2.5 Bäume

Suche wurde ursprünglich formuliert als finden in Mengen
=> Union-Find Srukturen optimiert auf Mengenzugehörigkeitsberechnung

Union-Find-Strukturen

- Datenstruktur zur Darstellung von disjunkten Mengen.
- Jede Menge A wird durch einen Repräsentant $x \in A$ identifiziert.
- Operationen:
  – MakeSet
  – Union
  – Find

Viele (!) disjunkte Mengen
Fragestellung: Sind zwei Elemente in gleicher Menge?
⇒ Haben zwei Elemente den gleichen Repräsentanten?
- MakeSet: Konstruktor, erzeugt einelementige Menge
- Union: Vereinigung von zwei Mengen
- Find: Finde Repräsentanten der Menge

Union-Find-Strukturen

- MakeSet(x)
  - Erzeugt eine neue Menge S_x deren einziges Element x ist.
  - Der Repräsentant von MakeSet(x) ist x.
  - Disjunktheit →
    - x darf nicht bereits in einer anderen Menge enthalten sein

Union-Find-Strukturen

- z = Union(x,y)
  - Erzeugt die Vereinigung $S_x \cup S_y$ der Mengen S_x und S_y (x und y müssen keine Repräsentanten sein!)
  - z ist der Repräsentant der Vereinigung
  - Disjunktheit →
    - S_x und S_y müssen disjunkt sein
    - S_x und S_y werden zerstört

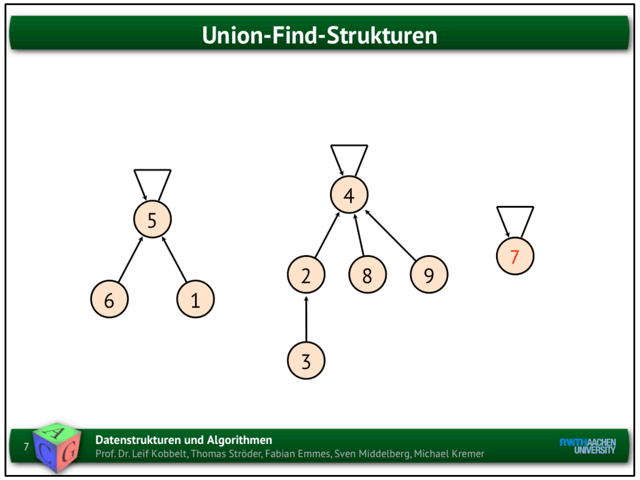
Union-Find-Strukturen

- z = Find(x)
  - Liefert den Repräsentant der Menge, die x enthält

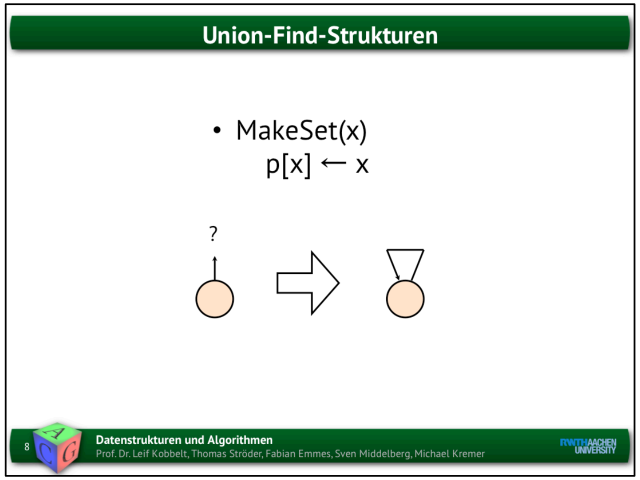
Union-Find-Strukturen

- Idee
  - Jede Menge A wird durch einen Baum dargestellt.
  - Die Knoten des Baums enthalten die Elemente von A.
  - Die Wurzel des Baums enthält den Repräsentant von A.
  - Implementierung: Jeder Knoten hält einen Zeiger auf seinen Vater.

Vater von Wurzel ist Wurzel selbst!
$\Rightarrow$ einziger Knoten, dessen Vater er selbst ist, ist die Wurzel
$\Rightarrow$ Eindeutige Identifikation der Wurzel!



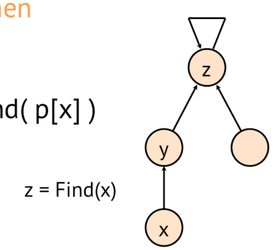
3 disjunkte Mengen, Repräsentanten 5, 4 und 7!



O(1) Aufwand

Union-Find-Strukturen

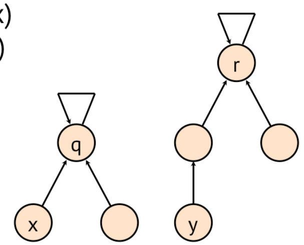
- Find(x)
  if x = p[x] then
    return x
  else
    return Find(p[x])



Rekursives Aufsteigen zur Wurzel
Aufwand: O(log |S_x|), wenn Bäume balanciert sind

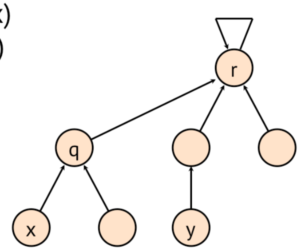
Union-Find-Strukturen

- Union(x,y)
  q ← Find(x)
  r ← Find(y)
  p[q] ← r
  return q



Union-Find-Strukturen

- Union(x,y)
  q ← Find(x)
  r ← Find(y)
  p[q] ← r
  return r



Setze Teilbaum mit Repräsentanten q, als Sohn von r
Problem: Wie Balance halten?

Union-Find-Strukturen

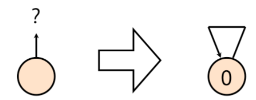
- Laufzeitverbesserungen
  1. Höhenbalancierung
  2. Pfadkompression

Höhenbalancierung

- Speichere in jedem Element x die Höhe h[x] des darunterhängenden Baumes
- Hänge bei Vereinigung stets die niedrigeren Bäume an die höheren und aktualisiere ggfs die Höhe der Wurzel.

Höhenbalancierung

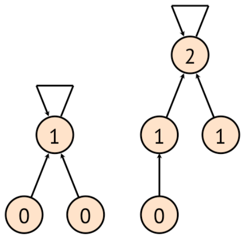
- MakeSet(x)
  $p[x] \leftarrow x$
  $h[x] \leftarrow 0$



Höhenbalancierung

- Union(x,y)

```
q ← Find(x)
r ← Find(y)
if h[q] > h[r] then
    p[r] ← q
    return q
else
    p[q] ← r
    if h[q] = h[r] then
        h[r] ← h[r] + 1
    return r
```

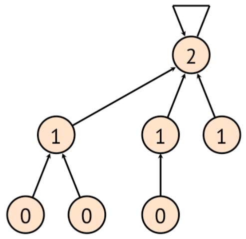


Prof. Dr. Leif Kobbelt, Thomas Ströder, Fabian Emmes, Sven Middelberg, Michael Kremer
RWTH AACHEN UNIVERSITY

Höhenbalancierung

- Union(x,y)

```
q ← Find(x)
r ← Find(y)
if h[q] > h[r] then
    p[r] ← q
    return q
else
    p[q] ← r
    if h[q] = h[r] then
        h[r] ← h[r] + 1
    return r
```



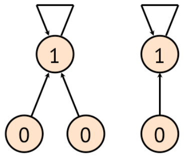
Prof. Dr. Leif Kobbelt, Thomas Ströder, Fabian Emmes, Sven Middelberg, Michael Kremer
RWTH AACHEN UNIVERSITY

Wir hängen niedrigeren Teilbaum an höheren.

Höhenbalancierung

- Union(x,y)

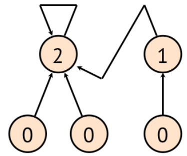
```
q ← Find(x)
r ← Find(y)
if h[q] > h[r] then
   p[r] ← q
   return q
else
   p[q] ← r
   if h[q] = h[r] then
      h[r] ← h[r] + 1
   return r
```



Höhenbalancierung

- Union(x,y)

```
q ← Find(x)
r ← Find(y)
if h[q] > h[r] then
   p[r] ← q
   return q
else
   p[q] ← r
   if h[q] = h[r] then
      h[r] ← h[r] + 1
   return r
```

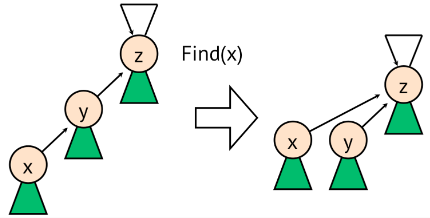
Falls Bäume die gleiche Höhe haben, wird die Gesamthöhe um 1 erhöht

Pfadkompression

- Hänge nach jeder Find(x) Operation die Elemente auf dem Pfad zu x an die Wurzel.

Pfadkompression

- Find(x)
  if x ≠ p[x] then
    p[x] ← Find(p[x])
  return p[x]



Union-Find-Strukturen

- Aufwand (ohne Beweis):
  Sei n die Zahl der MakeSet() Operationen und m die Zahl aller MakeSet(), Union() und Find() Operationen. Dann lässt sich der Aufwand bei Verwendung von Höhen-balancierung und Pfadkompression durch
  $$O(m \times a(n))$$
  abschätzen, wobei $a(n) \leqslant 5$ in allen praktischen Fällen.

a(n) ist im Erwartungsfall eine Konstante, im Worst-Case allerdings logarithmisch.