

# Melhores momentos

## AULA 22

# Conjuntos disjuntos dinâmicos

CLR 22    CLRS 21

# Conjuntos disjuntos

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

$$S_i \cap S_j = \emptyset$$

para todo  $i \neq j$ .

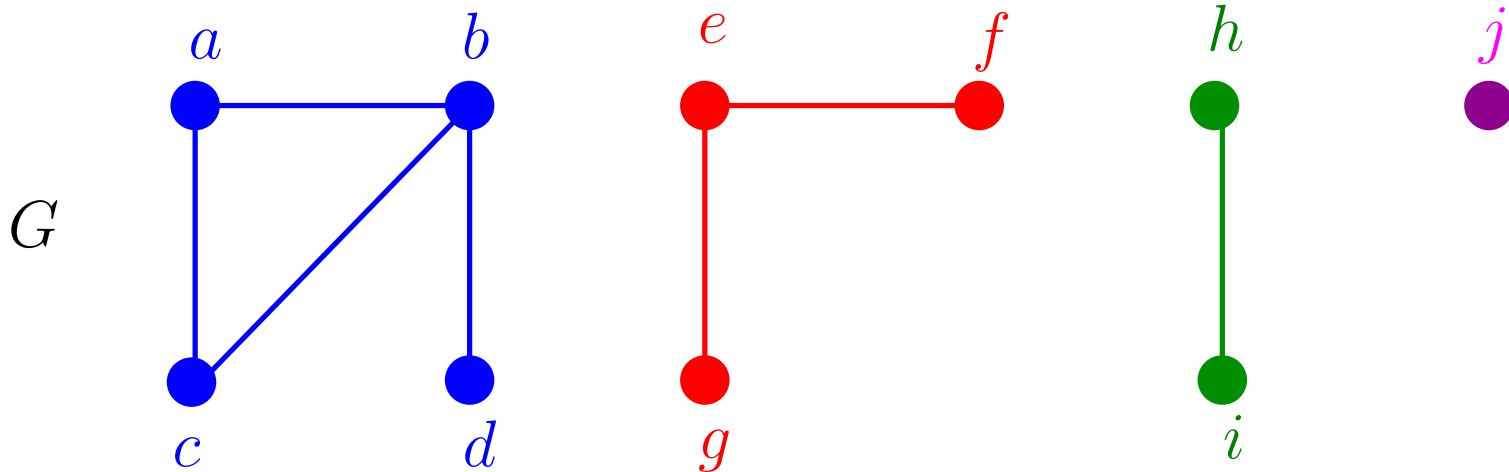
# Conjuntos disjuntos

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

$$S_i \cap S_j = \emptyset$$

para todo  $i \neq j$ .

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



componentes = conjuntos disjuntos de vértices

---

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

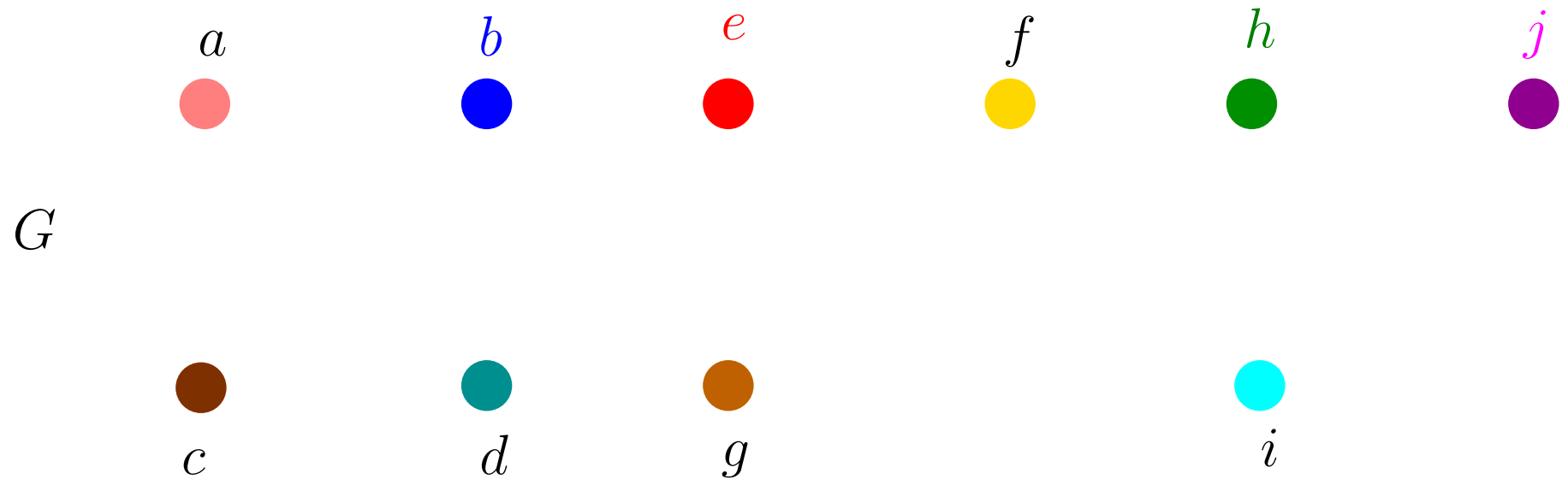
# Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

# Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: **grafo dinâmico**



aresta

componentes

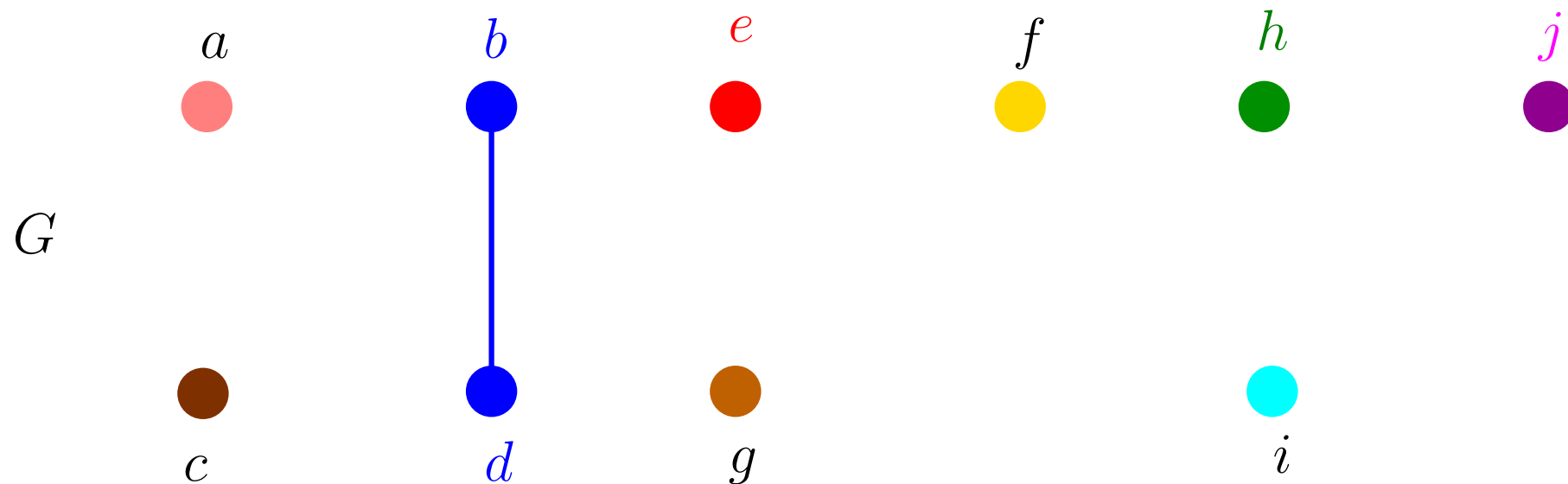
---

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

# Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: **grafo dinâmico**



aresta

componentes

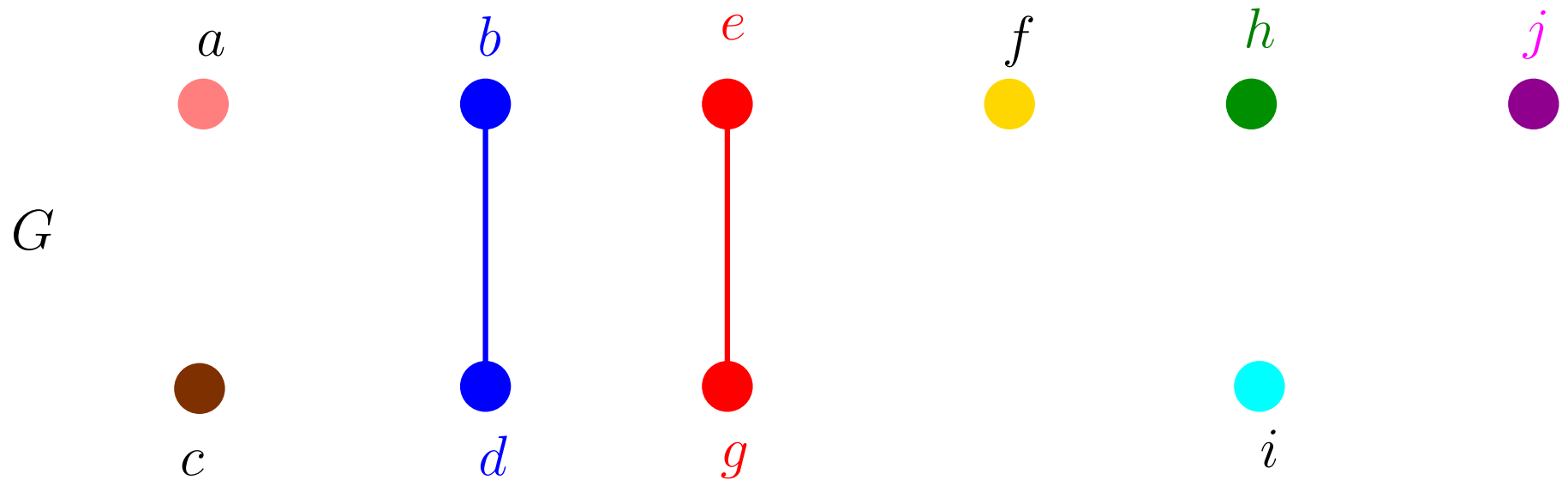
$(b, d)$

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

# Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: **grafo dinâmico**



aresta

componentes

$(e, g)$

$\{a\}$

$\{b, d\}$

$\{c\}$

$\{e, g\}$

$\{f\}$

$\{h\}$

$\{i\}$

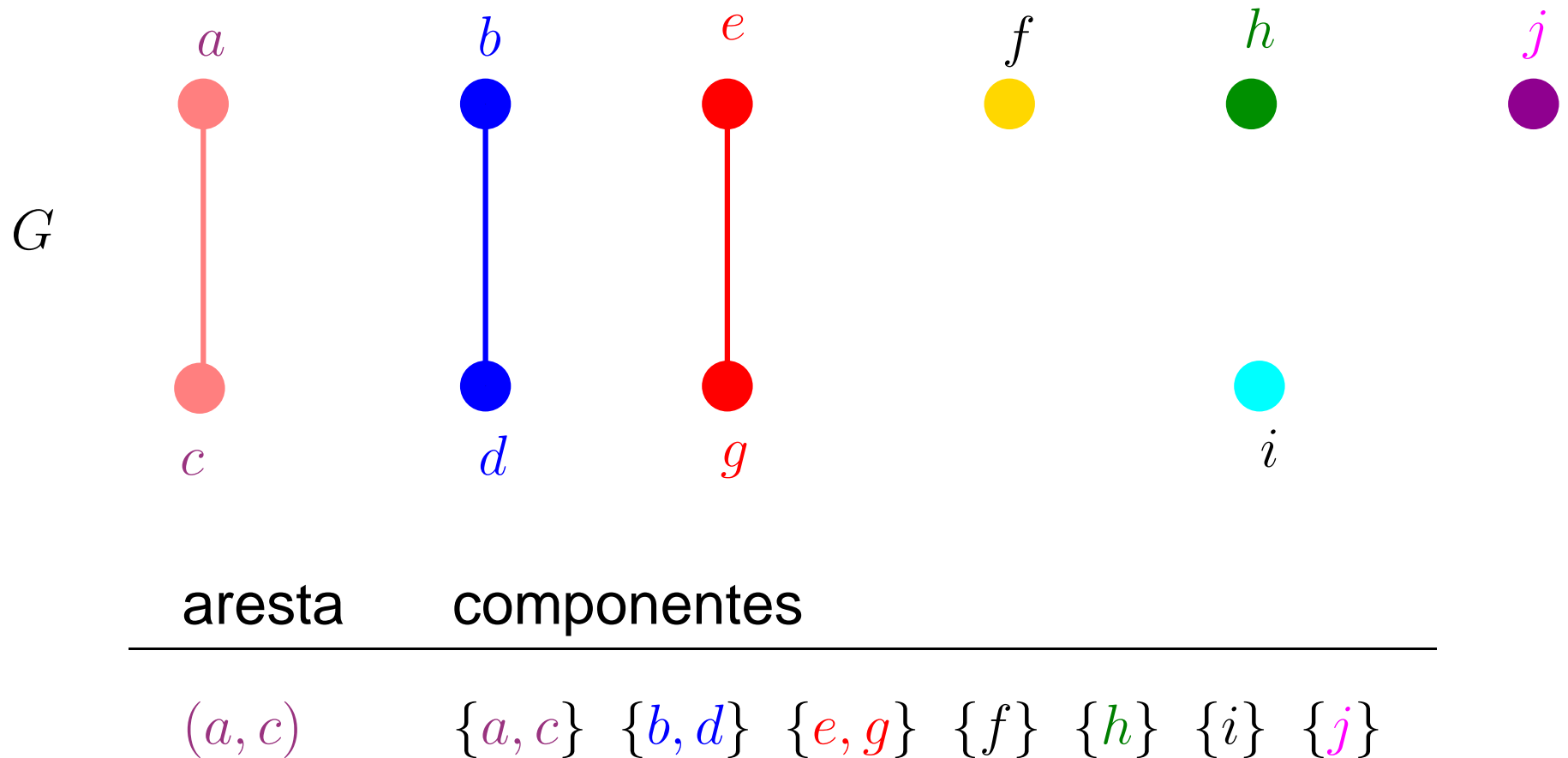
$\{j\}$



# Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

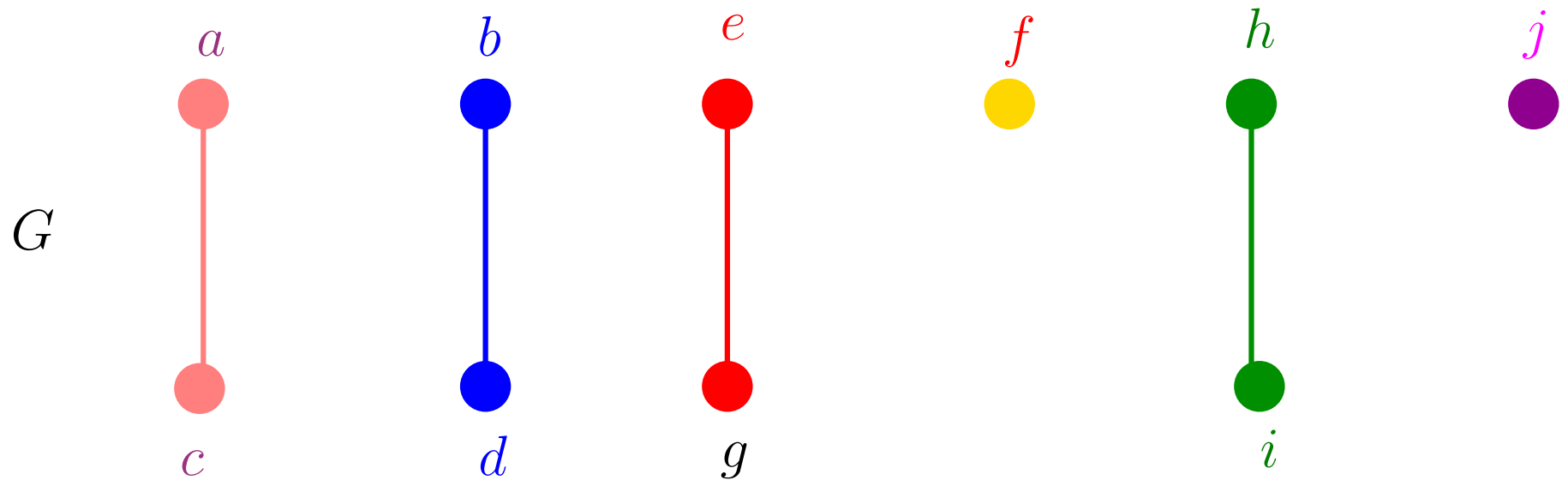
Exemplo: **grafo dinâmico**



# Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico



aresta

componentes

$(h, i)$

$\{a, c\}$

$\{b, d\}$

$\{e, g\}$

$\{f\}$

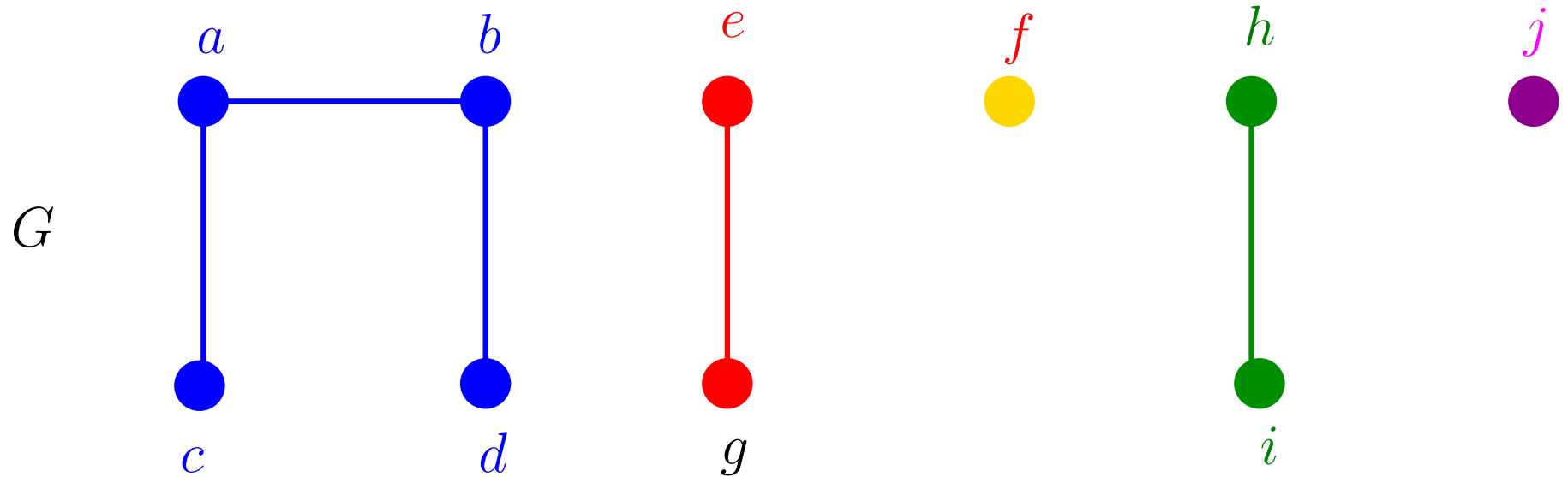
$\{h, i\}$

$\{j\}$

# Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico



aresta

componentes

$(a, b)$

$\{a, b, c, d\}$

$\{e, g\}$

$\{f\}$

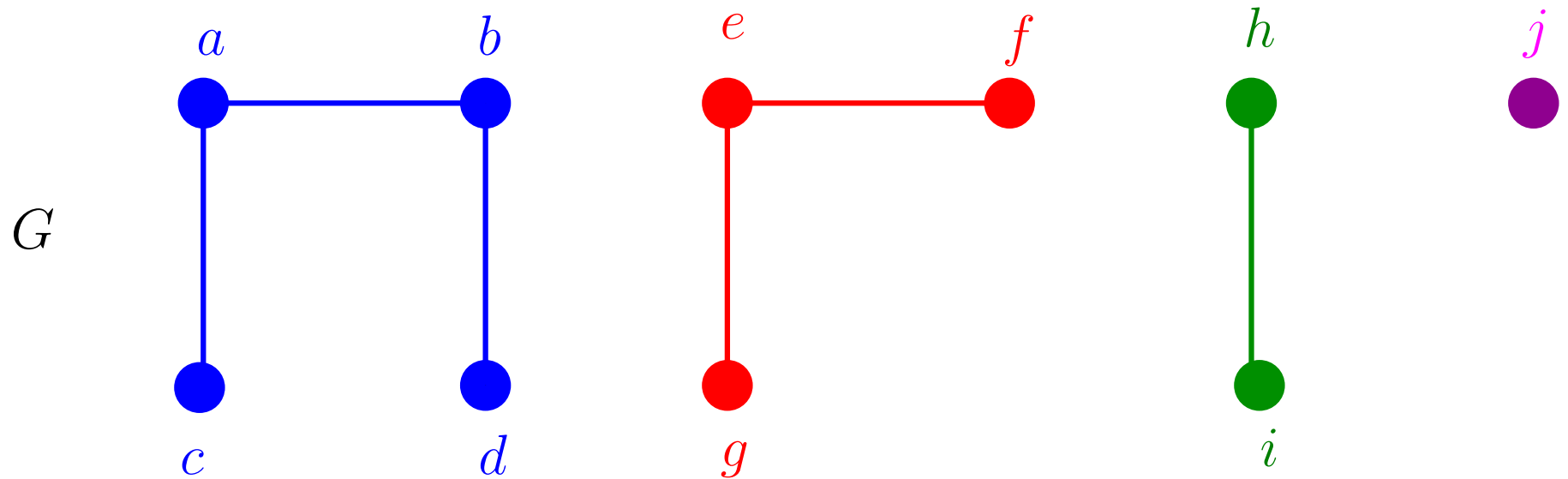
$\{h, i\}$

$\{j\}$

# Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico



aresta

componentes

$(e, f)$

$\{a, b, c, d\}$

$\{e, f, g\}$

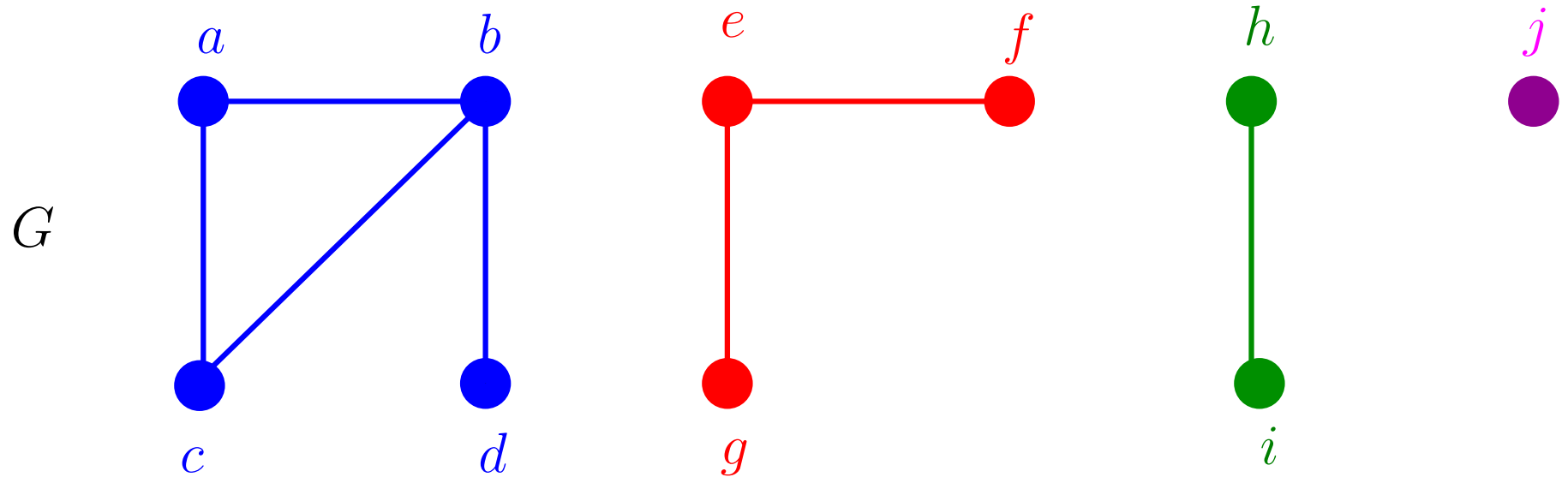
$\{h, i\}$

$\{j\}$

# Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: **grafo dinâmico**



aresta

componentes

$(b, c)$

$\{a, b, c, d\}$

$\{e, g\}$

$\{f\}$

$\{h, i\}$

$\{j\}$

# Operações básicas

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

Cada conjunto tem um **representante**.

**MAKESET** ( $x$ ):  $x$  é elemento novo

$$\mathcal{S} \leftarrow \mathcal{S} \cup \{x\}$$

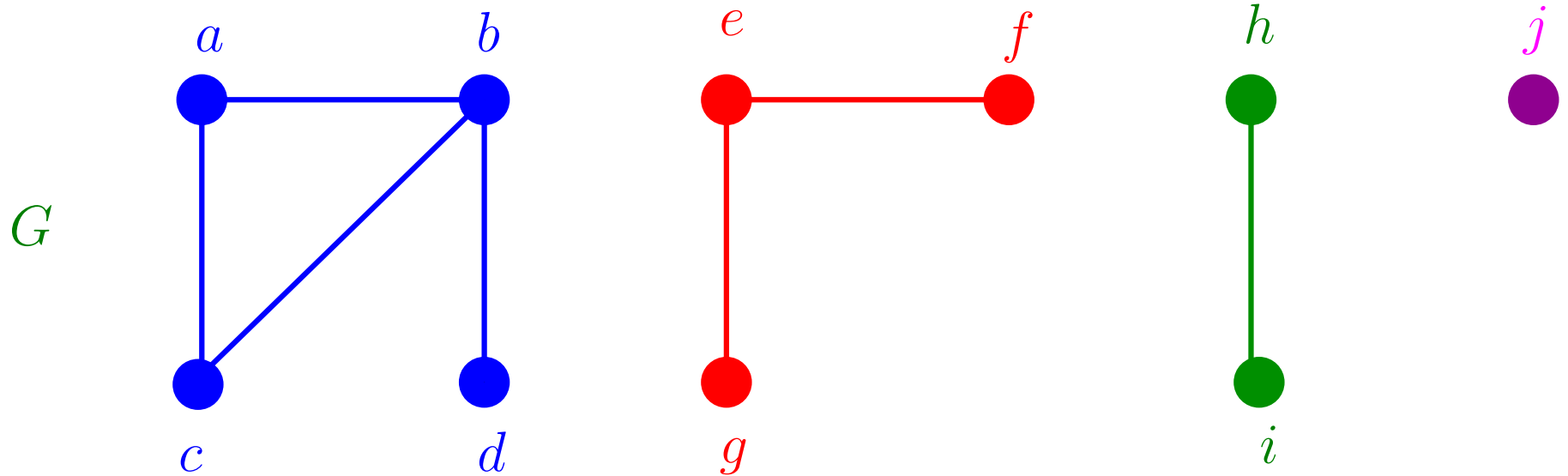
**UNION** ( $x, y$ ):  $x$  e  $y$  em conjuntos diferentes

$$\mathcal{S} \leftarrow \mathcal{S} - \{S_x, S_y\} \cup \{S_x \cup S_y\}$$

$x$  está em  $S_x$  e  $y$  está em  $S_y$

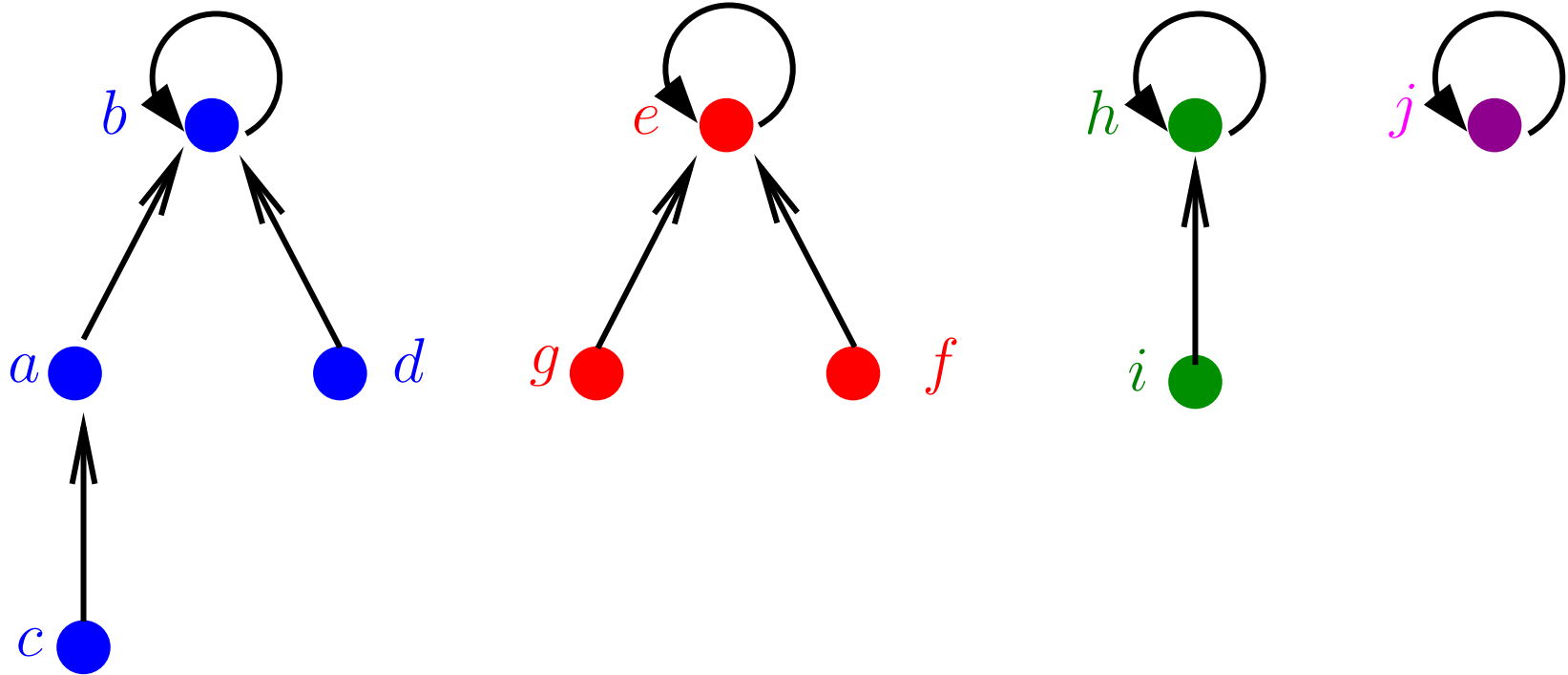
**FINDSET** ( $x$ ): devolve representante do conjunto que contém  $x$

# Estrutura *disjoint-set forest*



- cada conjunto tem uma *raiz*, que é o seu representante
- cada nó  $x$  tem um *pai*
- $\text{pai}[x] = x$  se e só se  $x$  é uma raiz

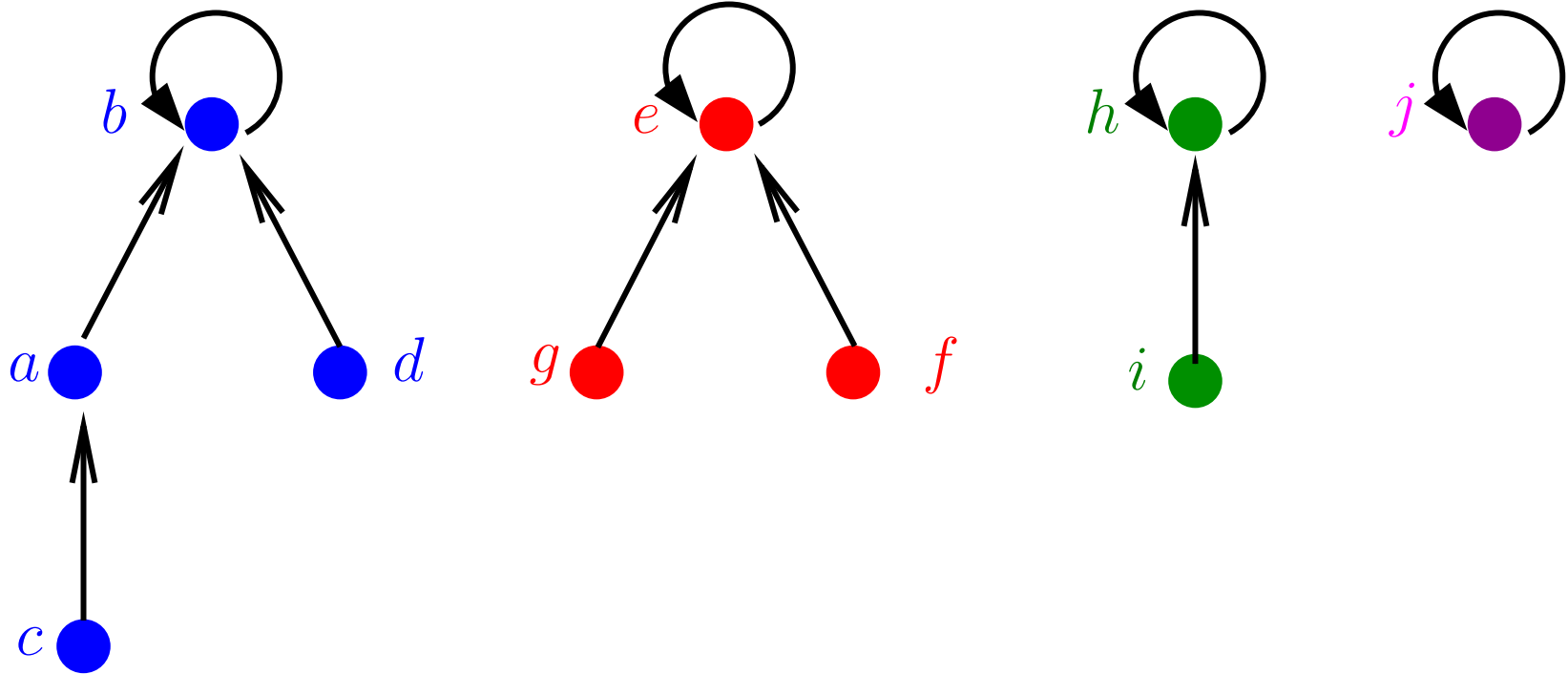
# Estrutura *disjoint-set forest*



- cada conjunto tem uma *raiz*
- cada nó  $x$  tem um *pai*
- $\text{pai}[x] = x$  se e só se  $x$  é uma raiz



# MakeSet<sub>0</sub> e FindSet<sub>0</sub>



MAKESET<sub>0</sub> ( $x$ )

1  $\text{pai}[x] \leftarrow x$

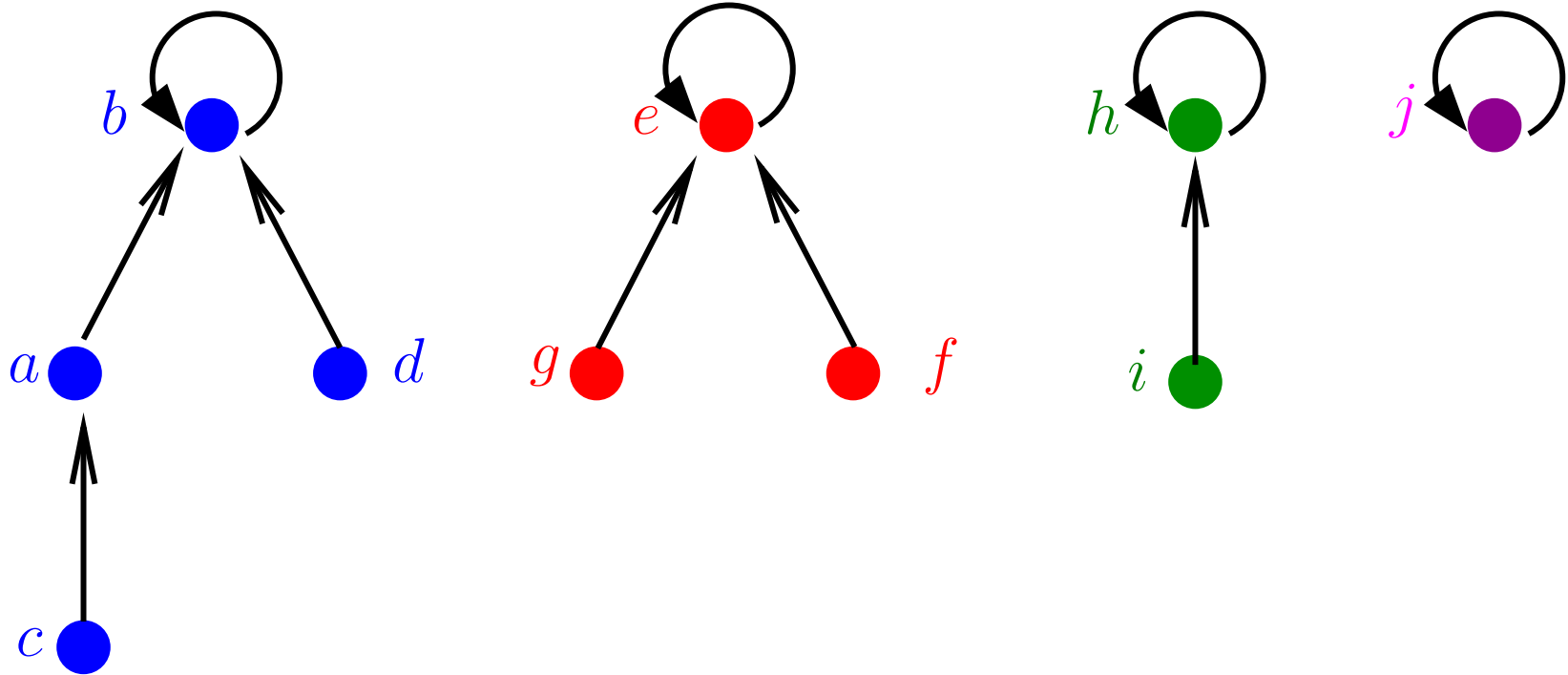
FINDSET<sub>0</sub> ( $x$ )

1 **enquanto**  $\text{pai}[x] \neq x$  **faça**

2      $x \leftarrow \text{pai}[x]$

3 **devolva**  $x$

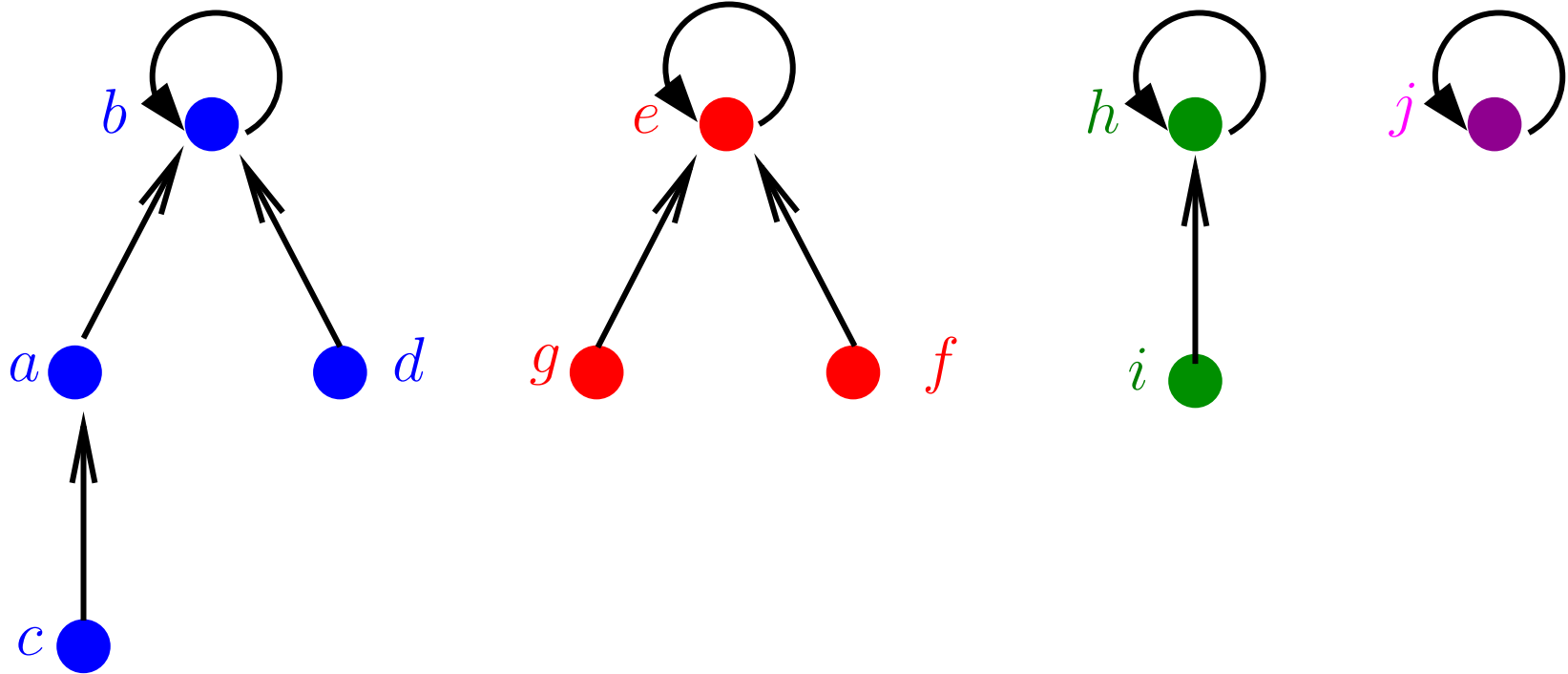
# FindSet<sub>1</sub>



**FINDSET**<sub>1</sub> (*x*)

- 1 **se**  $pai[x] = x$
- 2 **então devolva** *x*
- 3 **senão devolva** **FINDSET**<sub>1</sub> ( $pai[x]$ )

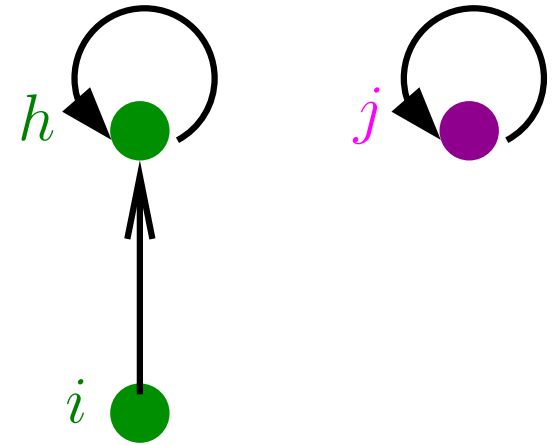
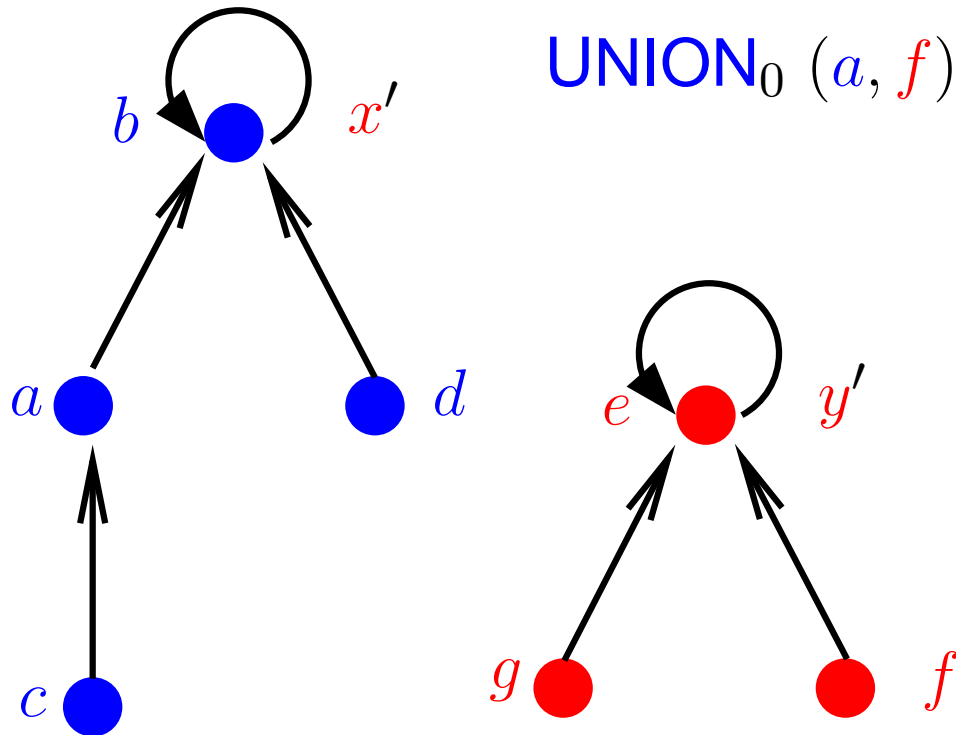
# Union<sub>0</sub>



**UNION<sub>0</sub>** ( $x, y$ )

- 1  $x' \leftarrow \text{FINDSET}_0(x)$
- 2  $y' \leftarrow \text{FINDSET}_0(y)$
- 3  $\text{pai}[y'] \leftarrow x'$

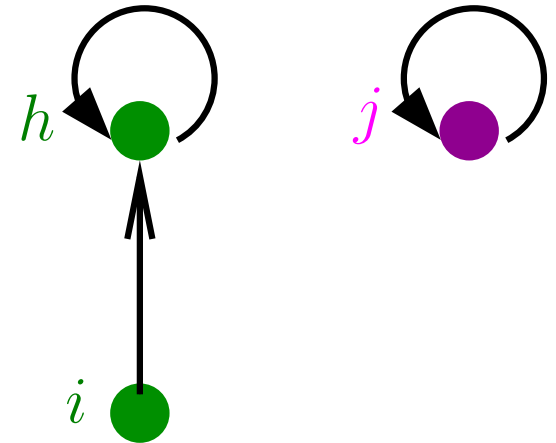
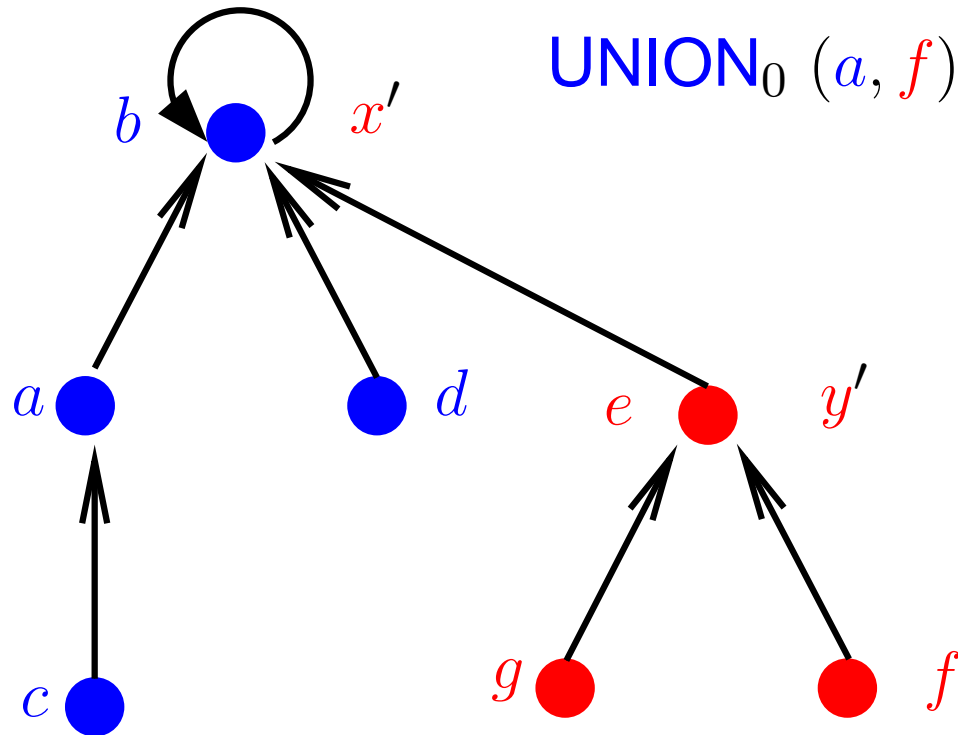
# Union<sub>0</sub>



UNION<sub>0</sub> ( $x, y$ )

- 1  $x' \leftarrow \text{FINDSET}_0(x)$
- 2  $y' \leftarrow \text{FINDSET}_0(y)$
- 3  $\text{pai}[y'] \leftarrow x'$

# Union<sub>0</sub>



UNION<sub>0</sub> ( $x, y$ )

- 1  $x' \leftarrow \text{FINDSET}_0(x)$
- 2  $y' \leftarrow \text{FINDSET}_0(y)$
- 3  $\text{pai}[y'] \leftarrow x'$

# MakeSet<sub>0</sub>, Union<sub>0</sub> e FindSet<sub>1</sub>

MAKESET<sub>0</sub> (*x*)

1 *pai*[*x*] ← *x*

UNION<sub>0</sub> (*x*, *y*)

1 *x'* ← FINDSET<sub>0</sub> (*x*)

2 *y'* ← FINDSET<sub>0</sub> (*y*)

3 *pai*[*y'*] ← *x'*

FINDSET<sub>1</sub> (*x*)

1 **se** *pai*[*x*] = *x*

2 **então devolva** *x*

3 **senão devolva** FINDSET<sub>1</sub> (*pai*[*x*])

# Consumo de tempo

MAKESET <sub>0</sub>	$\Theta(1)$
UNION <sub>0</sub>	$O(n)$
FINDSET <sub>0</sub>	$O(n)$

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



$n$

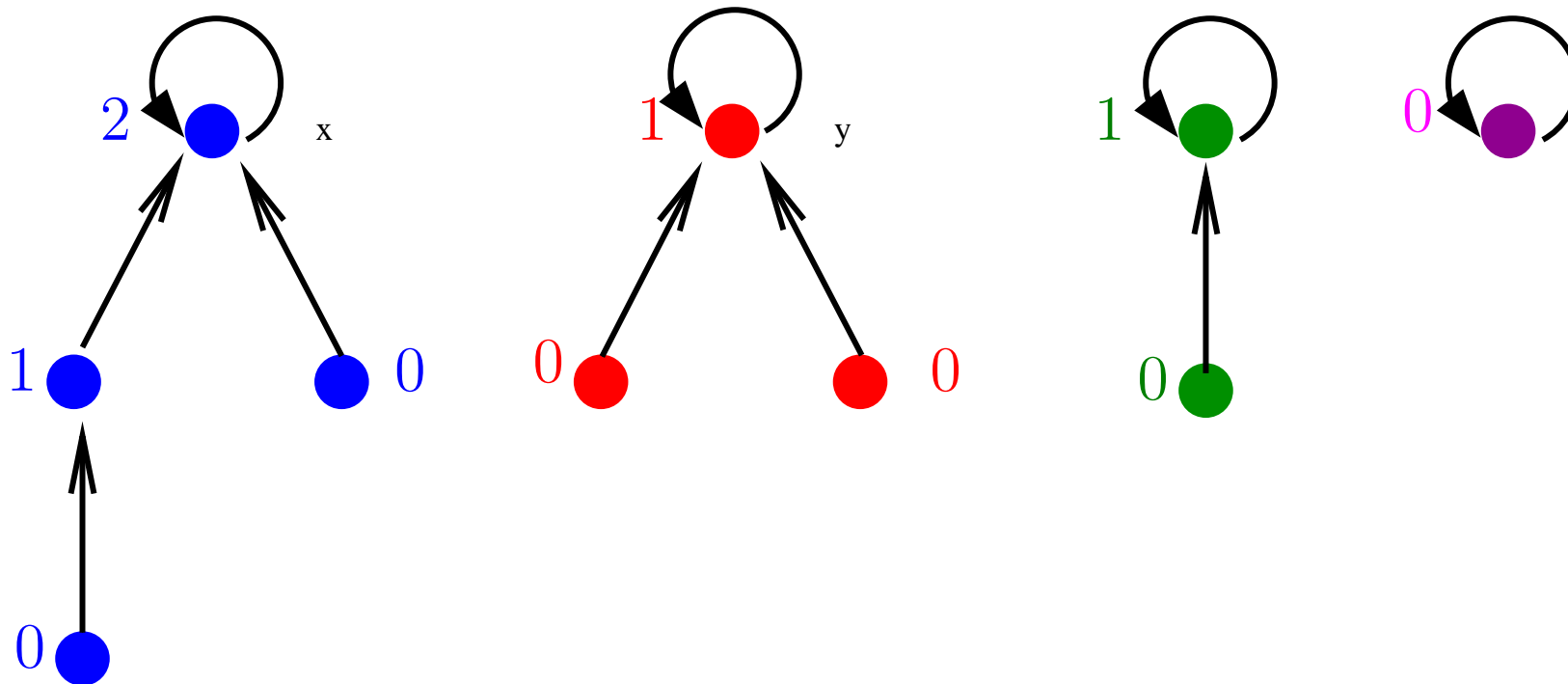


$m$

**Custo total da seqüência:**

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

# Melhoramento 1: *union by rank*



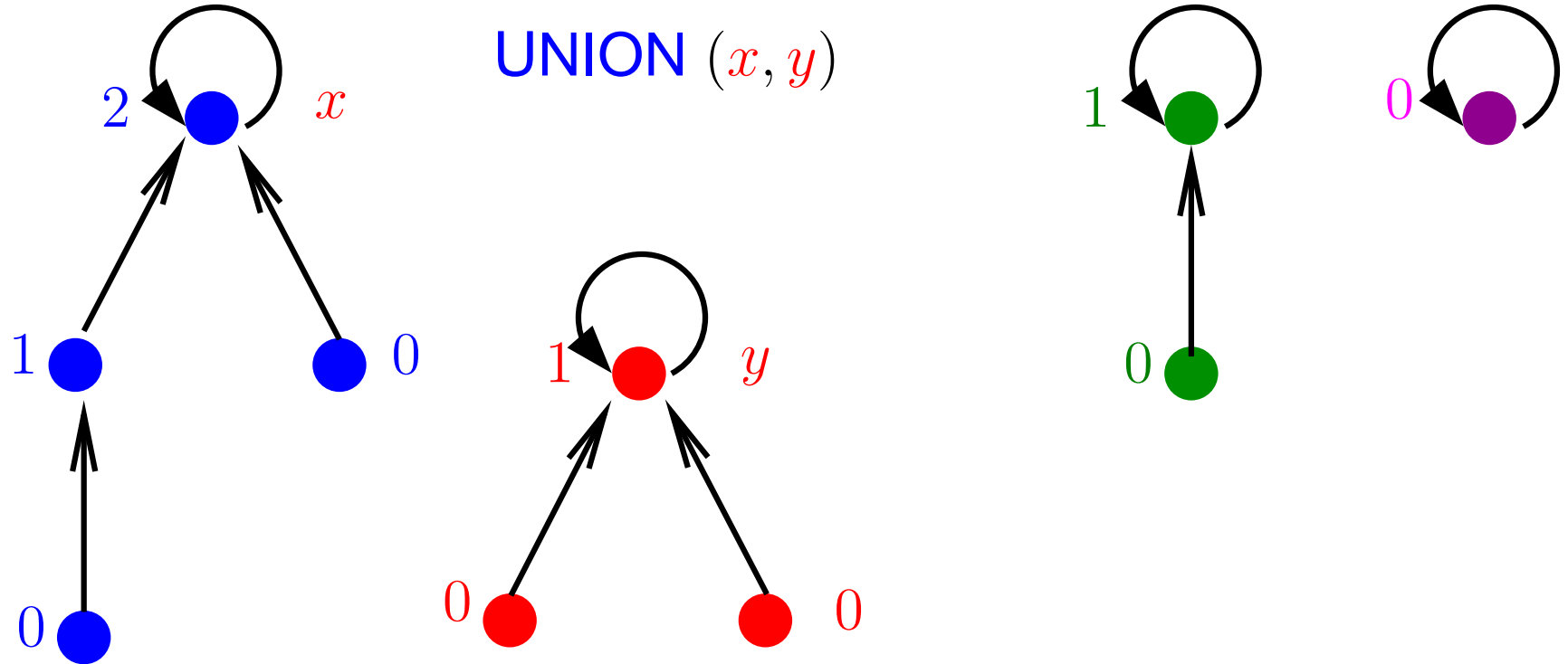
$rank[x]$  = posto do nó  $x$

**MAKESET** ( $x$ )

- 1  $pai[x] \leftarrow x$
- 2  $rank[x] \leftarrow 0$



# Melhoramento 1: *union by rank*



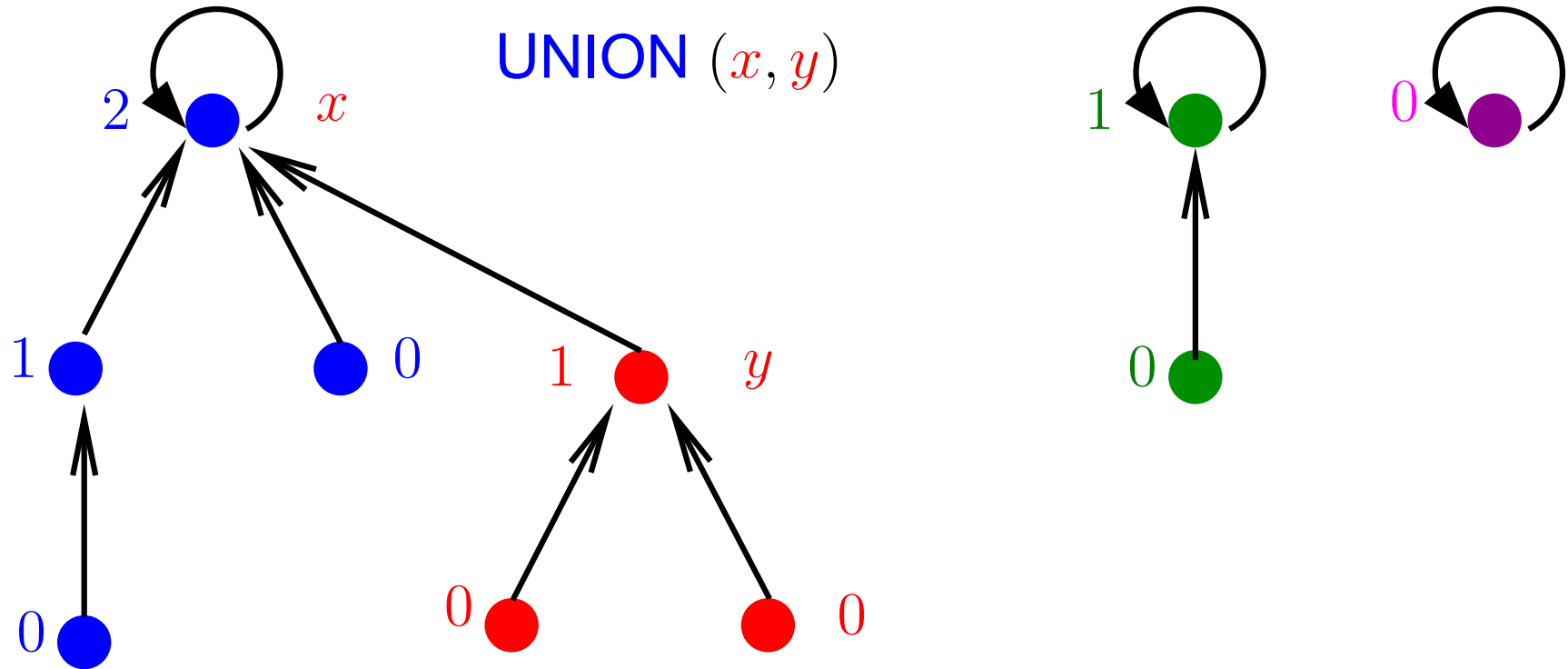
$rank[x]$  = posto do nó  $x$

MAKESET ( $x$ )

1  $pai[x] \leftarrow x$

2  $rank[x] \leftarrow 0$

# Melhoramento 1: *union by rank*



$rank[x]$  = posto do nó  $x$

MAKESET ( $x$ )

1  $pai[x] \leftarrow x$

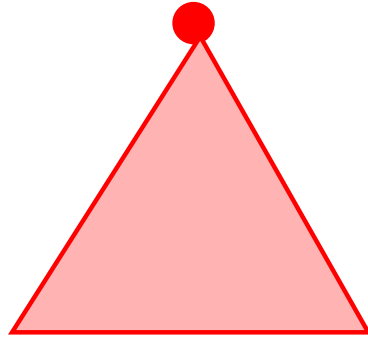
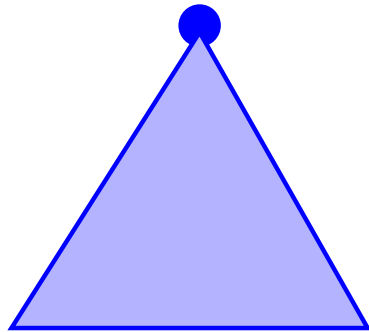
2  $rank[x] \leftarrow 0$

# Melhoramento 1: *union by rank*

$rank[x]$

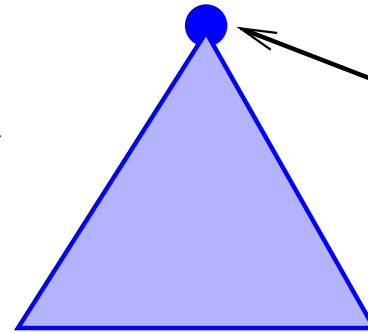
$>$

$rank[y]$

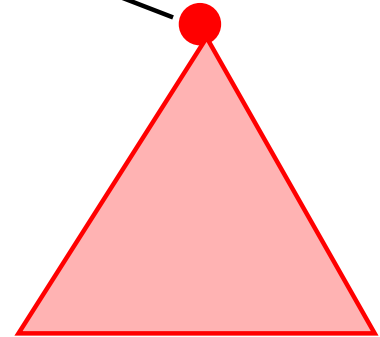


$\Rightarrow$

$rank[x]$



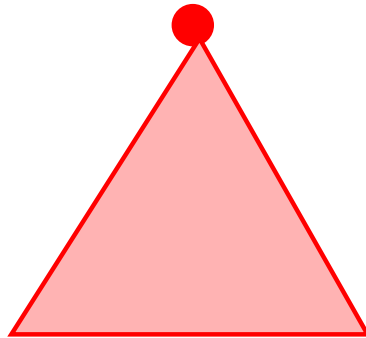
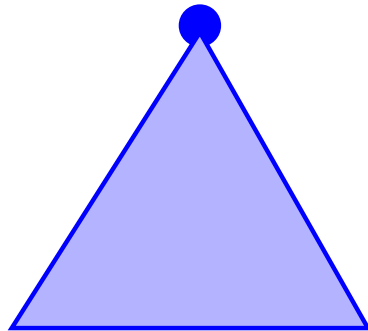
$rank[y]$



$rank[x]$

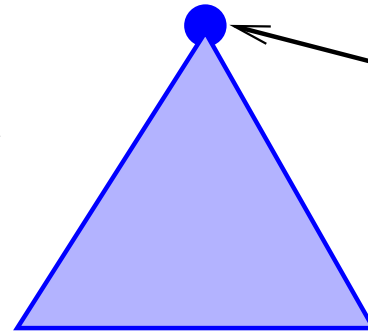
$=$

$rank[y]$

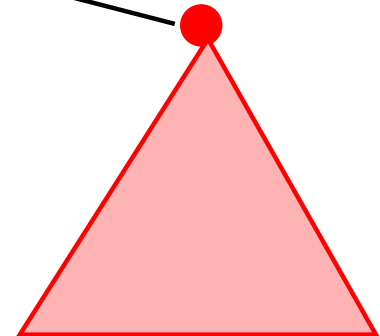


$\Rightarrow$

$rank[x] + 1$



$rank[y]$



# Melhoramento 1: *union by rank*

UNION ( $x, y$ )  $\triangleright$  com “union by rank”

1  $x' \leftarrow \text{FINDSET}(x)$

2  $y' \leftarrow \text{FINDSET}(y)$   $\triangleright$  supõe que  $x' \neq y'$

3 **se**  $\text{rank}[x'] > \text{rank}[y']$

4     **então**  $\text{pai}[y'] \leftarrow x'$

5     **senão**  $\text{pai}[x'] \leftarrow y'$

6             **se**  $\text{rank}[x'] = \text{rank}[y']$

7                 **então**  $\text{rank}[y] \leftarrow \text{rank}[y] + 1$

# Melhoramento 1: estrutura

- $rank[x] \leq rank[pai[x]]$  para cada nó  $x$
- $rank[x] = rank[pai[x]]$  se e só se  $x$  é raiz
- $rank[pai[x]]$  é uma função não-decrescente do tempo
- número de nós de uma árvore de raiz  $x$  é  $\geq 2^{rank[x]}$ .
- número de nós de posto  $k$  é  $\leq n/2^k$ .
- $altura(x) = rank[x] \leq \lg n$  para cada nó  $x$

$altura(x) :=$  comprimento do mais longo caminho que vai de  $x$  até uma folha

# Melhoramento 1: custo

Seqüência de operações MAKESET, UNION, FINDSET

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

⏟

$n$

⏟

$m$

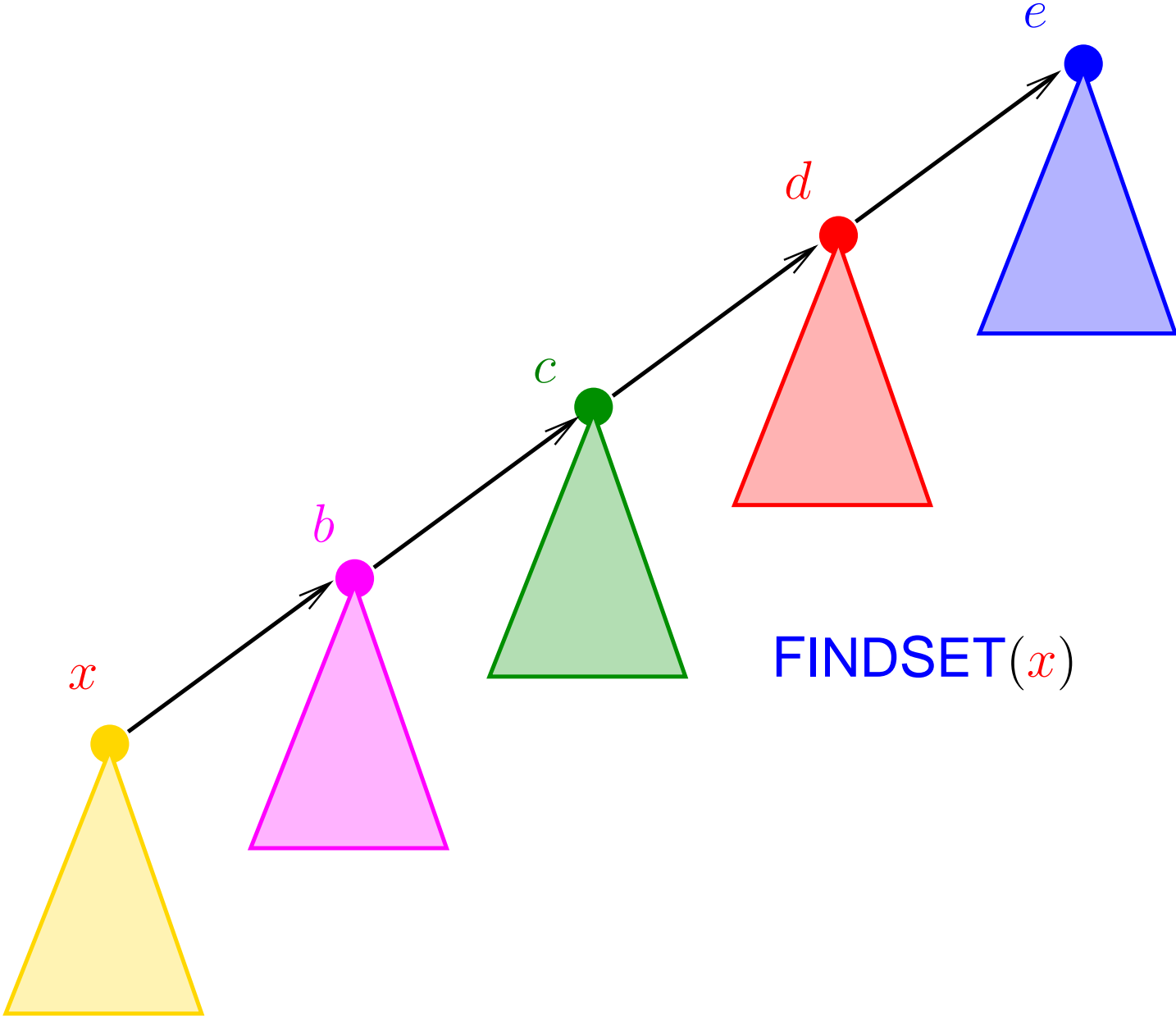
$altura(x) \leq \lg n$  para cada nó  $x$

Consumos de tempo:	MAKESET	$\Theta(1)$
	UNION	$O(\lg n)$
	FINDSET	$O(\lg n)$

Consumo de tempo total da seqüência:  $O(m \lg n)$

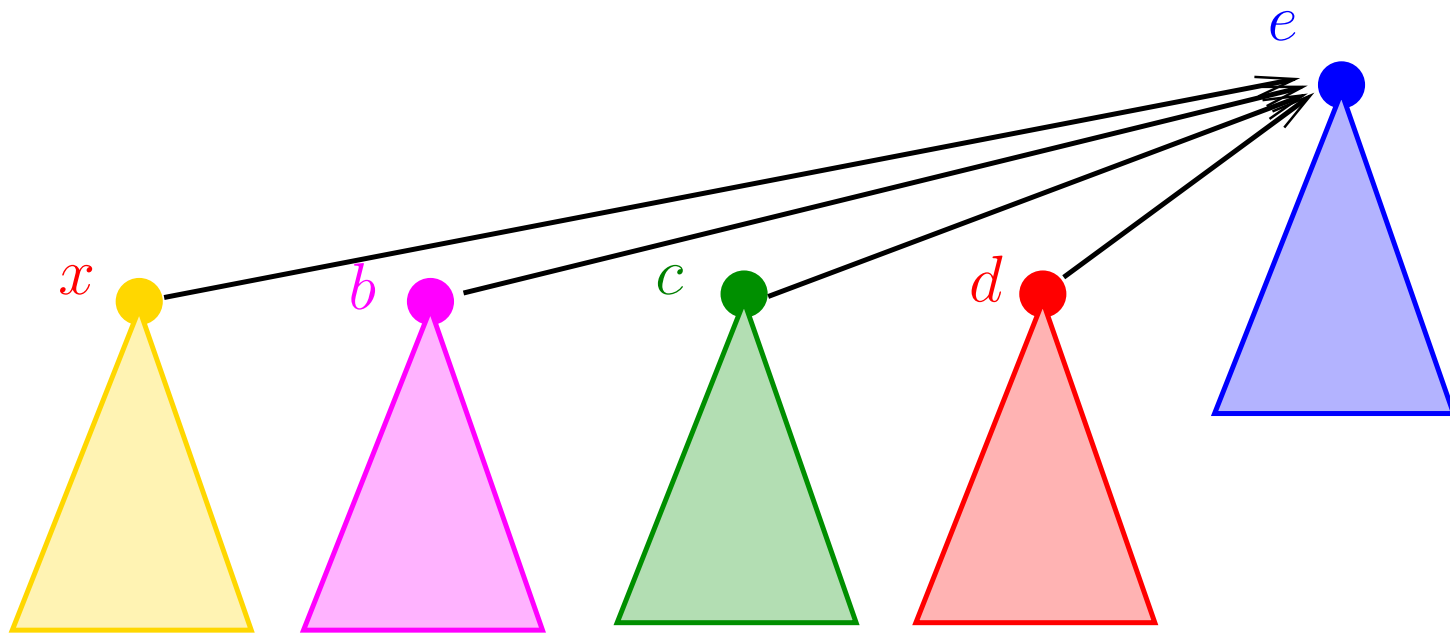
# AULA 23

# Melhoramento 2: *path compression*





# Melhoramento 2: *path compression*



FINDSET( $x$ )

# Melhoramento 2: *path compression*

**FINDSET** ( $x$ ) ▷ com “path compression”

1    **se**  $x \neq \text{pai}[x]$

2        **então**  $\text{pai}[x] \leftarrow \text{FINDSET}(\text{pai}[x])$

3    **devolva**  $\text{pai}[x]$

- $\text{rank}[x] \leq \text{rank}[\text{pai}[x]]$  para cada nó  $x$
- $\text{rank}[x] = \text{rank}[\text{pai}[x]]$  se e só se  $x$  é raiz
- $\text{rank}[\text{pai}[x]]$  é uma função não-decrescente do tempo
- número de nós de uma árvore de raiz  $x$  é  $\geq 2^{\text{rank}[x]}$
- número de nós de posto  $k$  é  $\leq n/2^k$
- $\text{altura}(x) \leq \text{rank}[x] \leq \lg n$  para cada nó  $x$

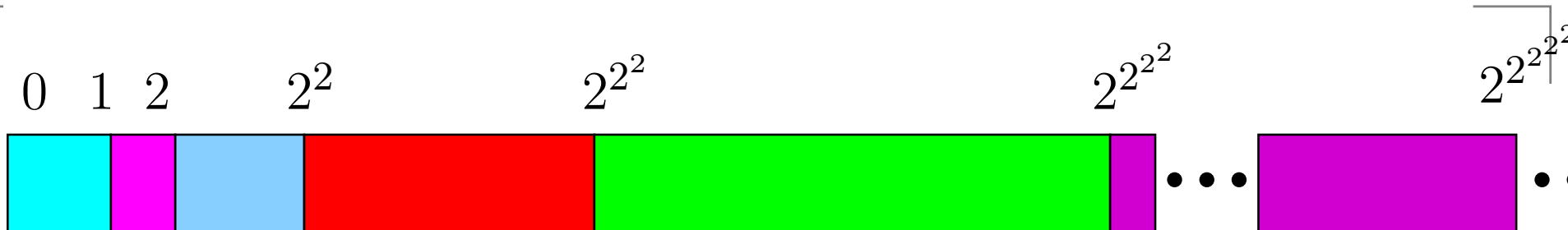


# Função 'torre'

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

$i$	$t(i)$
0	1
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^{2^2} = 16$
4	$2^{2^{2^2}} = 2^{16} = 65536$
5	$2^{2^{2^{2^2}}} > \underbrace{100000000000000000000 \dots 00000000000000000000}_{80}$
$\vdots$	$\vdots$

# Blocos



$$\text{bloco}[0] = [0 .. 1]$$

$$\text{bloco}[1] = [2 .. 2]$$

$$\text{bloco}[2] = [3 .. 4]$$

$$\text{bloco}[3] = [5 .. 16]$$

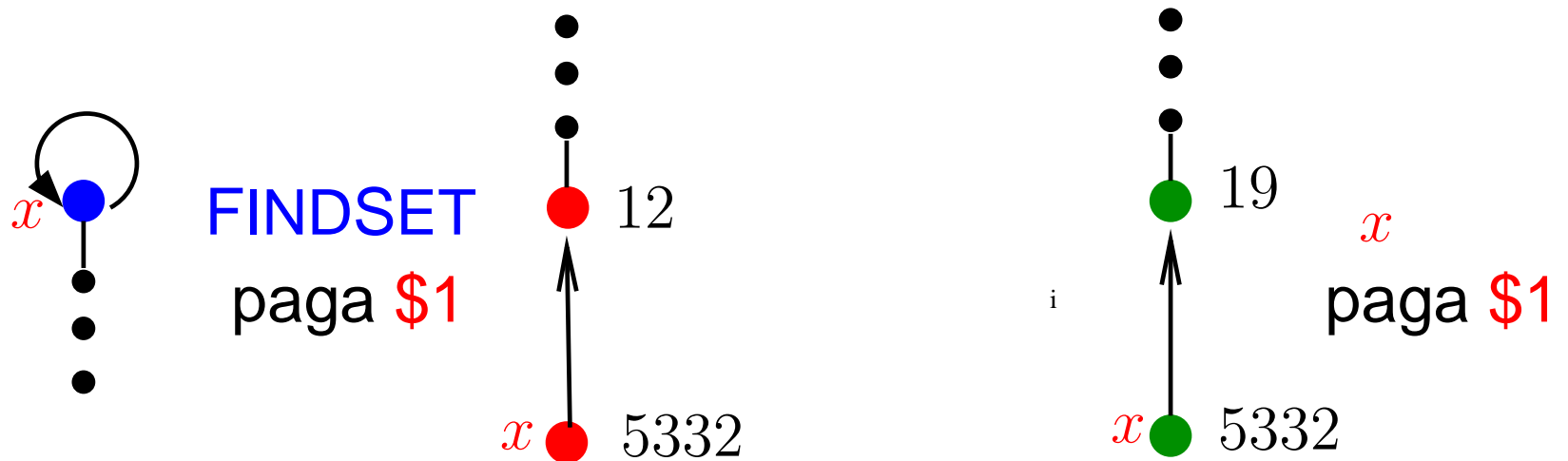
$$\text{bloco}[i] = [t(i-1)+1 .. t(i)]$$

# Contabilidade

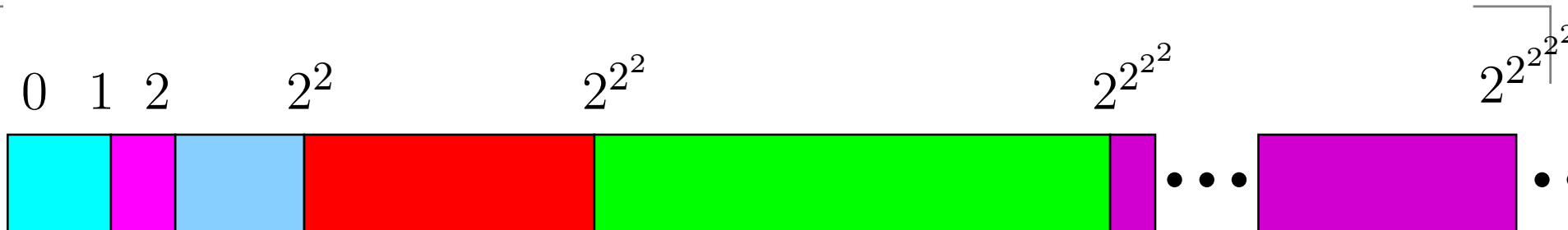
Custo de cada operação  $\text{FINDSET}(y)$  será contabilizado da seguinte maneira.

Para cada nó  $x$  no caminho de  $y$  até raiz:

- se  $x$  é a raiz ou  $\text{rank}[x]$  e  $\text{rank}[\text{pai}[x]]$  estão em blocos diferentes cobre \$1 da operação  $\text{FINDSET}$
- se  $\text{rank}[x]$  e  $\text{rank}[\text{pai}[x]]$  estão no mesmo bloco cobre \$1 de  $x$ .



# Pagamento de cada FindSet



Cada operação **FINDSET** paga  $O(\lg^* n)$ :

- $\text{rank}[x] \leq \log n$  para cada nó  $x$
- há vértices em  $\leq \lg^* n$  blocos:

$$\begin{aligned} t(i-1) &< \lg n \leq t(i) \\ 0 &< \lg^i(\lg n) \leq 1 \end{aligned}$$

$$\Rightarrow i \leq \lg^* \lg n < \lg^* n$$

# Pagamento de cada vértice

Suponha que  $x$  não é raiz.

Se  $\text{rank}[x]$  está em  $\text{bloco}[i]$ , então  
 $x$  paga  $\leq t(i) - t(i - 1)$ .

Seja  $N(i)$  o número de vértices em  $\text{bloco}[i]$ ,  $i > 0$ .  
Temos que

$$\begin{aligned} N(i) &\leq \frac{n}{2^{t(i-1)+1}} + \frac{n}{2^{t(i-1)+2}} + \cdots + \frac{n}{2^{t(i)}} \\ &< \frac{n}{2^{t(i-1)+1}} (1 + 1/2 + 1/4 + \cdots) \\ &= \frac{n}{2^{t(i-1)}} \\ &= \frac{n}{t(i)} \end{aligned}$$



# Pagamento de todos os vértices

Para cada  $i$  o valor pago por vértices em  $bloco[i]$  é limitado por

$$\frac{n}{t(i)} \times (t(i) - t(i - 1)) < n$$

Para cada  $x$  tem-se que  $rank[x] \leq \lg n$   
 $\Rightarrow$  há vértices em  $< \lg^* n$  blocos

Portanto, os vértices da *disjoint-set forest* pagam um total de  $< n \lg^* n$ .

Custo total da seqüência  $O(m \lg^* n + n \lg^* n) = O(m \lg^* n)$

# Conclusão

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

# Exercícios

## Exercício 24.A [CLRS 21.1-3]

Quando **CONNECTED-COMPONENTS** é aplicado a um grafo  $G = (V, E)$  com  $k$  componentes, quantas vezes **FINDSET** é chamado? Quantas vezes **UNION** é chamado? Dê respostas em termos de  $k$ ,  $|V|$  e  $|E|$ .

## Exercício 24.B [CLRS 21.3-1]

Faça uma figura da floresta produzida pela seguinte seqüência de operações:

```
01   para  $i \leftarrow 1$  até 16
02       faça MAKESET ( $x_i$ )
03   para  $i \leftarrow 1$  até 15 em passos de 2
04       faça UNION ( $x_i, x_{i+1}$ )
05   para  $i \leftarrow 1$  até 13 em passos de 4
06       faça UNION ( $x_i, x_{i+2}$ )
07   UNION ( $x_1, x_5$ )
08   UNION ( $x_{11}, x_{13}$ )
09   UNION ( $x_1, x_{10}$ )
10   FINDSET ( $x_2$ )
11   FINDSET ( $x_9$ )
```

# Mais exercícios

## Exercício 24.C [CLRS 21.3-2]

Escreva uma versão iterativa de **FINDSET** com “path compression”.

## Exercício 24.D [CLRS 21.3-3]

Dê uma seqüência de  $m$  **MAKESET**, **UNION** e **FINDSET**<sub>0</sub>,  $n$  das quais são **MAKESET**, que consome  $\Omega(m \lg n)$ .

## Exercício 24.E

Digamos que  $h[x]$  é a altura do nó  $x$  (= comprimento do mais longo caminho que vai de  $x$  até uma folha) na estrutura disjoint-set forest. Mostre que  $rank[x] \geq h[x]$ . Mostre que **UNION**( $x, y$ ) nem sempre pendura a árvore mais baixa na mais alta.

## Exercício 24.F [CLRS 21.4-2]

Mostre que o pôsto de cada nó na estrutura *disjoint-set forest* é no máximo  $\lfloor \lg n \rfloor$  ou seja, que  $rank[x] \leq \lg n$ . (Sugestão: Mostre inicialmente que para cada raiz  $x$  temos  $2^{rank[x]} \leq n_x$ , onde  $n_x$  é o número de nós na árvore que contém  $x$ .)

# Mais um exercício

## Exercício 24.G [CLRS 21.4-4]

Considere uma versão simplificada da estrutura *disjoint-set forest* que usa a heurística “union by rank” mas não usa a heurística “path compression”. (Em outras palavras, usa as operações **MAKESET**, **UNION** e **FINDSET**.) Mostre que essa simplificação consome tempo

$$O(m \lg n).$$

Como de hábito,  $m$  é o número total de operações e  $n$  é o número de operações **MAKESET**. (Sugestão: Use o exercício CLRS 21.4-2.)