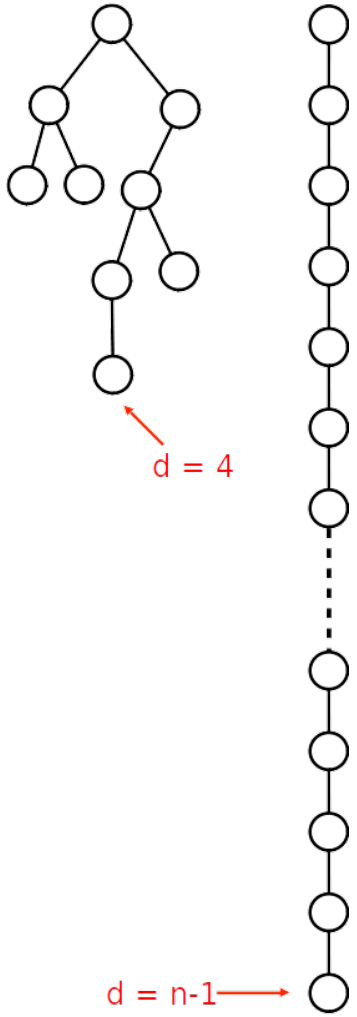
Tid: Init(n) $O(n)$. Find(i): $O(d_i)$, d_i er dybden af i i i's træ. Union(i, j): $O(d_i + d_j)$. (Igen er d_j dybden af j i j's træ.)

Hurtig forening

Union og Find afhænger af dybden af træet.

Dårlig nyhed: Dybden kan være $n - 1$.

Udfordring: Kan vi sætte træerne snedigt sammen så vi begrænser dybden?



Foren og find

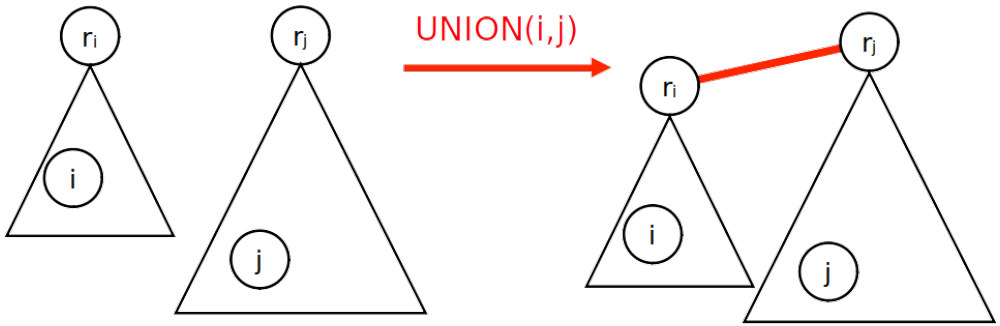
- Foren og find
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter

Vægtet forening

Vægtet forening: (weighted quick union) udvidelse af hurtig forening.
Vedligehold ekstra array sz[0..n-1] således at sz[i] = størrelse (antal elementer) af deltræet bestående af i og dens efterkommere.

Init	som før + initialisér sz[0..n-1]
Find	som før
Union(i,j)	hvis Find(i) $\neq$ Find(j), tag roden af det mindste af de to træer og gør det til barn af roden af det største træ. Opdater sz.

Intuition: Vægtet forening balancerer træerne.



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

Vægtet forening – analyse

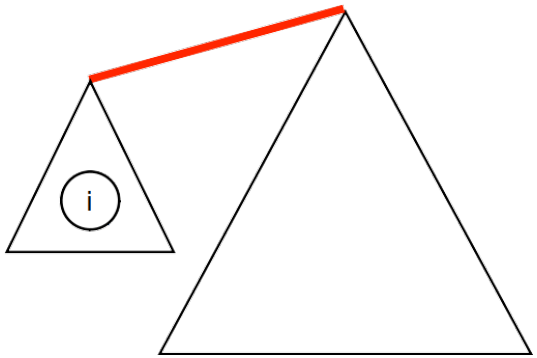
Vægtet forening: Det mindste træs rod bliver barn af roden af det største træ.

Lemma: Med vægtet forening er dybden af en knude højst $\log_2 n$.

Bevis: Kig på en knude i med dybde d_i.

Til at starte med er $d_i = 0$.

d_i øges med 1 når træet med i forenes med et større træ.



Efter en sådan forening er det nye træ mindst dobbelt så stort.

Hvor mange gange kan vi fordoble, før vi har størrelse n? $\log_2 n$.

Derfor: i har dybde $\leq \log_2 n$.

Foren 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 man gøre det bedre?

Foren og find

- Foren og find
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter

Stikompression

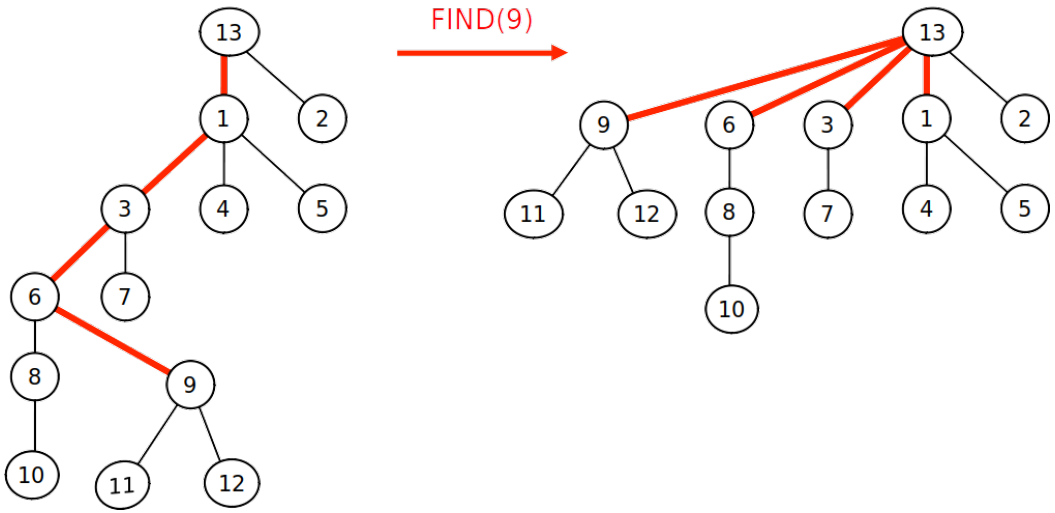
Stikompression: Komprimér stien ved Find

Dvs: alle knuder på stien bliver barn af rodknuden.

Ændrer ikke på tid for en Find operation.

Bonus: Efterfølgende Find operationer bliver hurtigere.

Virker både med hurtig forening og vægtet forening.

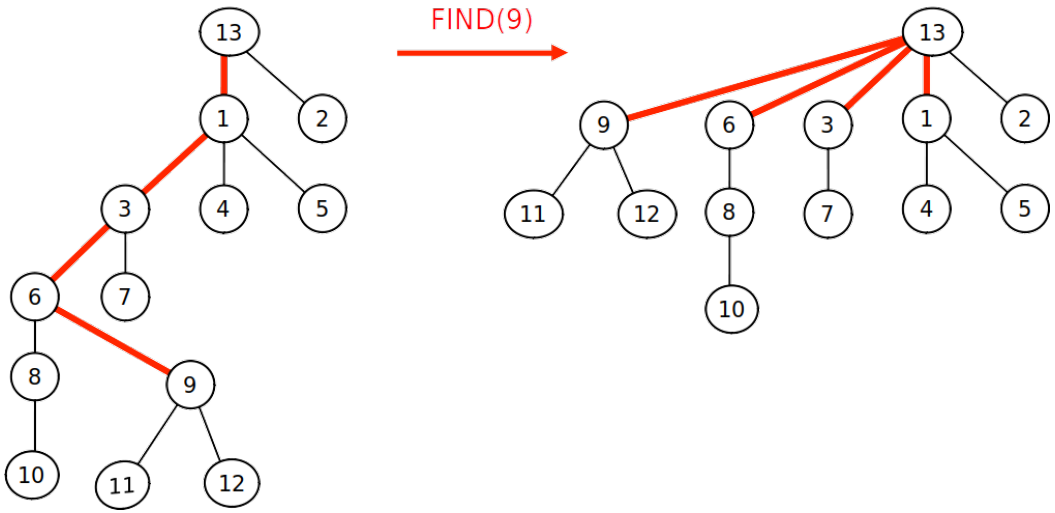


Stikompression

Teorem [Tarjan 1975]: Med stikompression tager enhver sekvens af m Find og Union operationer over n elementer $O(n + m\alpha(m, n))$ tid.

$\alpha(m, n)$ er den inverse til Ackermanns funktion. (α vokser meget langsomt.)

Teorem [Fredman-Saks 1985]: Det er ikke muligt at understøtte m Find og Union operationer over n elementer i $O(n + m)$ tid.



Foren og find

Datastruktur	Samlet tid m foreninger
hurtig find	$O(mn)$
hurtig forening	$O(mn)$
vægtet forening	$O(n + m \log n)$
vægtet+stikompr.	$O(n + m\alpha(m, n))$

Spørgsmål: Kan man opnå $O(n + m)$ samlet tid? Svar: Nej.

Foren og find

- Foren og find
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter

Dynamiske sammenhængskomponenter

Dynamiske sammenhængskomponenter. Vedligehold en dynamisk graf under følgende 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).

Spørgsmål: Effektiv løsning ved brug af foren og find?

Dynamiske sammenhængskomponenter

Dynamiske sammenhængskomponenter. via foren og find.

Init(n)	Init(n)	(opret foren-og-find-struktur).
Insert(u,v)	Union(u,v)	(foren u og v's mængder).
Connected(u,v)	Find(u)=Find(v)	('ja' når de er i samme mængde).

Tid: Samme tid som foren og find!

I denne lektion har vi set, at:

vi kan lave init i $O(n)$ tid, og Insert og Connected i $O(\log n)$ tid.

Foren og find

- Foren og find
- Hurtig find
- Hurtig forening
- Vægtet forening
- Stikompression
- Dynamiske sammenhængskomponenter