AULA 3

Conectividade dinâmica

1.5 Case Study: Union-Find

Leitura, vídeos, ...

Leitura: Case Study: Union-Find, S&W

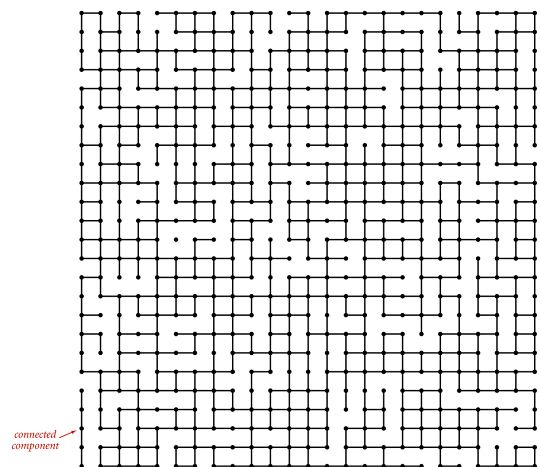
Vídeos: *Union-find* e Kruskal, Gabriel Russo, canal BCC e *Union-find*, Robert Sedgewick.

Considere uma coleção de conjuntos disjuntos $S_1, S_2, \ldots, S_n$.

Conjuntos são modificados ao longo do tempo.

Terminologia utiliza metáfora de redes: sítios/*sites*, conexão,. . .

Problema: p e q estão ligados?



Conjuntos disjuntos

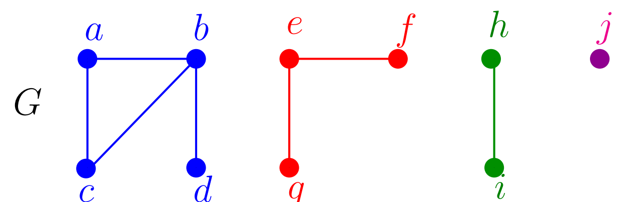
Seja $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ uma coleção de conjuntos disjuntos, ou seja,

$$S_i \cap S_j = \emptyset$$

para todo $i \neq j$.

Conjuntos disjuntos

Exemplo de coleção disjunta de conjuntos: **componentes conexos** de um grafo



componentes formam conjuntos disjuntos de vértices

$\{a, b, c, d\}$ $\{e, f, g\}$ $\{h, i\}$ $\{j\}$

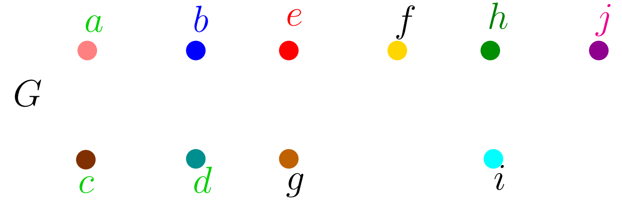
Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: grafo dinâmico

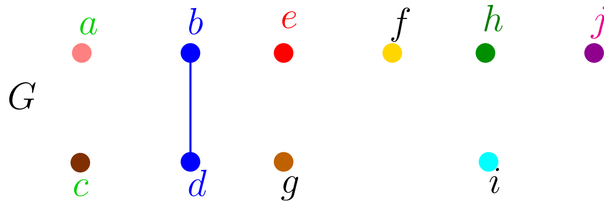


aresta	componentes
	$\{a\}$ $\{b\}$ $\{c\}$ $\{d\}$ $\{e\}$ $\{f\}$ $\{g\}$ $\{h\}$ $\{i\}$

Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: grafo dinâmico

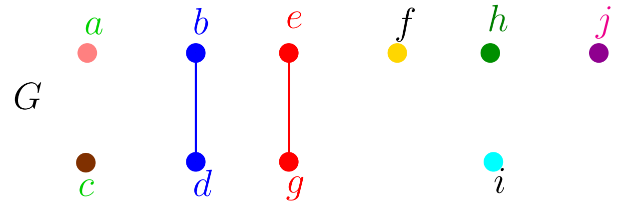


aresta	componentes
(b, d)	$\{a\}$ $\{b, d\}$ $\{c\}$ $\{e\}$ $\{f\}$ $\{g\}$ $\{h\}$ $\{i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: grafo dinâmico

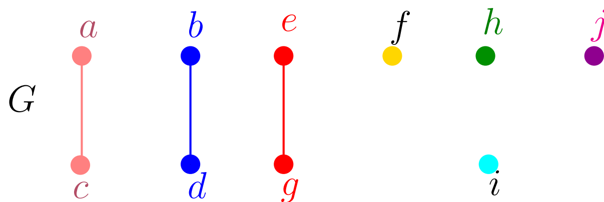


aresta	componentes
(e, g)	$\{a\}$ $\{b, d\}$ $\{c\}$ $\{e, g\}$ $\{f\}$ $\{h\}$ $\{i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: grafo dinâmico

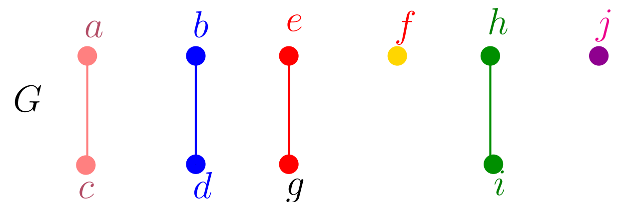


aresta	componentes
(a, c)	$\{a, c\}$ $\{b, d\}$ $\{e, g\}$ $\{f\}$ $\{h\}$ $\{i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: grafo dinâmico

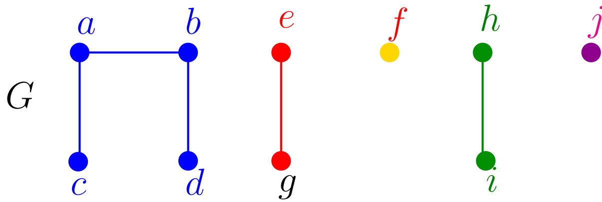


aresta	componentes
(h, i)	$\{a, c\}$ $\{b, d\}$ $\{e, g\}$ $\{f\}$ $\{h, i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: grafo dinâmico

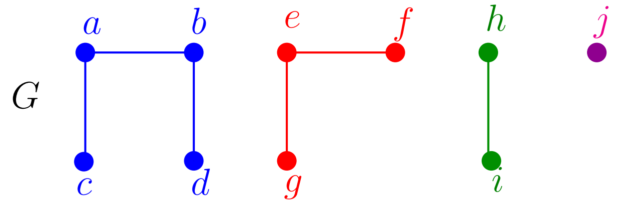


aresta	componentes
(a, b)	$\{a, b, c, d\}$ $\{e, g\}$ $\{f\}$ $\{h, i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: grafo dinâmico

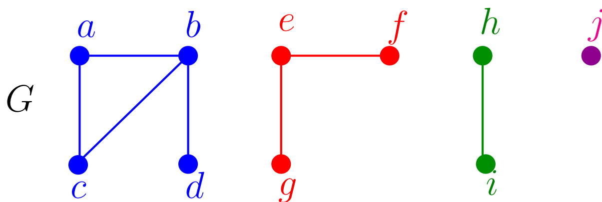


aresta	componentes
(e, f)	$\{a, b, c, d\}$ $\{e, f, g\}$ $\{h, i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: grafo dinâmico



aresta	componentes
(b, c)	$\{a, b, c, d\}$ $\{e, f, g\}$ $\{h, i\}$ $\{j\}$

Operações básicas

$\mathcal{S}$ coleção de conjuntos disjuntos.

Cada conjunto tem um **representante**.

MAKESET (x): x é elemento novo
$\mathcal{S} \leftarrow \mathcal{S} \cup \{\{x\}\}$

Operações básicas

$\mathcal{S}$ coleção de conjuntos disjuntos.

Cada conjunto tem um **representante**.

MAKESET (x): x é elemento novo
$\mathcal{S} \leftarrow \mathcal{S} \cup \{\{x\}\}$

UNION (x, y): x e y em conjuntos diferentes
$\mathcal{S} \leftarrow \mathcal{S} - \{S_x, S_y\} \cup \{S_x \cup S_y\}$

Operações básicas

$\mathcal{S}$ coleção de conjuntos disjuntos.

Cada conjunto tem um **representante**.

MAKESET (x): x é elemento novo
$\mathcal{S} \leftarrow \mathcal{S} \cup \{\{x\}\}$

UNION (x, y): x e y em conjuntos diferentes
$\mathcal{S} \leftarrow \mathcal{S} - \{S_x, S_y\} \cup \{S_x \cup S_y\}$
x está em S_x e y está em Sy

FINDSET (x): devolve representante do conjunto que contém x

Connected-Components

Recebe um grafo G e contrói uma representação dos componentes conexos.

CONNECTED-COMPONENTS (G)

1 **para cada** vértice v de G **faça**
2 MAKESET (v)
3 **para cada** aresta (u, v) de G **faça**
4 **se** FINDSET $(u) \neq$ FINDSET (v)
5 **então** UNION (u, v)

Detalhes de implementação: objeto representando vértice u aponta para para a representação de u como conjunto, e vice-versa.

Consumo de tempo

n := número de vértices do grafo
m := número de arestas do grafo

linha	consumo de todas as execuções da linha
1	$= \Theta(n)$
2	$=$ n $\times$ consumo de tempo MAKESET
3	$= \Theta(m)$
4	$= 2m \times$ consumo de tempo FINDSET
5	$\leq$ n $\times$ consumo de tempo UNION
total	$\leq \Theta(n + m) +$ n $\times$ consumo de tempo MAKESET $+2m \times$ consumo de tempo FINDSET $+n \times$ consumo de tempo UNION

Same-Component

Decide se u e v estão no mesmo componente:

SAME-COMPONENT (u, v)

1 **se** FINDSET $(u) =$ FINDSET (v)
2 **então devolva** SIM
3 **senão devolva** NÃO

Algoritmo de Kruskal

Encontra uma **árvore geradora mínima** (CLRS 23).

MST-KRUSKAL (G, w) ▷ G conexo

−1 coloque arestas em ordem crescente de w
0 $A \leftarrow \emptyset$
1 **para cada** vértice v **faça**
2 MAKESET (v)
3 **para cada** aresta uv em ordem crescente de w **faça**
4 **se** FINDSET $(u) \neq$ FINDSET (v)
5 **então** UNION (u, v)
8 $A \leftarrow A \cup \{uv\}$
9 **devolva** A

"Avô" de todos os algoritmos gulosos.

Conjuntos disjuntos dinâmicos

Sequência de operações MAKESET, UNION, FINDSET

M M M U F U U F U F F F U F

(the first three M's: n; the whole sequence: m)

Que estrutura de dados usar?

Compromissos (*trade-offs*).

API

public class	UF	
	UF(int n)	inicializa n sites com nomes inteiros 0,...,n-1
void	union(int p,int q)	acrescenta ligação entre p e q
int	find(int p)	retorna id do componente de p
boolean	connected(int p,int q)	true se p e q estão no mesmo componente
int	count()	número de componentes

Cliente

```
public static void main(String[] args) {
   int n = StdIn.readInt();
   UF uf = new UF(n);
   while (!StdIn.isEmpty()) {
       int p = StdIn.readInt();
       int q = StdIn.readInt();
       if (uf.connected(p, q)) continue;
       uf.union(p, q);
       StdOut.println(p + " " + q);
   }
   StdOut.println(uf.count()+" comps");
}
```

Quick-find

1.5 Case Study: Union-Find

QuickFindUF

QuickFindUF uf = new QuickFindUF(10);

```
 uf -----+
         |
         V
+-----------------------------------------------------+
| id (private)                    count: 10 (private) |
|  |                                                  |
|  |    +---+---+---+---+---+---+---+---+---+---+     |
|  +--> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |     |
|       +---+---+---+---+---+---+---+---+---+---+     |
|         0   1   2   3   4   5   6   7   8   9       |
|                                                     |
| Métodos: cont(), connected(), find(), union()       |
|                                                     |
+-----------------------------------------------------+
```

QuickFindUF

uf.find(3) retorna 3
uf.find(0) retorna 0

```
uf -----+
        |
        V
+-----------------------------------------------------+
| id (private)                    count: 10 (private) |
|  |                                                  |
|  |    +---+---+---+---+---+---+---+---+---+---+     |
|  +--> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |     |
|       +---+---+---+---+---+---+---+---+---+---+     |
|         0   1   2   3   4   5   6   7   8   9       |
|                                                     |
| Métodos: cont(), connected(), find(), union()       |
|                                                     |
+-----------------------------------------------------+
```

QuickFindUF

uf.union(4, 3);

```
uf -----+
        |
        V
+-----------------------------------------------------+
| id (private)                    count:  9 (private) |
|  |                                                  |
|  |    +---+---+---+---+---+---+---+---+---+---+     |
|  +--> | 0 | 1 | 2 | 3 | 3 | 5 | 6 | 7 | 8 | 9 |     |
|       +---+---+---+---+---+---+---+---+---+---+     |
|         0   1   2   3   4   5   6   7   8   9       |
|                                                     |
| Métodos: cont(), connected(), find(), union()       |
|                                                     |
+-----------------------------------------------------+
```

QuickFindUF

uf.find(3) retorna 3 uf.find(4) retorna 3
uf.union(3, 8);

```
uf -----+
        |
        V
+-----------------------------------------------------+
| id (private)                    count:  8 (private) |
|  |                                                  |
|  |    +---+---+---+---+---+---+---+---+---+---+     |
|  +--> | 0 | 1 | 2 | 8 | 8 | 5 | 6 | 7 | 8 | 9 |     |
|       +---+---+---+---+---+---+---+---+---+---+     |
|         0   1   2   3   4   5   6   7   8   9       |
|                                                     |
| Métodos: cont(), connected(), find(), union()       |
|                                                     |
+-----------------------------------------------------+
```

QuickFindUF

uf.union(6, 5);

```
uf ----+
       |
       V
+----------------------------------------------------+
| id (private)                   count:  7 (private) |
|  |                                                 |
|  |    +---+---+---+---+---+---+---+---+---+---+    |
|  +--> | 0 | 1 | 2 | 8 | 8 | 5 | 5 | 7 | 8 | 9 |    |
|       +---+---+---+---+---+---+---+---+---+---+    |
|         0   1   2   3   4   5   6   7   8   9      |
|                                                    |
| Métodos: cont(), connected(), find(), union()      |
|                                                    |
+----------------------------------------------------+
```

QuickFindUF

uf.union(9, 4);

```
uf ----+
       |
       V
+----------------------------------------------------+
| id (private)                   count:  6 (private) |
|  |                                                 |
|  |    +---+---+---+---+---+---+---+---+---+---+    |
|  +--> | 0 | 1 | 2 | 8 | 8 | 5 | 5 | 7 | 8 | 8 |    |
|       +---+---+---+---+---+---+---+---+---+---+    |
|         0   1   2   3   4   5   6   7   8   9      |
|                                                    |
| Métodos: cont(), connected(), find(), union()      |
|                                                    |
+----------------------------------------------------+
```

QuickFindUF

uf.union(2, 1);

```
uf ----+
       |
       V
+----------------------------------------------------+
| id (private)                   count:  5 (private) |
|  |                                                 |
|  |    +---+---+---+---+---+---+---+---+---+---+    |
|  +--> | 0 | 1 | 1 | 8 | 8 | 5 | 5 | 7 | 8 | 8 |    |
|       +---+---+---+---+---+---+---+---+---+---+    |
|         0   1   2   3   4   5   6   7   8   9      |
|                                                    |
| Métodos: cont(), connected(), find(), union()      |
|                                                    |
+----------------------------------------------------+
```

QuickFindUF

uf.union(8, 9);

```
uf ----+
       |
       V
+----------------------------------------------------+
| id (private)                   count:  5 (private) |
|  |                                                 |
|  |    +---+---+---+---+---+---+---+---+---+---+    |
|  +--> | 0 | 1 | 1 | 8 | 8 | 5 | 5 | 7 | 8 | 8 |    |
|       +---+---+---+---+---+---+---+---+---+---+    |
|         0   1   2   3   4   5   6   7   8   9      |
|                                                    |
| Métodos: cont(), connected(), find(), union()      |
|                                                    |
+----------------------------------------------------+
```

QuickFindUF

uf.union(5, 0);

```
uf ----+
       |
       V
+----------------------------------------------------+
| id (private)                   count:  5 (private) |
|  |                                                 |
|  |    +---+---+---+---+---+---+---+---+---+---+    |
|  +--> | 0 | 1 | 1 | 8 | 8 | 0 | 0 | 7 | 8 | 8 |    |
|       +---+---+---+---+---+---+---+---+---+---+    |
|         0   1   2   3   4   5   6   7   8   9      |
|                                                    |
| Métodos: cont(), connected(), find(), union()      |
|                                                    |
+----------------------------------------------------+
```

QuickFindUF

uf.union(7, 2);

```
uf ----+
       |
       V
+----------------------------------------------------+
| id (private)                   count:  3 (private) |
|  |                                                 |
|  |    +---+---+---+---+---+---+---+---+---+---+    |
|  +--> | 0 | 1 | 1 | 8 | 8 | 0 | 0 | 1 | 8 | 8 |    |
|       +---+---+---+---+---+---+---+---+---+---+    |
|         0   1   2   3   4   5   6   7   8   9      |
|                                                    |
| Métodos: cont(), connected(), find(), union()      |
|                                                    |
+----------------------------------------------------+
```

QuickFindUF

uf.union(6, 1);

```
uf ----+
       |
       V
+-----------------------------------------------------+
| id (private)                     count:  2 (private) |
|  |                                                   |
|  |    +---+---+---+---+---+---+---+---+---+---+     |
|  +--> | 1 | 1 | 1 | 8 | 8 | 1 | 1 | 1 | 8 | 8 |     |
|       +---+---+---+---+---+---+---+---+---+---+     |
|         0   1   2   3   4   5   6   7   8   9       |
|                                                     |
| Métodos: cont(), connected(), find(), union()       |
|                                                     |
+-----------------------------------------------------+
```

Consumo de tempo

UF(n)	$\Theta(n)$
union(p, q)	$O(n)$
find(p)	$\Theta(1)$

Uma sequência de m operações pode consumir tempo $\Theta(m^2)$ no pior caso.

Consumo de tempo amortizado de cada operação é $O(m)$.

Hmm. Em union() seria razoável alterarmos o menor número possível de posições do vetor id. Para isso precisamos saber qual conjunto tem o menor número de itens...

Experimentos

```
% java Driver < tinyUF.txt
2 components
0.002seg

% java Driver < mediumUF.txt
3 components
0.025seg

% java Driver < largeUF.txt
:-(
```

Quick-union

1.5 Case Study: Union-Find

QuickUnionUF

A **ideia** é trocar o indicador id[] do componente por um indicador do pai[] do sítio. Por sua vez, se p é um sítio, o pai pindica quem é avô (pai[pai[p]]), que indica que é o bisavô (pai[pai[pai[p]]]), que indica o sítio tataravô,...

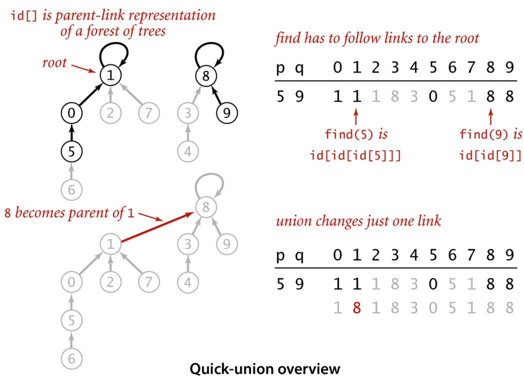
QuickUnionUF

O **representante** ou **nome** de um componente será o sítio que é o pai de si mesmo. Hmm. Aqui a metáfora fica meio estranha...

É intuitivo representarmos a estrutura através de um conjunto de árvores disjuntas (= **floresta**) onde as raizes das árvores são os sítios p tais que

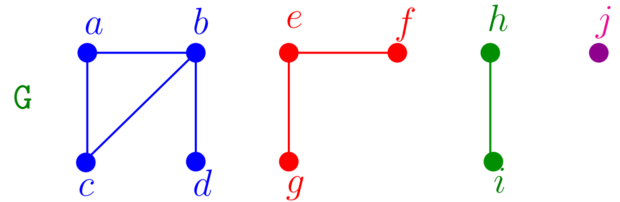
p == pai[p].

Estrutura *disjoint-set forest*



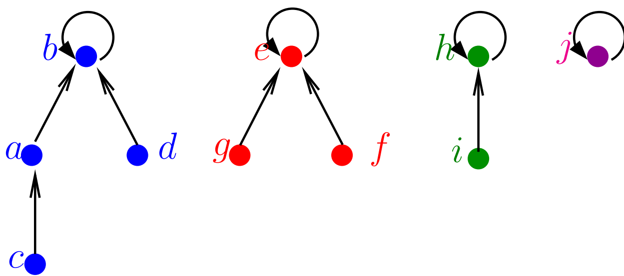
Quick-union overview

Estrutura *disjoint-set forest*



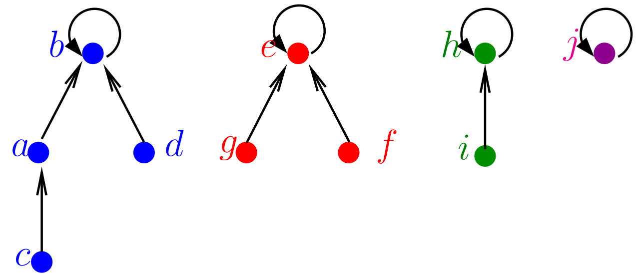
- cada conjunto tem uma `raiz`, que é o seu representante
- cada nó p tem um pai
- pai[p] = p se e só se p é uma raiz

Estrutura *disjoint-set forest*



- cada conjunto tem uma `raiz`
- cada nó p tem um pai
- pai[p] = p se e só se p é uma raiz

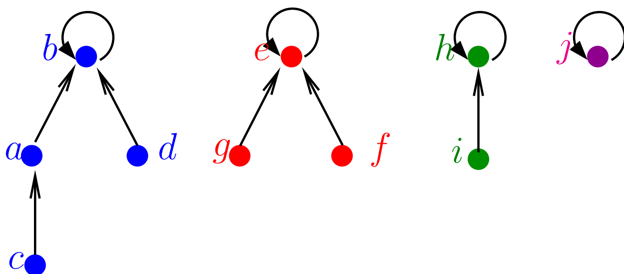
MakeSet e FindSet



MAKESET(p)

1 *pai*[p] ← p

MakeSet e FindSet



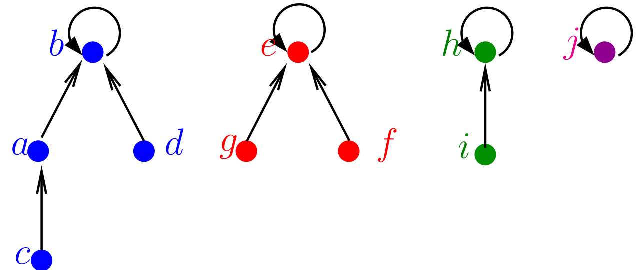
MAKESET(p)

1 *pai*[p] ← p

FINDSET(p)

1 **enquanto** pai[p] ≠ p **faça**
2 p ← pai[p]
3 **devolva** p

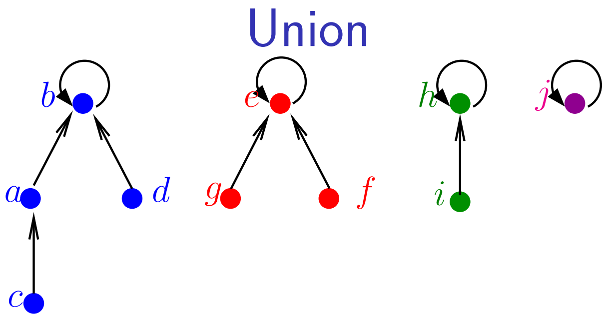
FindSet



FINDSET(p)

1 **se** pai[p] = p
2 **então devolva** p
3 **senão devolva** FINDSET (pai[p])

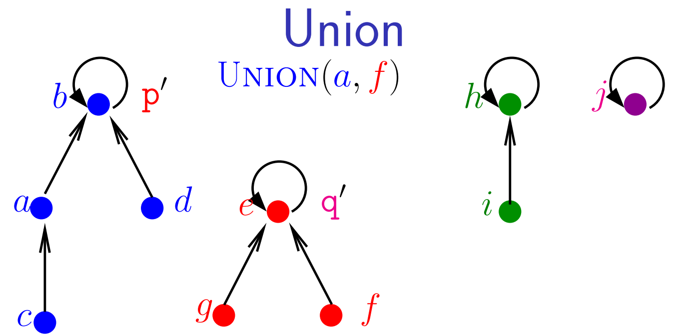
Union



UNION(p, q)
1 $p' \leftarrow$ FINDSET$_0$ (p)
2 $q' \leftarrow$ FINDSET$_0$ (q)
3 pai[q'] $\leftarrow$ p'

Union

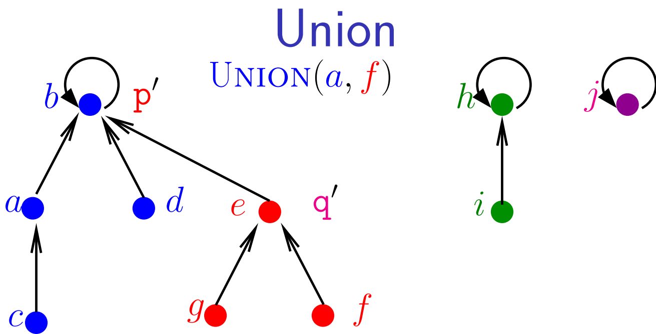
UNION(a, f)



UNION(p, q)
1 $p' \leftarrow$ FINDSET$_0$ (p)
2 $q' \leftarrow$ FINDSET$_0$ (q)
3 pai[q'] $\leftarrow$ p'

Union

UNION(a, f)



UNION(p, q)
1 $p' \leftarrow$ FINDSET$_0$ (p)
2 $q' \leftarrow$ FINDSET$_0$ (q)
3 pai[q'] $\leftarrow$ p'

MakeSet, Union e FindSet

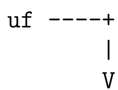
MAKESET(p)
1 *pai*[p] $\leftarrow$ p

UNION(p, q)
1 $p' \leftarrow$ FINDSET$_0$ (p)
2 $q' \leftarrow$ FINDSET$_0$ (q)
3 pai[q'] $\leftarrow$ p'

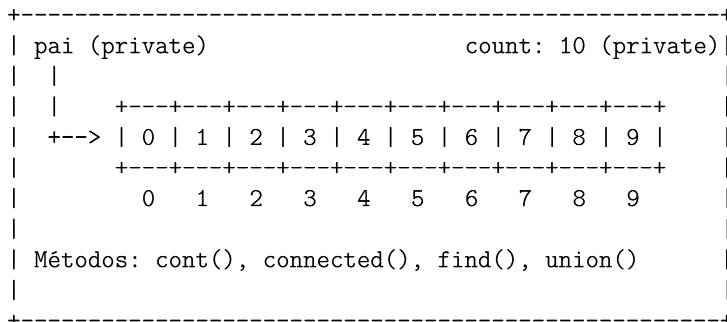
FINDSET(p)
1 **se** pai[p] = p
2 **então devolva** p
3 **senão devolva** FINDSET$_1$ (pai[p])

QuickUnionUF

`QuickUnionUF` uf = `new` `QuickUnionUF`(10);



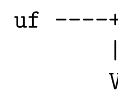
```
uf ----+
       |
       V
```



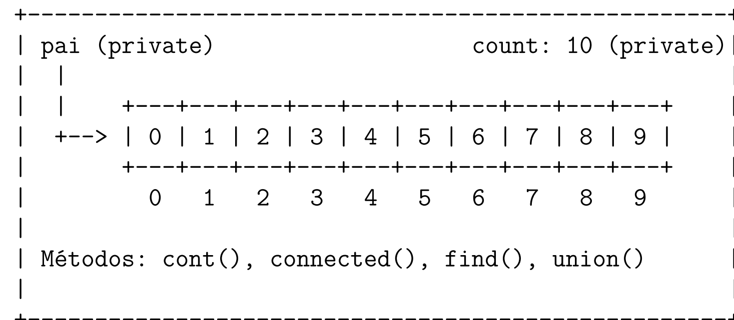
```
+-----------------------------------------------------+
| pai (private)                   count: 10 (private)|
|  |                                                  |
|  |     +---+---+---+---+---+---+---+---+---+---+    |
|  +---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |    |
|        +---+---+---+---+---+---+---+---+---+---+    |
|          0   1   2   3   4   5   6   7   8   9      |
|                                                     |
| Métodos: cont(), connected(), find(), union()       |
|                                                     |
+-----------------------------------------------------+
```

QuickUnionUF

uf.`find`(3) retorna 3
uf.`find`(0) retorna 0



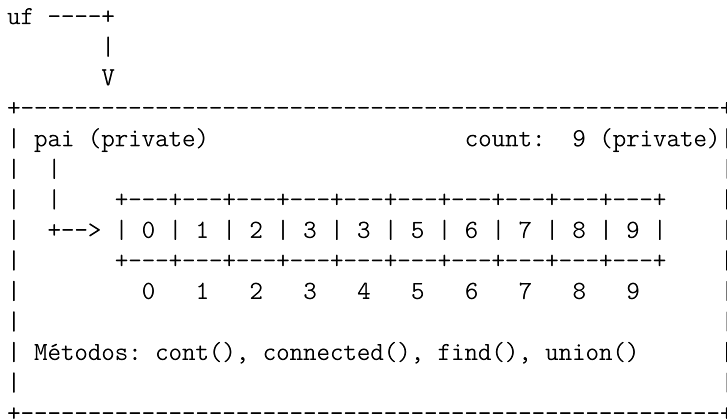
```
uf ----+
       |
       V
```



```
+-----------------------------------------------------+
| pai (private)                   count: 10 (private)|
|  |                                                  |
|  |     +---+---+---+---+---+---+---+---+---+---+    |
|  +---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |    |
|        +---+---+---+---+---+---+---+---+---+---+    |
|          0   1   2   3   4   5   6   7   8   9      |
|                                                     |
| Métodos: cont(), connected(), find(), union()       |
|                                                     |
+-----------------------------------------------------+
```

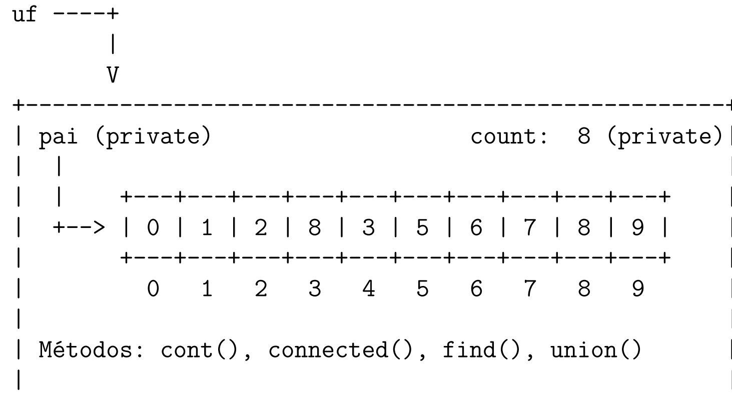
QuickUnionUF

uf.union(4, 3);



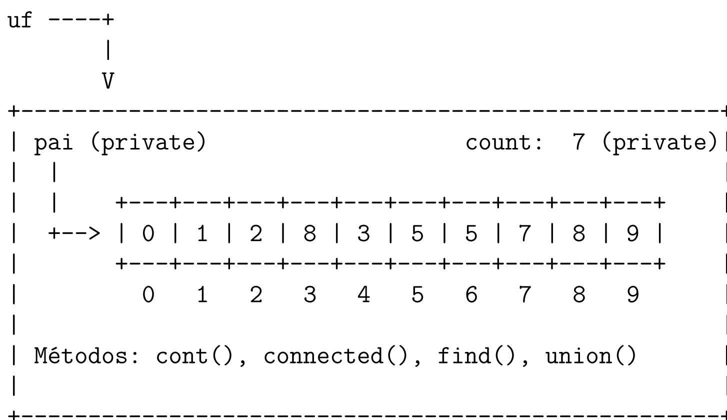
QuickUnionUF

uf.find(3) retorna 3 uf.find(4) retorna 3
uf.union(3, 8);



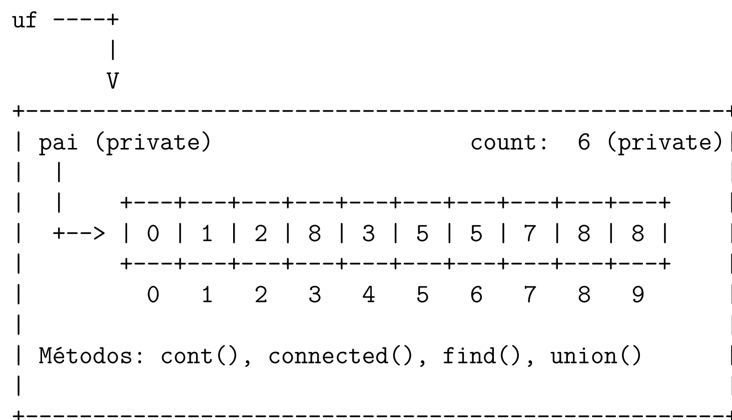
QuickUnionUF

uf.union(6, 5);



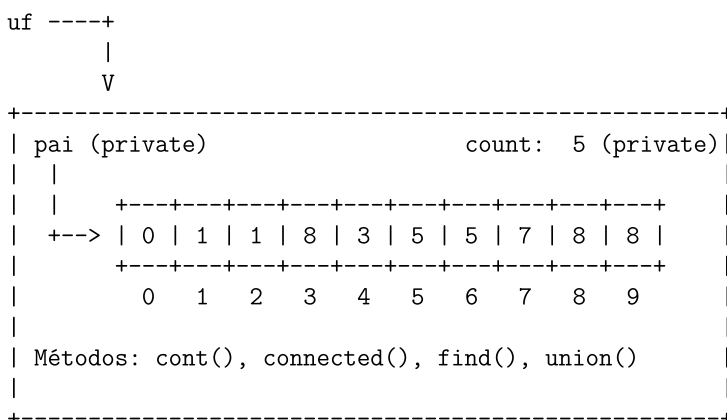
QuickUnionUF

uf.union(9, 4);



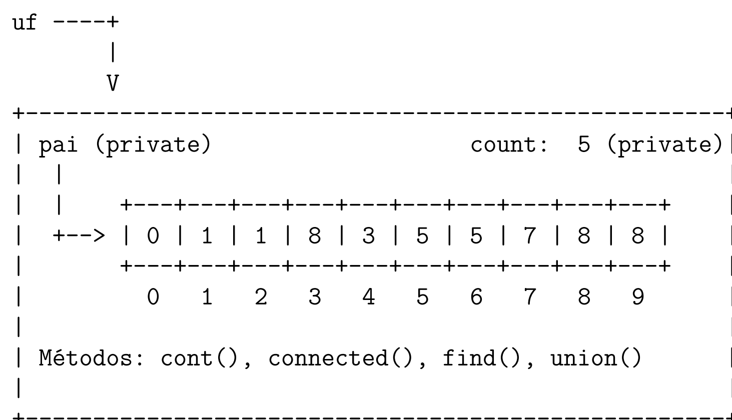
QuickUnionUF

uf.union(2, 1);



QuickUnionUF

uf.union(8, 9);



QuickUnionUF

uf.union(5, 0);

```
uf ----+
       |
       V
+-----------------------------------------------------+
| pai (private)                    count:  4 (private)|
|  |                                                  |
|  |    +---+---+---+---+---+---+---+---+---+---+     |
|  +--> | 0 | 1 | 1 | 8 | 3 | 0 | 5 | 7 | 3 | 3 |     |
|       +---+---+---+---+---+---+---+---+---+---+     |
|         0   1   2   3   4   5   6   7   8   9       |
|                                                     |
| Métodos: cont(), connected(), find(), union()       |
|                                                     |
+-----------------------------------------------------+
```

QuickUnionUF

uf.union(7, 2);

```
uf ----+
       |
       V
+-----------------------------------------------------+
| pai (private)                    count:  3 (private)|
|  |                                                  |
|  |    +---+---+---+---+---+---+---+---+---+---+     |
|  +--> | 0 | 1 | 1 | 8 | 3 | 0 | 5 | 1 | 3 | 3 |     |
|       +---+---+---+---+---+---+---+---+---+---+     |
|         0   1   2   3   4   5   6   7   8   9       |
|                                                     |
| Métodos: cont(), connected(), find(), union()       |
|                                                     |
+-----------------------------------------------------+
```

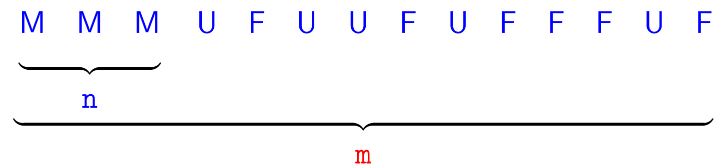
QuickUnionUF

uf.union(6, 1);

```
uf ----+
       |
       V
+-----------------------------------------------------+
| pai (private)                    count:  2 (private)
|  |                                                  |
|  |    +---+---+---+---+---+---+---+---+---+---+     |
|  +--> | 1 | 1 | 1 | 8 | 3 | 0 | 1 | 1 | 3 | 3 |     |
|       +---+---+---+---+---+---+---+---+---+---+     |
|         0   1   2   3   4   5   6   7   8   9       |
|                                                     |
| Métodos: cont(), connected(), find(), union()       |
|                                                     |
+-----------------------------------------------------+
```

Consumo de tempo

UF(n)	$\Theta(n)$
find(p)	$O(n)$
union(p, q)	$O(n)$



Custo total da sequência:

$$n\,\Theta(1) + m\,O(n) + n\,O(n) = O(mn)$$

Experimentos

```
% java Driver < tinyUF.txt
```
2 components
0.002seg

```
% java Driver < mediumUF.txt
```
3 components
0.032seg

```
% java Driver < largeUF.txt
```
:-(

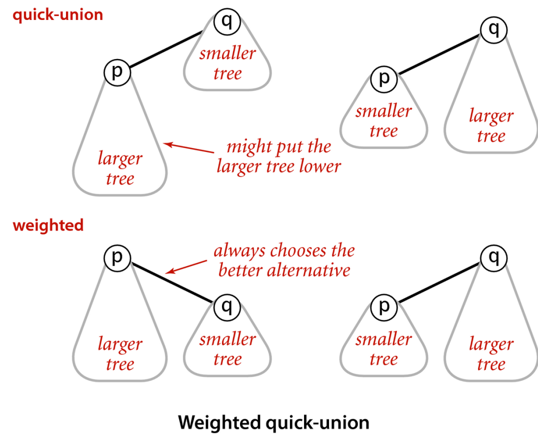
Weight-Quick-union

1.5 Case Study: Union-Find

WeightedQuickUnionUF

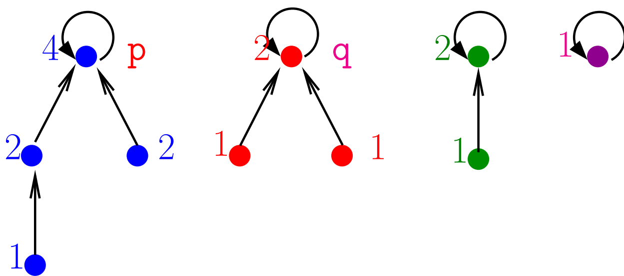
Ideia, ligar a raiz da árvore com menos sítios na raiz da árvore com mais sítios.
Isso seria a política natural para tornarmos o quick-find mais eficiente.

WeightedQuickUnionUF



Weighted quick-union

union by size

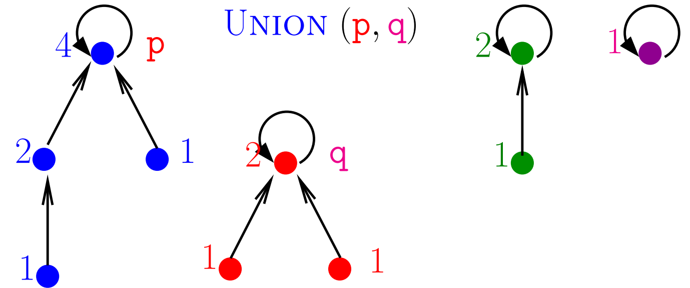


sz[p] = nós em p

MAKESET (p)

1 pai[p] ← p
2 sz[p] ← 0

union by size

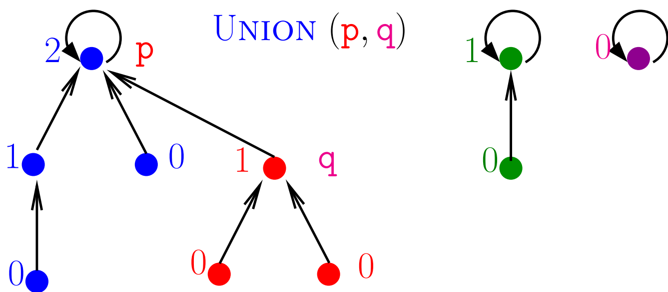


sz[p] = nós em p

MAKESET (p)

1 pai[p] ← p
2 sz[p] ← 0

union by size

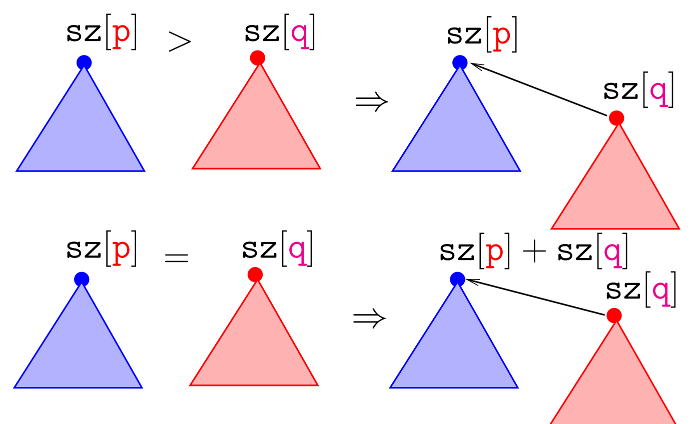


sz[p] = posto do nó p

MAKESET (p)

1 pai[p] ← p
2 sz[p] ← 0

union by size



union by size

UNION (p, q) $\triangleright$ com "union by size"

1. $p' \leftarrow$ FINDSET (p)
2. $q' \leftarrow$ FINDSET (q) $\triangleright$ supõe que $p' \neq q'$
3. **se** $sz[p'] > sz[q']$
4. **então** $pai[q'] \leftarrow p'$
5. $sz[p'] \leftarrow sz[p'] + sz[q']$
6. **senão** $pai[p'] \leftarrow q'$
7. $sz[q'] \leftarrow sz[q'] + sz[p']$

WeightedQuickUnionUF

WeightedQuickUnionUF uf = new WeightedQuickUnionUF(10);

```
uf
+-------------------------------------------------------+
| pai (private)                      count: 0 (private) |
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| sz (private)                                          |
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| Métodos: cont(), connected(), find(), union()         |
+-------------------------------------------------------+
```

WeightedQuickUnionUF

uf.find(3) retorna 3 uf.find(0) retorna 0

```
uf
+-------------------------------------------------------+
| pai (private)                      count: 10 (private)|
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| sz (private)                                          |
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| Métodos: cont(), connected(), find(), union()         |
+-------------------------------------------------------+
```

WeightedQuickUnionUF

uf.union(4, 3);

```
uf
+-------------------------------------------------------+
| pai (private)                      count:  9 (private)|
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 0 | 1 | 2 | 4 | 4 | 5 | 6 | 7 | 8 | 9 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| sz (private)                                          |
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| Métodos: cont(), connected(), find(), union()         |
+-------------------------------------------------------+
```

WeightedQuickUnionUF

uf.union(3, 8);

```
uf
+-------------------------------------------------------+
| pai (private)                      count:  8 (private)|
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 0 | 1 | 2 | 4 | 4 | 5 | 6 | 7 | 4 | 9 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| sz (private)                                          |
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 1 | 1 | 1 | 1 | 3 | 1 | 1 | 1 | 1 | 1 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| Métodos: cont(), connected(), find(), union()         |
+-------------------------------------------------------+
```

WeightedQuickUnionUF

uf.union(6, 5);

```
uf
+-------------------------------------------------------+
| pai (private)                      count:  7 (private) |
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 0 | 1 | 2 | 4 | 4 | 6 | 6 | 7 | 4 | 9 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| sz (private)                                          |
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 1 | 1 | 1 | 1 | 3 | 1 | 2 | 1 | 1 | 1 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| Métodos: cont(), connected(), find(), union()         |
+-------------------------------------------------------+
```

WeightedQuickUnionUF

uf.union(9, 4);

```
uf
+-------------------------------------------------------+
| pai (private)                 count:  6 (private)|
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 0 | 1 | 2 | 4 | 4 | 6 | 6 | 7 | 4 | 4 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| sz (private)                                          |
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 1 | 1 | 1 | 1 | 4 | 1 | 2 | 1 | 1 | 1 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| Métodos: cont(), connected(), find(), union()         |
+-------------------------------------------------------+
```

WeightedQuickUnionUF

uf.union(2, 1);

```
uf
+-------------------------------------------------------+
| pai (private)                 count:  5 (private)|
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 0 | 1 | 1 | 4 | 4 | 6 | 6 | 7 | 4 | 4 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| sz (private)                                          |
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 1 | 2 | 1 | 1 | 3 | 1 | 2 | 1 | 1 | 1 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| Métodos: cont(), connected(), find(), union()         |
+-------------------------------------------------------+
```

WeightedQuickUnionUF

uf.union(8, 9);

```
uf
+-------------------------------------------------------+
| pai (private)                 count:  4 (private)|
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 0 | 1 | 1 | 4 | 4 | 6 | 6 | 7 | 4 | 4 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| sz (private)                                          |
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 1 | 2 | 1 | 1 | 4 | 1 | 2 | 1 | 1 | 1 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| Métodos: cont(), connected(), find(), union()         |
+-------------------------------------------------------+
```

WeightedQuickUnionUF

uf.union(5, 0);

```
uf
+-------------------------------------------------------+
| pai (private)                 count:  3 (private)|
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 6 | 1 | 1 | 4 | 4 | 6 | 6 | 7 | 4 | 4 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| sz (private)                                          |
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 1 | 2 | 1 | 1 | 4 | 1 | 3 | 1 | 1 | 1 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| Métodos: cont(), connected(), find(), union()         |
+-------------------------------------------------------+
```

WeightedQuickUnionUF

uf.union(7, 2);

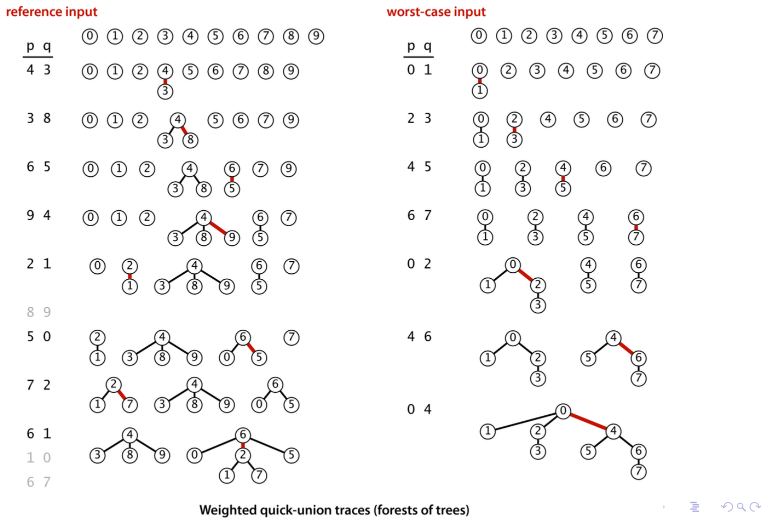
```
uf
+-------------------------------------------------------+
| pai (private)                 count:  2 (private)|
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 6 | 1 | 1 | 4 | 4 | 6 | 6 | 1 | 4 | 4 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| sz (private)                                          |
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 1 | 3 | 1 | 1 | 4 | 1 | 3 | 1 | 1 | 1 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| Métodos: cont(), connected(), find(), union()         |
+-------------------------------------------------------+
```

WeightedQuickUnionUF

uf.union(6, 1);

```
uf
+-------------------------------------------------------+
| pai (private)                 count:  2 (private)|
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 6 | 6 | 1 | 4 | 4 | 6 | 6 | 1 | 4 | 4 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| sz (private)                                          |
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 1 | 3 | 1 | 1 | 5 | 1 | 6 | 1 | 1 | 1 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9         |
|                                                       |
| Métodos: cont(), connected(), find(), union()         |
+-------------------------------------------------------+
```

Simulação



Estrutura *disjoint-set forest*

Para verificar que o consumo de tempo de union() e find() é não superior a $\lg n$, basta demonstrar que

> *Na floresta de árvores disjuntas produzida durante uma sequência de operações union(), toda árvore com altura h tem pelo menos 2^h nós.*

A demonstração é por indução no número de operações union() realizadas.

Estrutura *disjoint-set forest*

Inicialmente nenhuma operação union() foi realizada e toda árvore tem altura zero e possui um nó. Logo vale a afirmação.

Sejam p e q sítios e considere a operação union(p, q).

Se p e q estão em uma mesma árvore não há o que demonstrar. Portanto, podemos supor que a árvore T_p que contém p e árvore T_q que contém q são distintas.

Estrutura *disjoint-set forest*

Sejam

- h_p e n_p a altura e número de nós de T_p e
- h_q e n_q a altura e número de nós de T_q.

Pela hipóse de indução $n_p \geq 2^{h_p}$ e $n_q \geq 2^{h_q}$.

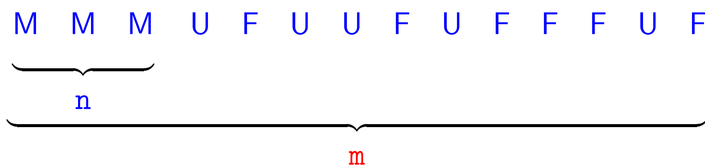
Seja T a árvore de altura h resultante da operação union(p,q). Se $h \leq \max\{h_p, h_q\}$, não há o que demonstrar. Assim, podemos supor que, digamos, $n_p \geq n_q$ e $h = h_q + 1$. Logo,

$$n = n_p + n_q \geq n_q + n_q \geq 2^{h_q} + 2^{h_q} = 2^{h_q+1} = 2^h.$$

O que encerra este rascunho de demonstração.

Consumo de tempo

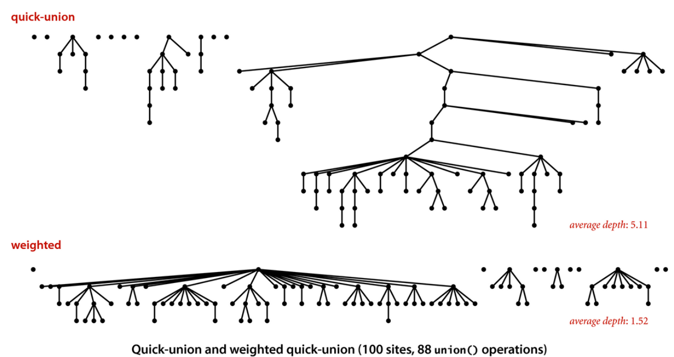
UF(n)	$\Theta(n)$
find(p)	$O(\lg n)$
union(p, q)	$O(\lg n)$



Custo total da sequência:

$$\Theta(n) + m\,O(\lg n) + n\,O(\lg n) = O(m \lg n)$$

Ilustração



Quick-union and weighted quick-union (100 sites, 88 union() operations)

Experimentos

```
% java Driver < tinyUF.txt
2 components
0.0003seg

% java Driver < mediumUF.txt
3 components
0.027seg

% java Driver < largeUF.txt
6 components
4.079seg
```

Encurtamento de caminhos

Acrescentando uma linha a `find()` encurtamos o comprimento do caminho à metade.

```
public int find(int p) {
    while (p != pai[p]) {
        // encurta caminho à metade
        pai[p] = pai[pai[p]];
        p = pai[p];
    }
    return p;
}
```

Mais experimentos

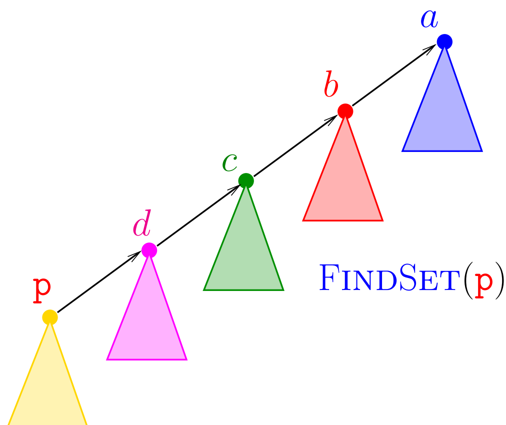
```
% java Driver < tinyUF.txt
2 components
0.0003seg

% java Driver < mediumUF.txt
3 components
0.025seg

% java Driver < largeUF.txt
6 components
3.923seg
```

Weight-Quick-union with path compression

1.5 Case Study: Union-Find

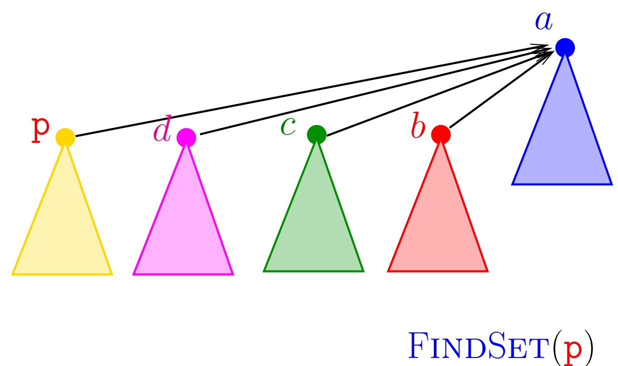
path compression

Ideia: encurtar os caminhos durante cada `find()`.



path compression

Ideia: encurtar os caminhos durante cada `find()`.



FINDSET(p)

path compression

FINDSET (p) ▷ com "path compression"
1 **se** p ≠ pai[p]
2 **então** pai[p] ← FINDSET (pai[p])
3 **devolva** pai[p]

path compression

```
public int find(int p) {
    if (p != pai[p])
        pai[p] = find[pai[p]];
    return pai[p];
}
```

PathCompressionUF

PathCompressionUF uf = new PathCompressionUF(10);

```
uf
+----------------------------------------------------+
| pai (private)                   count: 0 (private) |
|  |                                                 |
|  |     +---+---+---+---+---+---+---+---+---+---+   |
|  +---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |   |
|        +---+---+---+---+---+---+---+---+---+---+   |
|          0   1   2   3   4   5   6   7   8   9     |
|                                                    |
| sz (private)                                       |
|  |                                                 |
|  |     +---+---+---+---+---+---+---+---+---+---+   |
|  +---> | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |   |
|        +---+---+---+---+---+---+---+---+---+---+   |
|          0   1   2   3   4   5   6   7   8   9     |
|                                                    |
| Métodos: cont(), connected(), find(), union()      |
+----------------------------------------------------+
```

PathCompressionUF

uf.find(3) retorna 3

uf.find(0) retorna 0

```
uf
+----------------------------------------------------+
| pai (private)                  count: 10 (private) |
|  |                                                 |
|  |     +---+---+---+---+---+---+---+---+---+---+   |
|  +---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |   |
|        +---+---+---+---+---+---+---+---+---+---+   |
|          0   1   2   3   4   5   6   7   8   9     |
|                                                    |
| sz (private)                                       |
|  |                                                 |
|  |     +---+---+---+---+---+---+---+---+---+---+   |
|  +---> | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |   |
|        +---+---+---+---+---+---+---+---+---+---+   |
|          0   1   2   3   4   5   6   7   8   9     |
|                                                    |
| Métodos: cont(), connected(), find(), union()      |
+----------------------------------------------------+
```

PathCompressionUF

uf.union(4, 3);

```
uf
+----------------------------------------------------+
| pai (private)                   count:  9 (private)|
|  |                                                 |
|  |     +---+---+---+---+---+---+---+---+---+---+   |
|  +---> | 0 | 1 | 2 | 4 | 4 | 5 | 6 | 7 | 8 | 9 |   |
|        +---+---+---+---+---+---+---+---+---+---+   |
|          0   1   2   3   4   5   6   7   8   9     |
|                                                    |
| sz (private)                                       |
|  |                                                 |
|  |     +---+---+---+---+---+---+---+---+---+---+   |
|  +---> | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |   |
|        +---+---+---+---+---+---+---+---+---+---+   |
|          0   1   2   3   4   5   6   7   8   9     |
|                                                    |
| Métodos: cont(), connected(), find(), union()      |
+----------------------------------------------------+
```

PathCompressionUF

uf.union(3, 8);

```
uf
+----------------------------------------------------+
| pai (private)                   count:  8 (private)|
|  |                                                 |
|  |     +---+---+---+---+---+---+---+---+---+---+   |
|  +---> | 0 | 1 | 2 | 4 | 4 | 5 | 6 | 7 | 4 | 9 |   |
|        +---+---+---+---+---+---+---+---+---+---+   |
|          0   1   2   3   4   5   6   7   8   9     |
|                                                    |
| sz (private)                                       |
|  |                                                 |
|  |     +---+---+---+---+---+---+---+---+---+---+   |
|  +---> | 1 | 1 | 1 | 1 | 3 | 1 | 1 | 1 | 1 | 1 |   |
|        +---+---+---+---+---+---+---+---+---+---+   |
|          0   1   2   3   4   5   6   7   8   9     |
|                                                    |
| Métodos: cont(), connected(), find(), union()      |
+----------------------------------------------------+
```

PathCompressionUF

uf.union(6, 5);

```
uf
+-------------------------------------------------+
| pai (private)                count:  7 (private) |
|  |                                              |
|  |    +---+---+---+---+---+---+---+---+---+---+  |
|  +--> | 0 | 1 | 2 | 4 | 4 | 6 | 6 | 7 | 4 | 9 |  |
|       +---+---+---+---+---+---+---+---+---+---+  |
|         0   1   2   3   4   5   6   7   8   9   |
|                                                 |
| sz (private)                                    |
|  |                                              |
|  |    +---+---+---+---+---+---+---+---+---+---+  |
|  +--> | 1 | 1 | 1 | 1 | 3 | 1 | 2 | 1 | 1 | 1 |  |
|       +---+---+---+---+---+---+---+---+---+---+  |
|         0   1   2   3   4   5   6   7   8   9   |
|                                                 |
| Métodos: cont(), connected(), find(), union()   |
+-------------------------------------------------+
```

PathCompressionUF

uf.union(9, 4);

```
uf
+-------------------------------------------------+
| pai (private)                count:  6 (private)|
|  |                                              |
|  |    +---+---+---+---+---+---+---+---+---+---+  |
|  +--> | 0 | 1 | 2 | 4 | 4 | 6 | 6 | 7 | 4 | 4 |  |
|       +---+---+---+---+---+---+---+---+---+---+  |
|         0   1   2   3   4   5   6   7   8   9   |
|                                                 |
| sz (private)                                    |
|  |                                              |
|  |    +---+---+---+---+---+---+---+---+---+---+  |
|  +--> | 1 | 1 | 1 | 1 | 4 | 1 | 2 | 1 | 1 | 1 |  |
|       +---+---+---+---+---+---+---+---+---+---+  |
|         0   1   2   3   4   5   6   7   8   9   |
|                                                 |
| Métodos: cont(), connected(), find(), union()   |
+-------------------------------------------------+
```

PathCompressionUF

uf.union(2, 1);

```
uf
+-------------------------------------------------+
| pai (private)                count:  5 (private)|
|  |                                              |
|  |    +---+---+---+---+---+---+---+---+---+---+  |
|  +--> | 0 | 1 | 1 | 4 | 4 | 6 | 6 | 7 | 4 | 4 |  |
|       +---+---+---+---+---+---+---+---+---+---+  |
|         0   1   2   3   4   5   6   7   8   9   |
|                                                 |
| sz (private)                                    |
|  |                                              |
|  |    +---+---+---+---+---+---+---+---+---+---+  |
|  +--> | 1 | 2 | 1 | 1 | 3 | 1 | 2 | 1 | 1 | 1 |  |
|       +---+---+---+---+---+---+---+---+---+---+  |
|         0   1   2   3   4   5   6   7   8   9   |
|                                                 |
| Métodos: cont(), connected(), find(), union()   |
+-------------------------------------------------+
```

PathCompressionUF

uf.union(8, 9);

```
uf
+-------------------------------------------------+
| pai (private)                count:  4 (private)|
|  |                                              |
|  |    +---+---+---+---+---+---+---+---+---+---+  |
|  +--> | 0 | 1 | 1 | 4 | 4 | 6 | 6 | 7 | 4 | 4 |  |
|       +---+---+---+---+---+---+---+---+---+---+  |
|         0   1   2   3   4   5   6   7   8   9   |
|                                                 |
| sz (private)                                    |
|  |                                              |
|  |    +---+---+---+---+---+---+---+---+---+---+  |
|  +--> | 1 | 2 | 1 | 1 | 4 | 1 | 2 | 1 | 1 | 1 |  |
|       +---+---+---+---+---+---+---+---+---+---+  |
|         0   1   2   3   4   5   6   7   8   9   |
|                                                 |
| Métodos: cont(), connected(), find(), union()   |
+-------------------------------------------------+
```

PathCompressionUF

uf.union(5, 0);

```
uf
+-------------------------------------------------+
| pai (private)                count:  3 (private)|
|  |                                              |
|  |    +---+---+---+---+---+---+---+---+---+---+  |
|  +--> | 6 | 1 | 1 | 4 | 4 | 6 | 6 | 7 | 4 | 4 |  |
|       +---+---+---+---+---+---+---+---+---+---+  |
|         0   1   2   3   4   5   6   7   8   9   |
|                                                 |
| sz (private)                                    |
|  |                                              |
|  |    +---+---+---+---+---+---+---+---+---+---+  |
|  +--> | 1 | 2 | 1 | 1 | 4 | 1 | 3 | 1 | 1 | 1 |  |
|       +---+---+---+---+---+---+---+---+---+---+  |
|         0   1   2   3   4   5   6   7   8   9   |
|                                                 |
| Métodos: cont(), connected(), find(), union()   |
+-------------------------------------------------+
```

PathCompressionUF

uf.union(7, 2);

```
uf
+-------------------------------------------------+
| pai (private)                count:  2 (private)|
|  |                                              |
|  |    +---+---+---+---+---+---+---+---+---+---+  |
|  +--> | 6 | 1 | 1 | 4 | 4 | 6 | 6 | 1 | 4 | 4 |  |
|       +---+---+---+---+---+---+---+---+---+---+  |
|         0   1   2   3   4   5   6   7   8   9   |
|                                                 |
| sz (private)                                    |
|  |                                              |
|  |    +---+---+---+---+---+---+---+---+---+---+  |
|  +--> | 1 | 3 | 1 | 1 | 4 | 1 | 3 | 1 | 1 | 1 |  |
|       +---+---+---+---+---+---+---+---+---+---+  |
|         0   1   2   3   4   5   6   7   8   9   |
|                                                 |
| Métodos: cont(), connected(), find(), union()   |
+-------------------------------------------------+
```

PathCompressionUF

uf.union(6, 1);

```
uf
+------------------------------------------------------+
| pai (private)                      count:  2 (private)|
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 6 | 6 | 1 | 4 | 4 | 6 | 6 | 1 | 4 | 4 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9        |
|                                                       |
| sz (private)                                          |
|  |                                                    |
|  |    +---+---+---+---+---+---+---+---+---+---+       |
|  +--> | 1 | 3 | 1 | 1 | 5 | 1 | 6 | 1 | 1 | 1 |       |
|       +---+---+---+---+---+---+---+---+---+---+       |
|         0   1   2   3   4   5   6   7   8   9        |
|                                                       |
| Métodos: cont(), connected(), find(), union()         |
+------------------------------------------------------+
```

Função log-estrela

$\lg^* n$ é o menor k tal que

$$\underbrace{\lg \lg \ldots \lg}_{k} n \leq 1$$

n	$\lg^*$ n
1	0
2	1
3	2
4	2
5	3
⋮	⋮
15	3
16	3
⋮	⋮
65535	4
65536	4
⋮	⋮
$\underbrace{100000000000000000 \cdots 000000000000}_{80}$	5
⋮	⋮

Função 'torre'

$$t(i) := \begin{cases} 1 & \text{se } i = 0 \\ 2^{t(i-1)} & \text{se } i = 1, 2, 3, \ldots \end{cases}$$

i	$t(i)$
0	1
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^{2^2} = 16$
4	$2^{2^{2^2}} = 2^{16} = 65536$
5	$2^{2^{2^{2^2}}} > \underbrace{100000000000000000 \cdots 000000000000}_{80}$
⋮	⋮

Consumo de tempo

UF(n)	$\Theta(n)$	
find(p)	$O(\lg^* n)$	**amortizado!**
union(p, q)	$O(\lg^* n)$	**amortizado!**

M M M U F U U F U F F F U F

$\underbrace{\text{M M M}}_{n}$; whole sequence $\underbrace{}_{m}$

Custo total da sequência:

$$\Theta(n) + m\,O(\lg^* n) + n\,O(\lg^* n) = O(m \lg^* n)$$

Conclusões

Se conjuntos disjuntos são representados através de disjoint-set forest com *union by rank* e *path compression*, então uma sequência de UF(n) e m operações union() e find(), sendo que n, consome tempo $O(m \lg^* n)$.

Experimentos

Acrescentando a linha

```
// diminui a altura pela metade
pai[p] = pai[pai[p]]
```

Experimentos

```
% java Driver < tinyUF.txt
```

2 components
0.0003seg

```
% java Driver < mediumUF.txt
```

3 components
0.025seg

```
% java Driver < largeUF.txt
```

6 components
3.923seg

Parece que na prática weighted quick-union e weighted quick-union com path-compression não são muito diferentes.