

Aula 14 - Estruturas de Dados para Conjuntos Disjuntos

- Algoritmo Union-Find
 - Propriedades Elementares
 - Análise de Complexidade
- Heurística de Compressão de Caminho

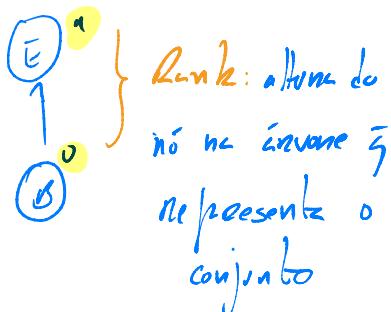
Estruturas de Dados p/ Conjuntos Disjuntos

- $\text{Make-Set}(x)$ - Cria o conjunto $\{x\}$
- $\text{Union}(x, y)$ - $\text{SetOf}(x) \cup \text{SetOf}(y)$
 - ↳ Conjunto que contém y
 - ↳ Conjunto que contém x
- $\text{FindSet}(x)$ - Retorna o ponteiro p_i o representante do conjunto que contém x

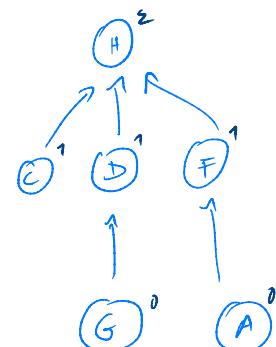
* Observação:
Cada conjunto tem um representante associado

Ideia: Os conjuntos são representados por árvores n-árias

Conjunto $\{B, E\}$



Conjunto $\{A, C, D, F, G, H\}$



* Cada nó tem:

- um rank
- um pai

- Make-Set(x) - Cria o conjunto $\{x\}$

Make-Set(x)

$$x.p = x$$

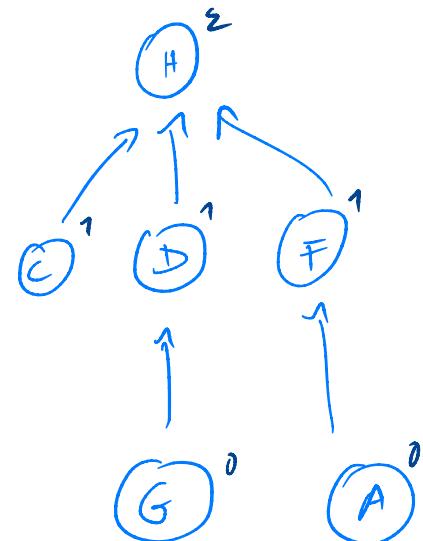
$$x.rank = 0$$

Make-Set(x) \Rightarrow 

- Find-Set(x) - Retorna o ponteiro p do representante do conjunto que contém x

Find-Set(x)

```
while ( $x \neq x.p$ )
     $x = x.p$ 
return  $x$ 
```



• Find-Set(A) ?

17

• Find-Set(C) ?

17

- $\text{Union}(x, y) = \text{SetOf}(x) \cup \text{SetOf}(y)$
 - ↳ Conjunto \bar{q} contém y
 - ↳ Conjunto \bar{q} contém x

$\text{Union}(x, y)$

let $\alpha_x = \text{FindSet}(x)$

let $\alpha_y = \text{FindSet}(y)$

if $\alpha_x == \alpha_y$ return

if $\alpha_x.\text{rank} > \alpha_y.\text{rank}$

$\alpha_y.p = \alpha_x$

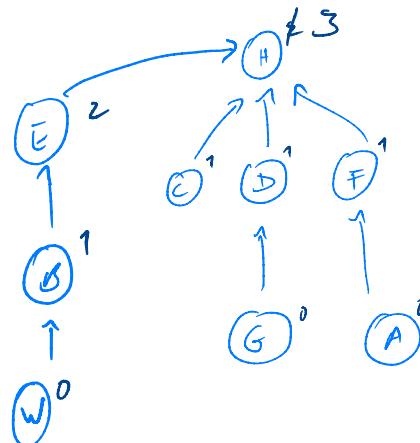
else $\alpha_y.\text{rank} > \alpha_x.\text{rank}$

$\alpha_x.p = \alpha_y$

else if $\alpha_x.\text{rank} == \alpha_y.\text{rank}$

$\alpha_x.\text{rank} += 1;$

$\alpha_y.p = \alpha_x$



- $\text{Union}(A, B) : R_A : H , R_B : E$
 $H.\text{rank} = 2 > E.\text{rank} = 1$

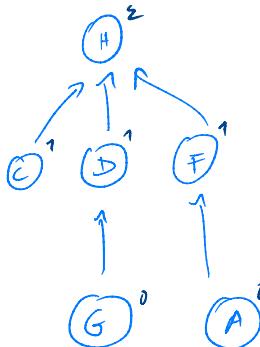
- $\text{Union}(x, y) = \text{Set of } (x) \cup \text{Set of } (y)$
 - ↳ Conjunto \bar{q} contém y
 - ↳ Conjunto \bar{q} contém x

$\text{Union}(x, y)$

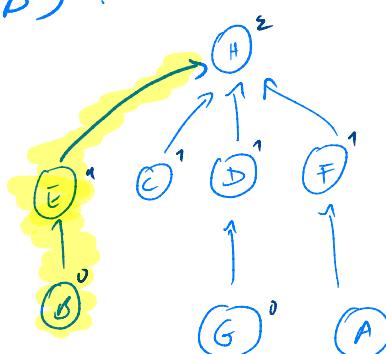
```

let  $\alpha_x = \text{FindSet}(x)$ 
let  $\alpha_y = \text{FindSet}(y)$ 
if ( $\alpha_x == \alpha_y$ ) return
if  $\alpha_x.\text{rank} > \alpha_y.\text{rank}$ 
     $\alpha_y.\text{p} = \alpha_x$ 
else  $\alpha_y.\text{rank} > \alpha_x.\text{rank}$ 
     $\alpha_x.\text{p} = \alpha_y$ 
else if  $\alpha_x.\text{rank} == \alpha_y.\text{rank}$ 
     $\alpha_x.\text{rank} += 1$ ;
     $\alpha_y.\text{p} = \alpha_x$ 

```



• $\text{Union}(A, B)$:



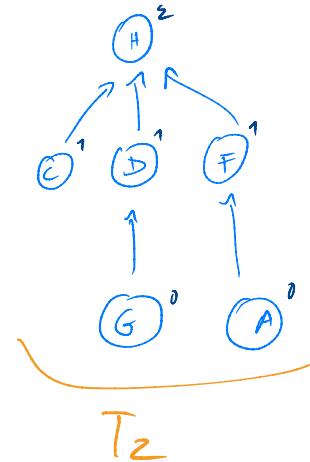
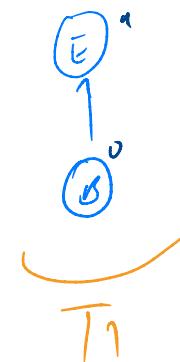
Com pressão de Caminho

Find Set (x)

if ($x \neq x.p$)

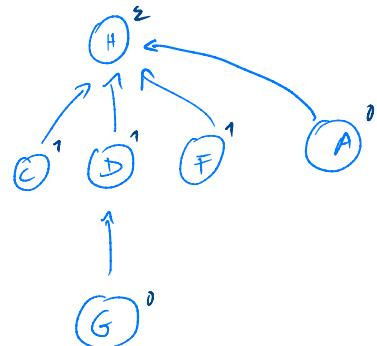
$x.p = \text{FindSet}(x.p)$

return $x.p$

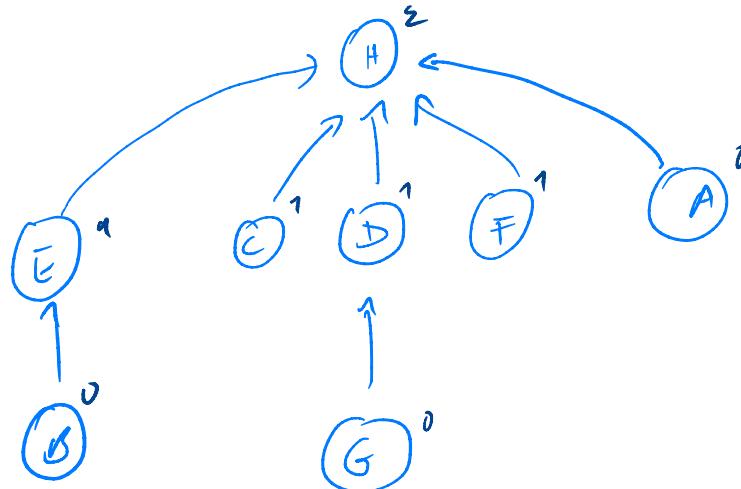


• Union (A, B) :

$R_A : 17$



$R_B : 15$



Objetivo: Provar que a complexidade de $\text{Find}(x)$ e $\text{Union}(x, y)$ é $O(\log(n))$ no pior caso.

Com n igual ao nº de elementos considerados

Propriedade 1 [Nós raiz]

Um nó que deixa de ser raiz **nunca** mais pode ser volta a ser-lo.

Propriedade 2 [Variação do Rank - 1]

O rank de qualquer nó só pode crescer com o tempo.

Propriedade 3 [Variação do Rank - 2]

Só os ranks de **nós raiz** é que podem aumentar.

↳ Quando um nó deixa de ser raiz o seu rank **nunca** mais vai ser alterado.

Propriedade 4 [Variação do Rank - 3]

Os ranks **crescem estatimamente** ao longo de cada caminho a ligar um nó folha a um nó raiz.

Lema das Ranks

O n° de nós com rank R é: $n/2^R$.

↳ **Conclusão:** Altura máxima de um nó:

$$R = \log n \\ \text{---} \\ 2$$

$$\frac{n}{2^{\log n}} = \frac{n}{n} = 1$$

$$\left\lceil \frac{\log n}{2} \right\rceil$$

Lema das Ranks

O n° de nós com rank r é: n/z^r .

→ cujos representantes têm o mesmo rank

Lema Auxiliar 1: Duas árvores com o mesmo rank são disjuntas

Lema Auxiliar 2: Um nó com rank r é raiz de uma árvore com pelo menos z^r elementos

Lema das Ranks

O n° de nós com rank r é: n/z^r .

Lema Auxiliar 1: Duas árvores com o mesmo rank são disjuntas

Lema Auxiliar 2: Um nó com rank r é raiz de uma árvore com pelo menos z^r elementos

- Suponhamos por contradição que x, y têm o mesmo rank e não
disjuntas
- Então existe um caminho φ :

$f \rightsquigarrow x \rightsquigarrow g \rightsquigarrow r$

→ os ranks não são estatamente
crescentes

Lema das Ranks

O n° de nós com rank r é: n/z^r .

Lema Auxiliar 1: Duas árvores com o mesmo rank são disjuntas

Lema Auxiliar 2: Um nó com rank r é raiz de uma árvore com pelo menos z^r elementos

Por indução no n° de union, n

$$\underline{n=0} \Rightarrow \forall v. \text{rank}(v) = 0, z^0 = 1 \xrightarrow{\substack{\text{nº de elementos} \\ \text{do conjunto } v}}$$

$$n = n' + 1$$

- Ajos n' union a propriedade é verificada
 $\forall x, y \in X. \text{Union}(x, y)$

Lema das Ranks

O n° de nós com rank R é: n/z^R .

Lema Auxiliar 1: Duas árvores com o mesmo rank são disjuntas

Lema Auxiliar 2: Um nó com rank R é raiz de uma árvore com pelo menos z^R elementos

Indução no n° de uniões n

$$\underline{n=0} \quad z^R = 1 \Rightarrow \checkmark$$

$$\underline{n=n+1} \quad \text{uniou } (x, y) \quad -$$

- 3 casos:

$$- R_x = R_y$$

$$- R_x > R_y \vee R_y > R_x$$

$$- R_x = R_y$$

$$R'_x = R_x + 1$$

$$n'_x = n_x + n_y$$

$$n_x \geq z^{R_x}$$

$$n_y \geq z^{R_y} = z^{R_x}$$

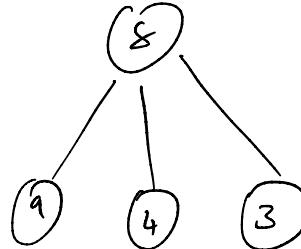
$$\begin{aligned} n'_x &= n_x + n_y \geq z^{R_x} + z^{R_x} \\ &= z^{R_x + 1} = z^{R'_x} \end{aligned}$$

Compressão de Caminho

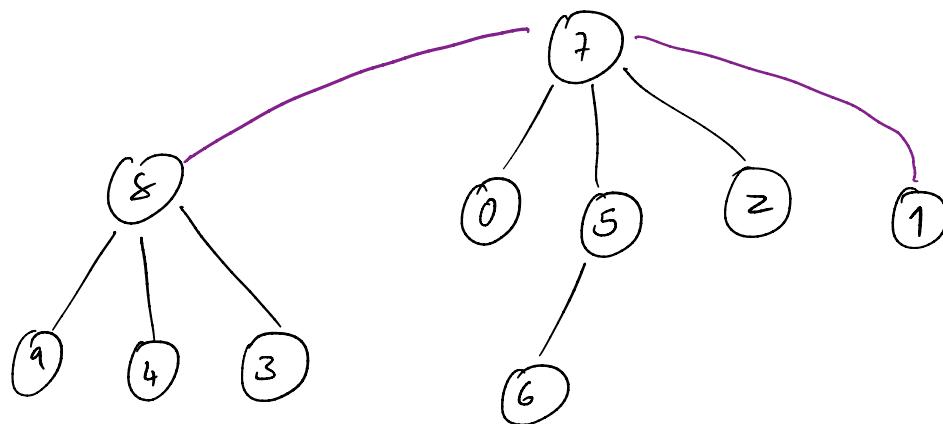
$\text{Find}(n)$

```
if ( $x \neq x.p$ )  
     $x.p = \text{Find}(x.p)$   
return  $x.p$ 
```

Exemplo:



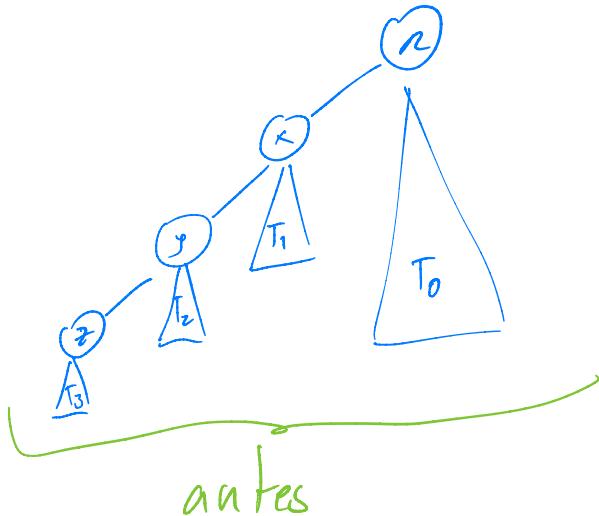
Union(1,3):



Compressão de Caminho

Find(n)

```
if ( $x \neq x.p$ )
     $x.p = \text{Find}(x.p)$ 
return  $x.p$ 
```



Complexidade:

- m operações de Union-Find com compressão de caminho numa estrutura com n nós têm complexidade:

- $O(m \log^* n)$ [Hopcroft-Ullman]
- $O(m \alpha(n))$ [Tauman]

Inverse Ackermann Function

