

2.5 Bäume

- 2.5.1 Binäre Suchbäume
- 2.5.2 Optimale Suchbäume
- 2.5.3 Balancierte Bäume
- 2.5.4 Skip-Listen
- 2.5.5 Union-Find-Strukturen



Union-Find-Strukturen

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



- 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

- $z = \text{Union}(x,y)$
 - 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 \rightarrow
 - S_x und S_y müssen disjunkt sein
 - S_x und S_y werden zerstört



Union-Find-Strukturen

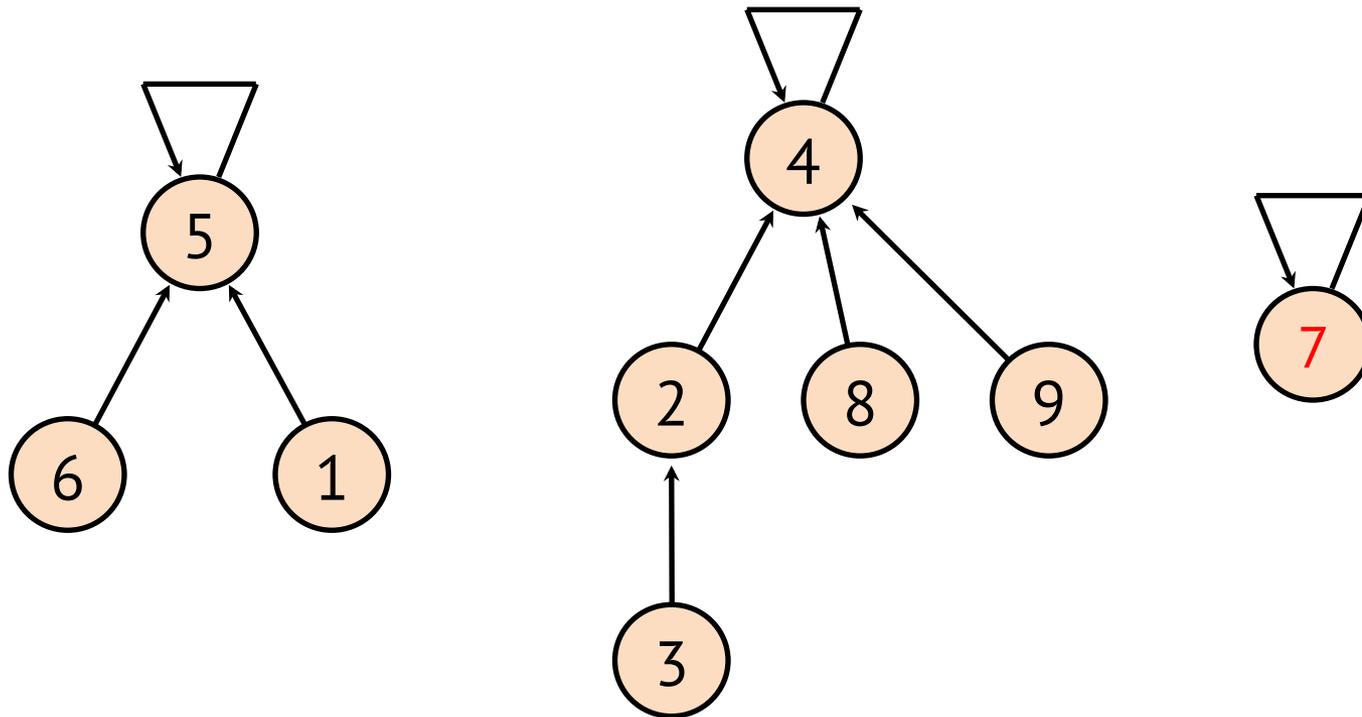
- $z = \text{Find}(x)$
 - Liefert den Repräsentant der Menge, die x enthält



- 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.

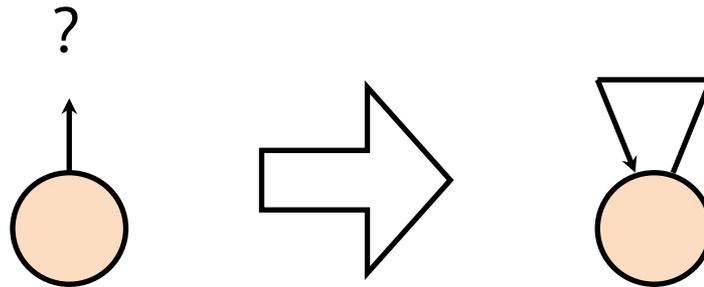


Union-Find-Strukturen



Union-Find-Strukturen

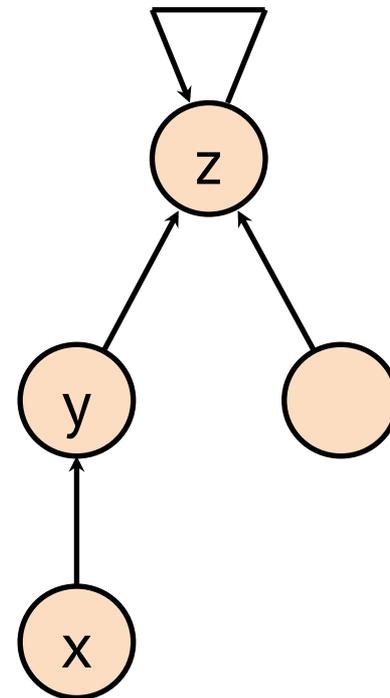
- MakeSet(x)
 $p[x] \leftarrow x$



Union-Find-Strukturen

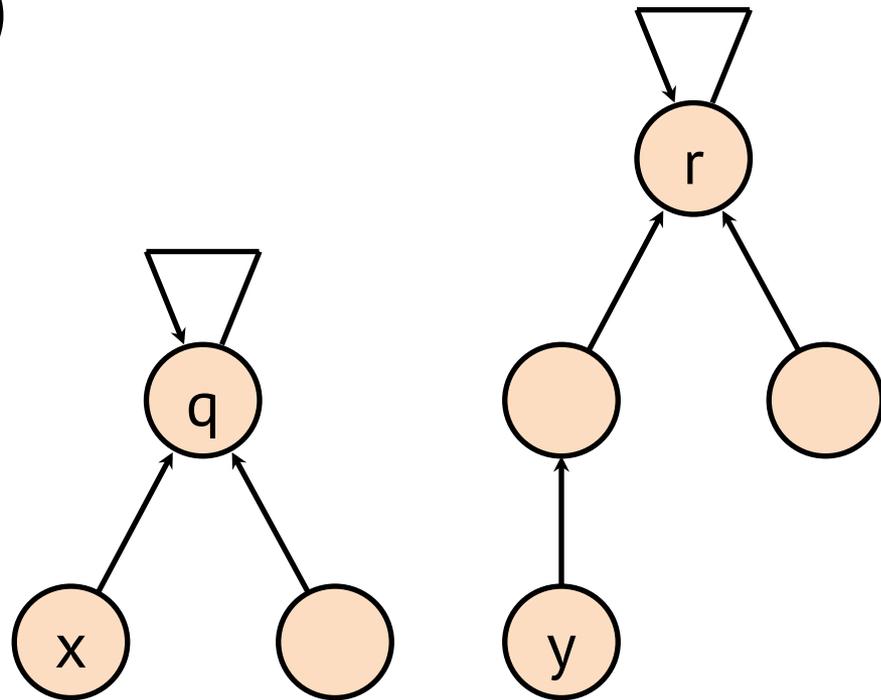
- Find(x)
 - if $x = p[x]$ then
 - return x
 - else
 - return Find(p[x])

$z = \text{Find}(x)$



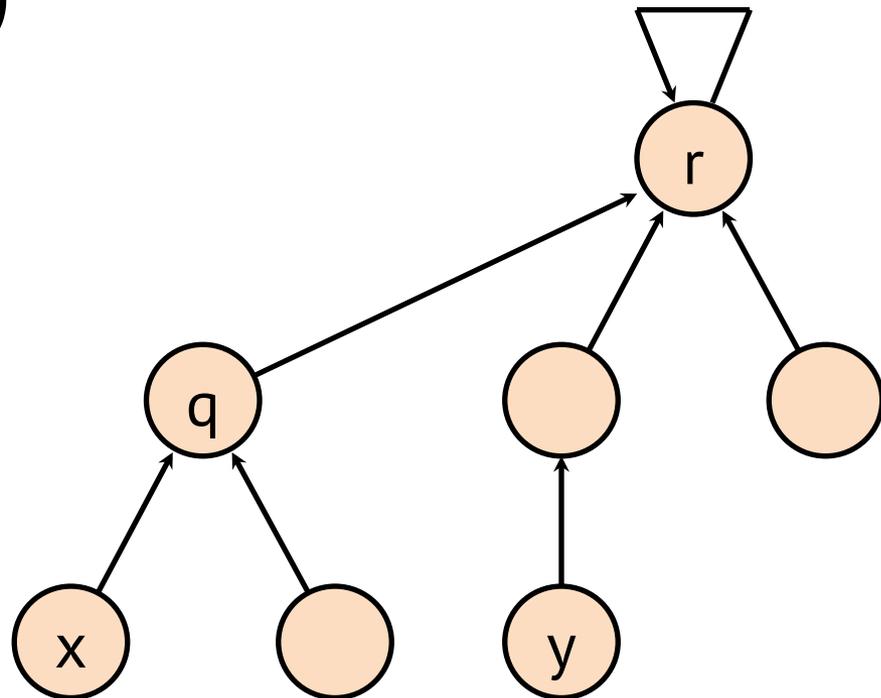
Union-Find-Strukturen

- Union(x,y)
 $q \leftarrow \text{Find}(x)$
 $r \leftarrow \text{Find}(y)$
 $p[q] \leftarrow r$
 return q



Union-Find-Strukturen

- Union(x,y)
 $q \leftarrow \text{Find}(x)$
 $r \leftarrow \text{Find}(y)$
 $p[q] \leftarrow r$
 return r



- 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.

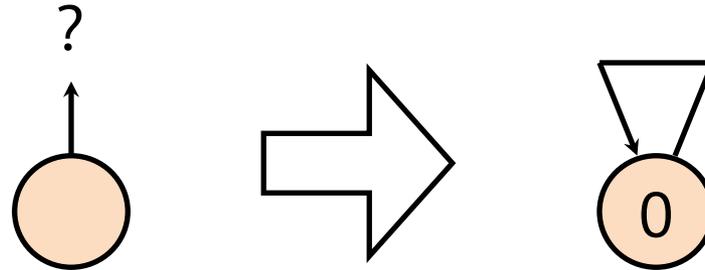


Höhenbalancierung

- MakeSet(x)

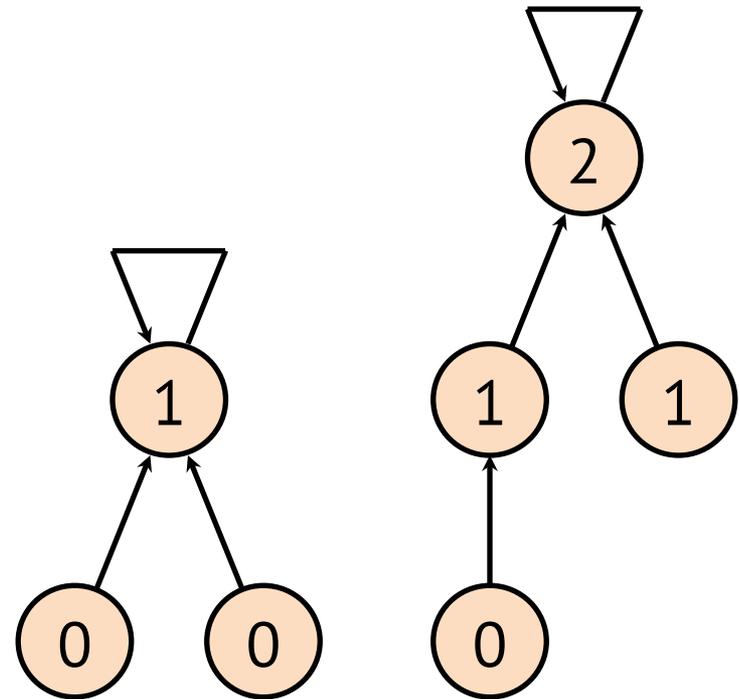
$p[x] \leftarrow x$

$h[x] \leftarrow 0$



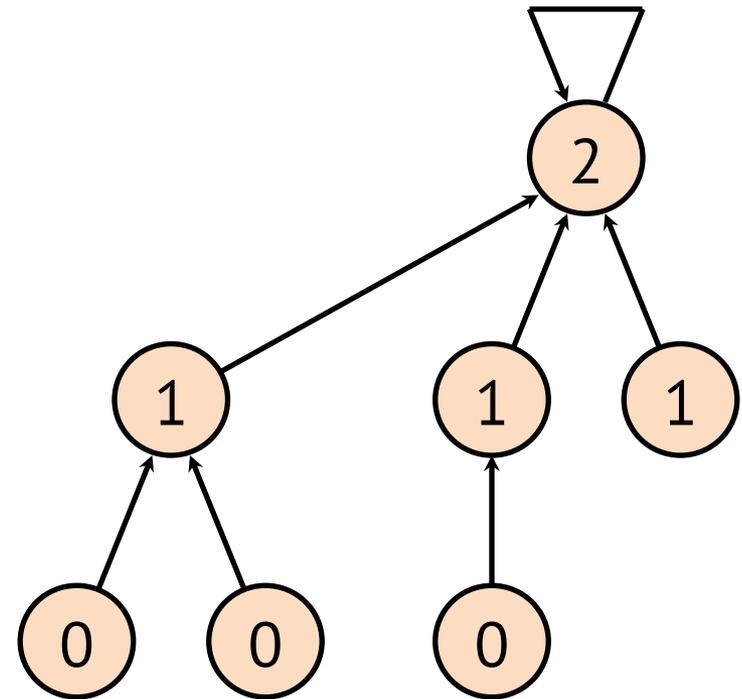
Höhenbalancierung

- Union(x,y)
 - $q \leftarrow \text{Find}(x)$
 - $r \leftarrow \text{Find}(y)$
 - if $h[q] > h[r]$ then
 - $p[r] \leftarrow q$
 - return q
 - else
 - $p[q] \leftarrow r$
 - if $h[q] = h[r]$ then
 - $h[r] \leftarrow h[r] + 1$
 - return r



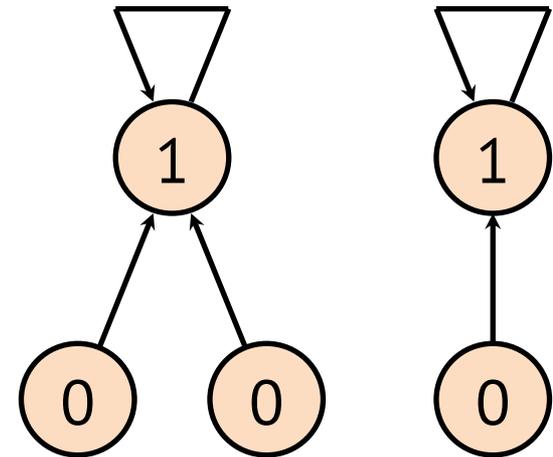
Höhenbalancierung

- Union(x,y)
 - $q \leftarrow \text{Find}(x)$
 - $r \leftarrow \text{Find}(y)$
 - if $h[q] > h[r]$ then
 - $p[r] \leftarrow q$
 - return q
 - else
 - $p[q] \leftarrow r$
 - if $h[q] = h[r]$ then
 - $h[r] \leftarrow h[r] + 1$
 - 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
 - $h[r] \leftarrow h[r] + 1$
 - return r

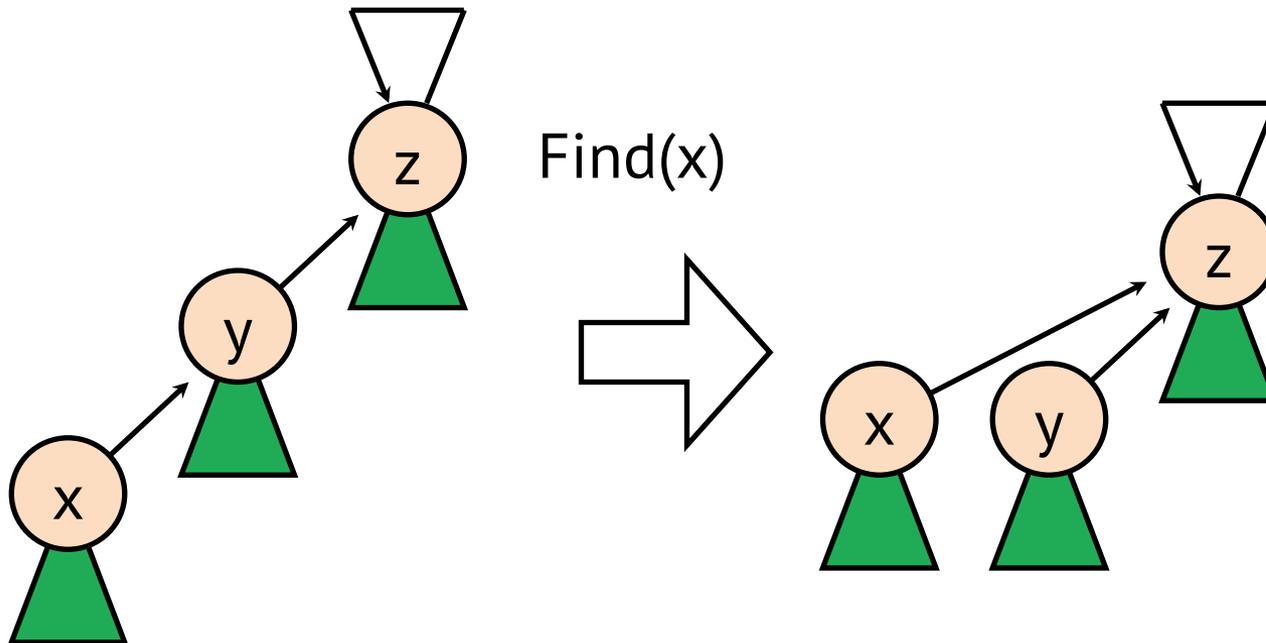


- Hänge nach jeder Find(x) Operation die Elemente auf dem Pfad zu x an die Wurzel.



Pfadkompression

- Find(x)
 if $x \neq p[x]$ then
 $p[x] \leftarrow \text{Find}(p[x])$
 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 Höhenbalancierung und Pfadkompression durch

$$O(m \times a(n))$$

abschätzen, wobei $a(n) \leq 5$ in allen praktischen Fällen.

