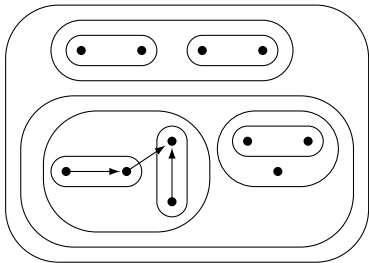


## 8. Union/Find-Datenstrukturen

### 8.1 Motivation

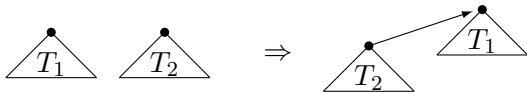


- $Union(T_1, T_2)$ : Vereinige  $T_1$  und  $T_2$   
 $T_1 \cap T_2 = \emptyset$
- $Find(x)$ : Finde den Repräsentanten der (größten) Teilmenge, in der sich  $x$  gerade befindet.

## 8.2 Union/Find-Datenstruktur

### 8.2.1 Intrees

- 1 Initialisierung:  $x \rightarrow \bullet x$ : Mache  $x$  zur Wurzel eines neuen (einelementigen) Baumes.
- 2  $Union(T_1, T_2)$ :



- 3  $Find$ : Suche Wurzel des Baumes, in dem sich  $x$  befindet.

**Bemerkung:** Naive Implementation: worst-case-Tiefe =  $n$

- Zeit für  $Find = \Omega(n)$
- Zeit für  $Union = \mathcal{O}(1)$

## 8.2.2 Gewichtete Union (erste Verbesserung)

Mache die Wurzel des kleineren Baumes zu einem Kind der Wurzel des größeren Baumes. Die Tiefe des Baumes ist dann  $\mathcal{O}(\log n)$ .

- Zeit für *Find* =  $\mathcal{O}(\log n)$
- Zeit für *Union* =  $\mathcal{O}(1)$

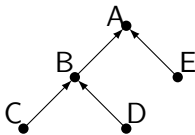
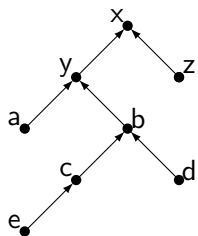
Es gilt auch: Tiefe des Baumes im worst-case:

$$\Omega(\log n)$$

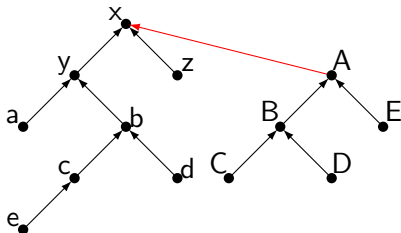
### 8.2.3 Pfad-Kompression mit gewichteter Union (zweite Verbesserung)

Wir betrachten eine Folge von  $k$  *Find*- und *Union*-Operationen auf einer Menge mit  $n$  Elementen, darunter  $n - 1$  *Union*.

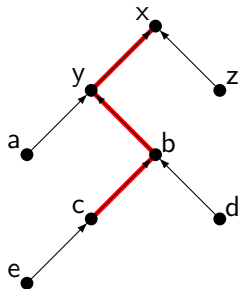
## Implementierung: Gewichtete Union für Pfad-Kompression:



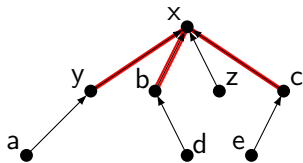
Union  
⇒



## Implementierung: *Find* für Pfad-Kompression:



Find(*c*)  
(Pfadkompression)  
 $\Rightarrow$



## Bemerkung:

Nach Definition ist

$$\log^* n = \min\{i \geq 0; \underbrace{\log \log \log \dots \log n}_{i \text{ log's}} \leq 1\}$$

## Beispiel 73

$$\log^* 0 = \log^* 1 = 0$$

$$\log^* 2 = 1$$

$$\log^* 3 = 2$$

$$\log^* 16 = 3$$

$$\text{da } 16 = 2^{2^2}$$

$$\log^* 2^{65536} = 5$$

$$\text{da } 2^{65536} = 2^{2^{2^{2^2}}}$$

## Satz 74

Bei der obigen Implementierung ergibt sich eine amortisierte Komplexität von  $\mathcal{O}(\log^* n)$  pro Operation.

### Beweis:

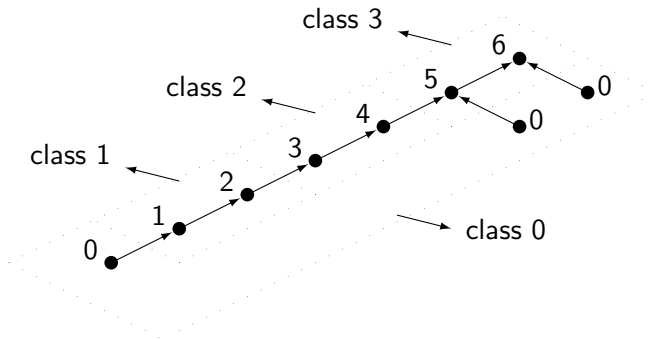
Sei  $T'$  der (endgültige) In-Baum, der durch die Folge der *Union*'s, ohne die *Find*'s, entstehen würde (also keine Pfad-Kompression). Ordne jedem Element  $x$  drei Werte zu:

- $\text{rank}(x) :=$  Höhe des Unterbaums in  $T'$  mit Wurzel  $x$
- $\text{class}(x) := \begin{cases} i \geq 1 & \text{falls } a_{i-1} < \text{rank}(x) \leq a_i \text{ ist } (i \geq 1) \\ 0 & \text{falls } \text{rank}(x) = 0 \end{cases}$

Dabei gilt:  $a_0 = 0, a_i = 2^{2^i}$  für  $i \geq 1$ .

Setze zusätzlich  $a_{-1} := -1$ .





## Beweis (Forts.):

- $\text{dist}(x)$  ist die Distanz von  $x$  zu einem Vorfahr  $y$  im momentanen Union/Find-Baum (mit Pfad-Kompression), so dass  $\text{class}(y) > \text{class}(x)$  bzw.  $y$  die Wurzel des Baumes ist.

Definiere die Potenzialfunktion

$$\text{Potenzial} := c \sum_x \text{dist}(x), \quad c \text{ eine geeignete Konstante } > 0$$

## Beweis (Forts.):

### Beobachtungen:

- i) Sei  $T$  ein Baum in der aktuellen Union/Find-Struktur (mit Pfad-Kompression), seien  $x, y$  Knoten in  $T$ ,  $y$  Vater von  $x$ . Dann ist  $\text{class}(x) \leq \text{class}(y)$ .
- ii) Aufeinander folgende  $\text{Find}(x)$  durchlaufen (bis auf eine) verschiedene Kanten. Diese Kanten sind (im wesentlichen) eine Teilfolge der Kanten in  $T'$  auf dem Pfad von  $x$  zur Wurzel.

## Beweis (Forts.):

### Amortisierte Kosten $\text{Find}(x)$ :

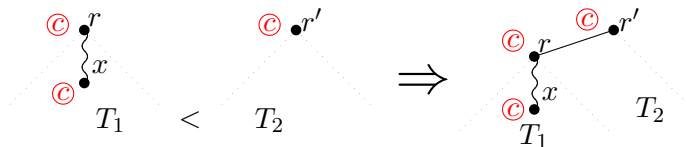
Sei  $x_0 \rightarrow x_1 \rightarrow x_2 \dots x_k = r$  der Pfad von  $x_0$  zur Wurzel. Es gibt höchstens  $\log^* n$ -Kanten  $(x_{i-1}, x_i)$  mit  $\text{class}(x_{i-1}) < \text{class}(x_i)$ . Ist  $\text{class}(x_{i-1}) = \text{class}(x_i)$  und  $i < k$  (also  $x_i \neq r$ ), dann ist  $\text{dist}(x_{i-1})$  vor der  $\text{Find}(x)$ -Operation  $\geq 2$ , nachher gleich 1.

Damit können die Kosten für alle Kanten  $(x_{i-1}, x_i)$  mit  $\text{class}(x_{i-1}) = \text{class}(x_i)$  aus der Potenzialverringerung bezahlt werden. Es ergeben sich damit amortisierte Kosten

$$\mathcal{O}(\log^* n)$$

## Beweis (Forts.):

### Amortisierte Gesamtkosten aller $(n - 1)$ -Union's:



Die gesamte Potenzialerhöhung durch alle *Union*'s ist nach oben durch das Potenzial von  $T'$  beschränkt (Beobachtung ii).

## Beweis (Forts.):

$$\begin{aligned}\text{Potenzial}(T') &\leq c \cdot \sum_{i=0}^{\log^* n} \sum_{\text{rank}(x)=j=a_{i-1}+1}^{a_i} \text{dist}(x) \\ &\leq c \cdot \sum_{i=0}^{\log^* n} \sum_{\text{rank}(x)=j=a_{i-1}+1}^{a_i} \frac{n}{2^j} a_i \\ &\leq c \cdot n \sum_{i=0}^{\log^* n} a_i \frac{1}{2^{a_{i-1}}} = c \cdot n \sum_{i=0}^{\log^* n} 1 \\ &= \mathcal{O}(n \log^* n).\end{aligned}$$

Die zweite Ungleichung ergibt sich, da alle Unterbäume, deren Wurzel  $x$   $\text{rank}(x) = j$  hat, disjunkt sind und jeweils  $\geq 2^j$  Knoten enthalten. □

## 8.2.4 Erweiterungen

- 1) Bessere obere Schranke  $\alpha(k, n)$ ,  $k \geq n$ . Betrachte die (Variante der) Ackermannfunktion  $A(m, n)$  mit:

$$A(0, n) = 2n; \quad n \geq 0$$

$$A(m, 0) = 2; \quad m \geq 1$$

$$A(m + 1, n + 1) = A(m, A(m + 1, n))$$

	$n \rightarrow$					
		0	2	4	6	8
$m \downarrow$		2	4	8	16	32
		2	8	$2^9$		
		2				
		2				
		$\vdots$				

Die Ackermannfunktion  $A(\cdot, \cdot)$  steigt asymptotisch schneller als jede primitiv-rekursive Funktion.

## Definition 75

Die Klasse der primitiv-rekursiven Funktionen (auf den natürlichen Zahlen) ist induktiv wie folgt definiert:

- 1 Alle konstanten Funktionen sind primitiv-rekursiv.
- 2 Alle Projektionen sind primitiv-rekursiv.
- 3 Die Nachfolgerfunktion auf den natürlichen Zahlen ist primitiv-rekursiv.
- 4 Jede Funktion, die durch Komposition von primitiv-rekursiven Funktionen entsteht, ist primitiv-rekursiv.
- 5 Jede Funktion, die durch sog. primitive Rekursion aus primitiv-rekursiven Funktionen entsteht, ist primitiv-rekursiv. Primitive Rekursion bedeutet folgendes Schema für die Definition von  $f$ :

$$f(0, \dots) = g(\dots)$$

$$f(n + 1, \dots) = h(f(n, \dots), \dots)$$

wobei  $g, h$  bereits primitiv-rekursive Funktionen sind.



Weiter wird gesetzt:

$$\alpha(k, n) := \min\left\{i \geq 1; A\left(i, \left\lfloor \frac{k}{n} \right\rfloor\right) > \log n\right\}$$

Dann gilt: Der Zeitbedarf für eine Folge von  $k$  *Find*- und *Union*-Operationen auf einer Menge mit  $n$  Elementen, darunter  $n - 1$  *Union*, ist

$$\mathcal{O}(k\alpha(k, n)).$$

Es gilt auch eine entsprechende untere Schranke.

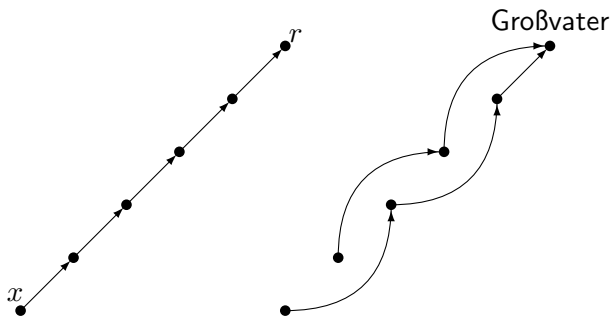


Robert E. Tarjan:

*Data Structures and Network Algorithms*

SIAM CBMS-NSF Regional Conference Series in Applied  
Mathematics Bd. 44 (1983)

## 2) Variante der Pfadkompression:



Diese Variante der **Pfadhalbierung** erfüllt ebenfalls die  $\mathcal{O}(k\alpha(k, n))$  Schranke.