Algorithmen und Datenstrukturen (Th. Ottmann und P. Widmayer)

Folien: Union-Find Strukturen

Autor: Sven Schuierer

Institut für Informatik

Georges-Köhler-Allee

Albert-Ludwigs-Universität Freiburg

1 Union-Find Strukturen

Problem:

Verwaltung einer Familie von disjunkten Mengen $M_1, \ldots, M_k$ unter den Operationen:

$\boldsymbol{e.make\text{-}set(M_i)}$: erzeugt neue Menge M_i mit Element e.

$\boldsymbol{e.find\text{-}set()}$: liefert diejenige Menge M_i, die das Element e enthält.

$\boldsymbol{union(M_i, M_j, M_k)}$: vereinigt die Mengen M_i und M_j zur Menge M_k.

Repräsentation der Mengen M_i:

M_i wird durch ein repräsentatives Element aus M_i identifiziert.

Operationen

Operationen:

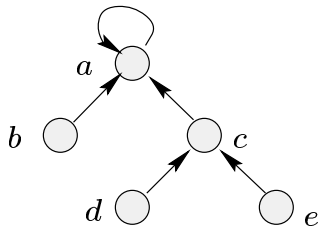
$e.\mathit{make\text{-}set}()$: erzeugt eine neue Menge mit kanonischem Element e. Die Menge wird durch e repräsentiert.

$e.\mathit{find\text{-}set}()$: liefert den Namen des Repräsentanten derjenigen Menge, die das Element e enthält.

$e.\mathit{union}(f)$: vereinigt die Mengen M_e und M_f, die die Elemente e und f enthalten, zu einer neuen Menge M und liefert ein Element aus $M_e \cup M_f$ als Repräsentanten von M. M_e und M_f werden zerstört.

Implementation

Wald:



Array:

x:	a	b	c	d	e
$p[x]$:	a	a	a	c	c

$f.make\text{-}set()$

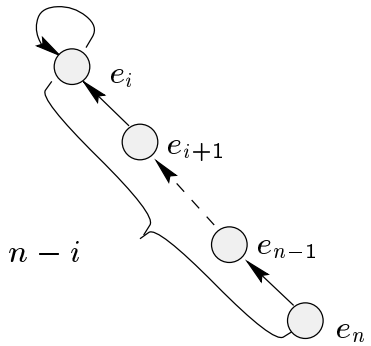
$e.find\text{-}set()$

$e.union(f)$: Mache den Repräsentanten einer Menge (z.B. $a = e.find\text{-}set()$) zum direkten Vater des Repräsentanten der anderen Menge.

Beispiel

$m =$ Gesamtanzahl der Operationen $(\geq 3n)$

for $i = 1$ to n **do** $e_i.make\text{-}set()$
for $i = n$ to 2 **do** $e_{i-1}.union(e_i)$
for $i = 1$ to $m - 2n + 1$ **do** $e_n.find\text{-}set()$



Kosten für $m - 2n + 1$ $Find\text{-}set$ Operationen:

.

Vereinigung nach Größe

$e.size =$ (#Knoten im Teilbaum von e)

e.make-Set( )

1 $e.parent = e$

2 $e.size = 1$

e.union(f)

1 $Link(e.find\text{-}set(), f.find\text{-}set())$

Link(e, f)

1 if $e.size \geq f.size$

2 then $f.parent = e$

$e.size = e.size + f.size$

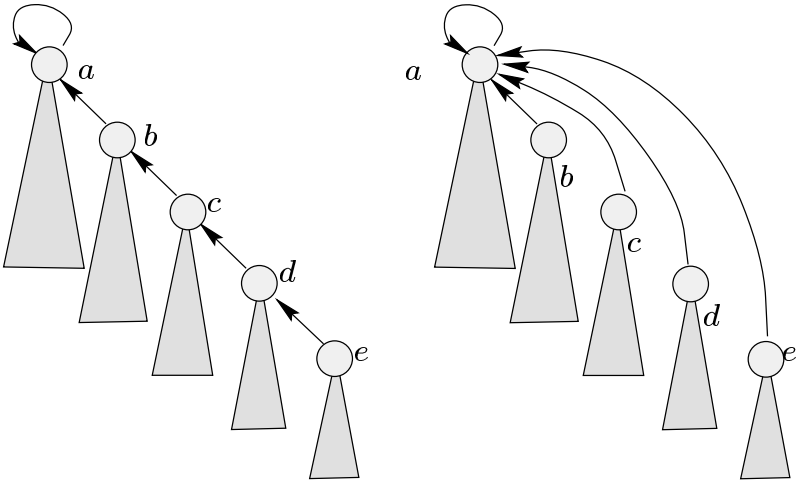
3 else $/* \; e.size < f.size \; */$

4 $e.parent = f$

5 $f.size = e.size + f.size$

2 Pfadverkürzung

$e.find\text{-}set()$




```
e.find-set()
1 if e ≠ e.parent
2   then e.parent = e.parent.find-set()
3 return e.parent
```

Analyse der Laufzeit

m Gesamtanzahl der Operationen, davon

f $find$-Set-Operationen und
n $make$-Set-Operationen.
$\Rightarrow$ höchstens $n-1$ $union$-Operationen

Vereinigung nach Größe:

Vereinigung mit Pfadverkürzung:
Falls $f < n$, $\Theta(n + f \log n)$
Falls $f \geq n$, $\Theta(f \log_{1+f/n} n)$
Beide zusammen:

$\alpha(n, m) =$ Inverse der Ackermann-Funktion

Ackermann Funktion

$$\begin{aligned} A(0,j) &= j+1, && \text{für } j \geq 0 \\ A(i,0) &= A(i-1,1) && \text{für } i \geq 1 \\ A(i,j) &= A(i-1,A(i,j-1)) && \text{für } i,j \geq 1 \end{aligned}$$

Inverse Ackermann Funktion

$$\alpha(m,n) = \min\{i \geq 1 \mid A(i,\lfloor m/n \rfloor) > \log n\}$$

$$\begin{aligned} A(i,\lfloor m/n \rfloor) &\geq A(i,1) \\ A(5,\lfloor m/n \rfloor) &\geq A(5,1) \geq 2^{2^{2^{2}}} = 2^{65536} \end{aligned}$$

$\alpha(m,n) \leq 5$, für $\log n < 2^{65536}$