

# UNION-FIND

---

Verwalten einer Partition einer endlichen Menge  $M$ .

(0) **INIT\_UNION\_FIND**( $M$ ) ... initialisiert mit Partition von  $M$  in einelementige Mengen.

(1) **UNION**( $x, y$ ) ... vereinigt die Mengen mit Namen  $x$  und  $y$  (und gibt einen Namen für die Vereinigung zurück).

(2) Für  $j \in M$ : **FIND**( $j$ ) ... gibt den Namen der Menge zurück, die  $j$  enthält.

$$\underbrace{\{1, 5, 6\}}_{\text{Menge 5}}, \underbrace{\{2, 3, 9, 8\}}_3, \underbrace{\{4\}}_4, \underbrace{\{7, 0\}}_7$$

$\text{FIND}(9) = 3, \text{FIND}(7) = 7, \text{FIND}(2) = 3, \dots$

Nach **UNION**(4, 5):

$$\underbrace{\{1, 5, 6, 4\}}_5, \underbrace{\{2, 3, 9, 8\}}_3, \underbrace{\{7, 0\}}_7$$

# Zusammenhang von Graphen\_\_\_\_\_

PRE: Graph  $G = (V, E)$ ,  $|V| = n$ ,  
 $E = \{e_1, e_2, \dots, e_m\}$ ,  $e_i = \{u_i, v_i\}$  für  $1 \leq i \leq m$ ,  
UNION-FIND für  $V$  initialisiert.

```
c := n;  
for i := 1 to m do  
    if FIND(ui) ≠ FIND(vi) then  
        c := c - 1;  
        UNION(FIND(ui), FIND(vi));
```

POST:  $c$  ... # Zusammenhangskomponenten,  
UNION-FIND-Datenstruktur hält Partition in Zusammenhangskomponenten.

Analyse:  $n - c$  UNIONS,  $2m + 2(n - c)$  FINDs.

# Kruskal- oder Greedy-Algorithmus\_\_\_\_\_

PRE: Gew. Graph  $G = (V, E, w)$ ,  $w : E \rightarrow \mathbb{R}$ ,  
 $|V| = n$ ,  $E = \{e_1, e_2, \dots, e_m\}$ ,  $w(e_i) \leq w(e_{i+1})$ ,  
für  $1 \leq i < m$ , und  $e_i = \{u_i, v_i\}$ , für  $1 \leq i \leq m$ .  
UNION-FIND für  $V$  initialisiert.

```
 $F := \emptyset;$   
for  $i := 1$  to  $m$  do  
  if  $\text{FIND}(u_i) \neq \text{FIND}(v_i)$  then  
     $F := F \cup \{e_i\};$   
     $\text{UNION}(\text{FIND}(u_i), \text{FIND}(v_i));$ 
```

POST:  $(V, F)$  minimal aufspannender Baum von  $G$ ,  
falls  $G$  zusammenhängend ( $\Leftrightarrow |F| = n - 1$ ).

Analyse: Sortieren der Kanten  $O(m \log m)$ ,  
 $n - 1$  UNIONS,  $2m + 2(n - 1)$  FINDs.