

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 $i \neq j$.

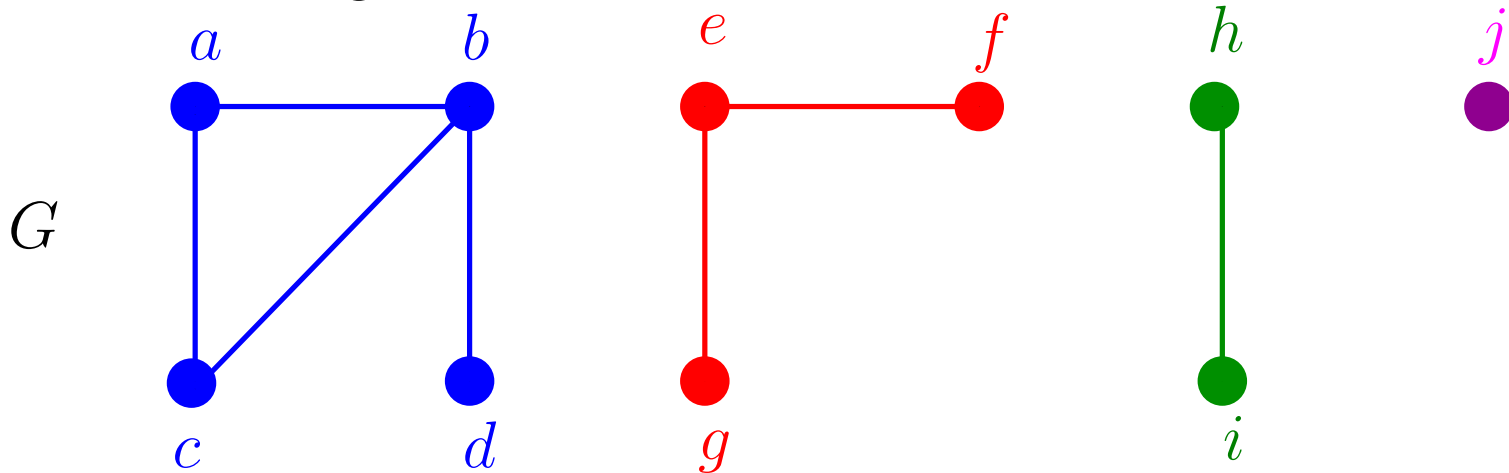
Conjuntos disjuntos

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

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

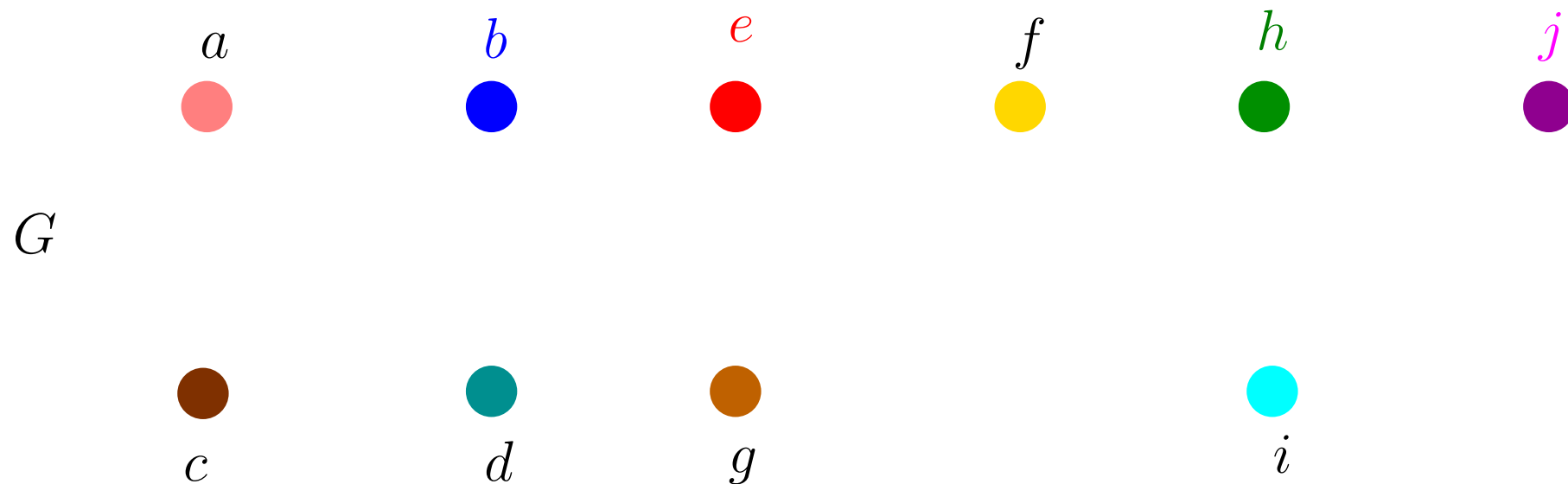
Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico



aresta

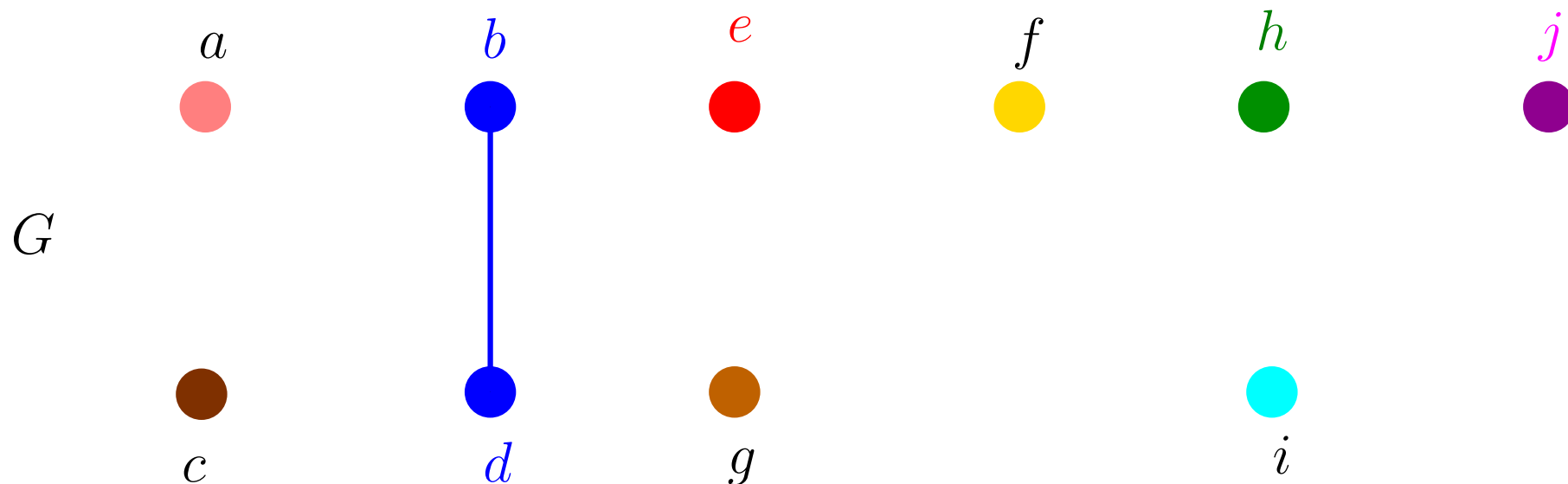
componentes

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

Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: **grafo dinâmico**



aresta

componentes

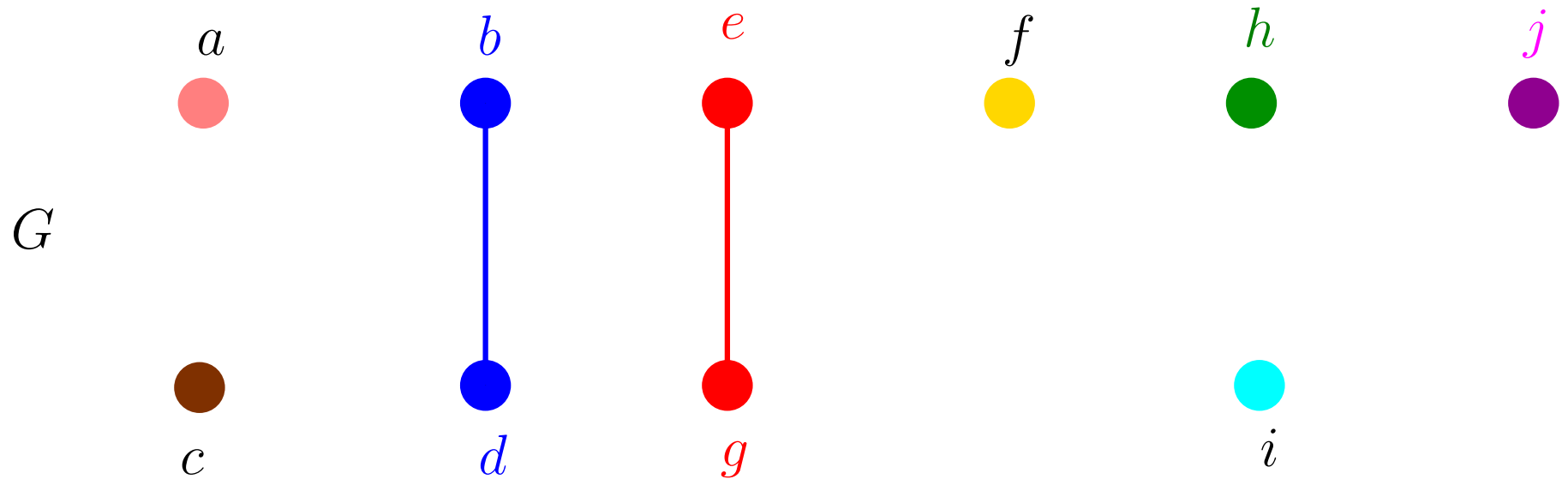
(b, d)

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

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico



aresta

componentes

(e, g)

$\{a\}$

$\{b, d\}$

$\{c\}$

$\{e, g\}$

$\{f\}$

$\{h\}$

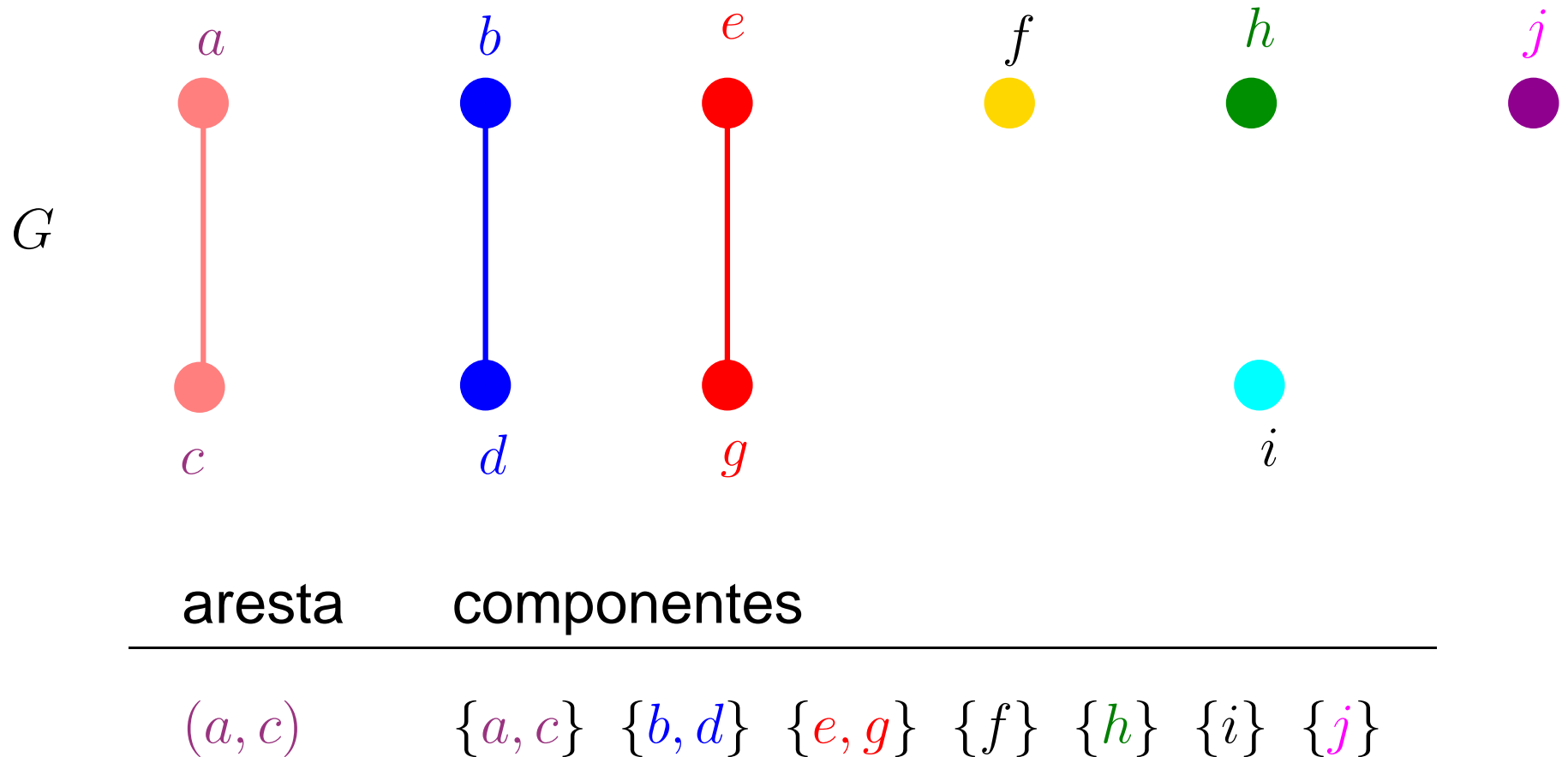
$\{i\}$

$\{j\}$

Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

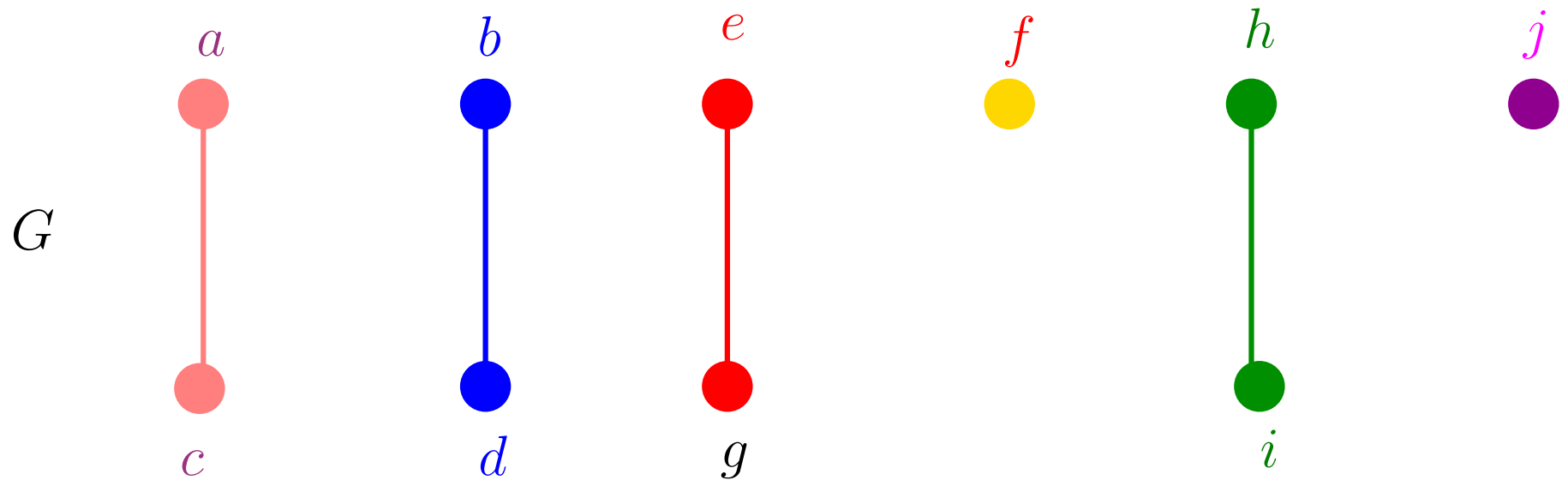
Exemplo: **grafo dinâmico**



Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: **grafo dinâmico**



aresta

componentes

(h, i)

$\{a, c\}$

$\{b, d\}$

$\{e, g\}$

$\{f\}$

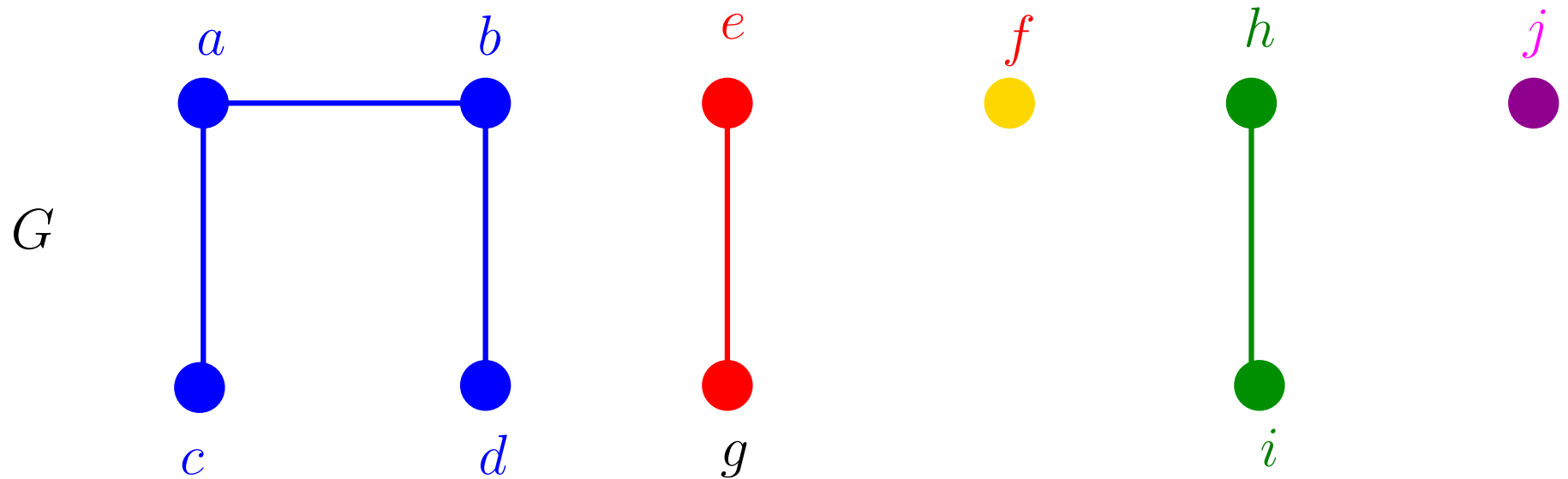
$\{h, i\}$

$\{j\}$

Coleção disjunta dinâmica

Conjuntos são **modificados ao longo do tempo**

Exemplo: **grafo dinâmico**



aresta

componentes

(a, b)

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

$\{e, g\}$

$\{f\}$

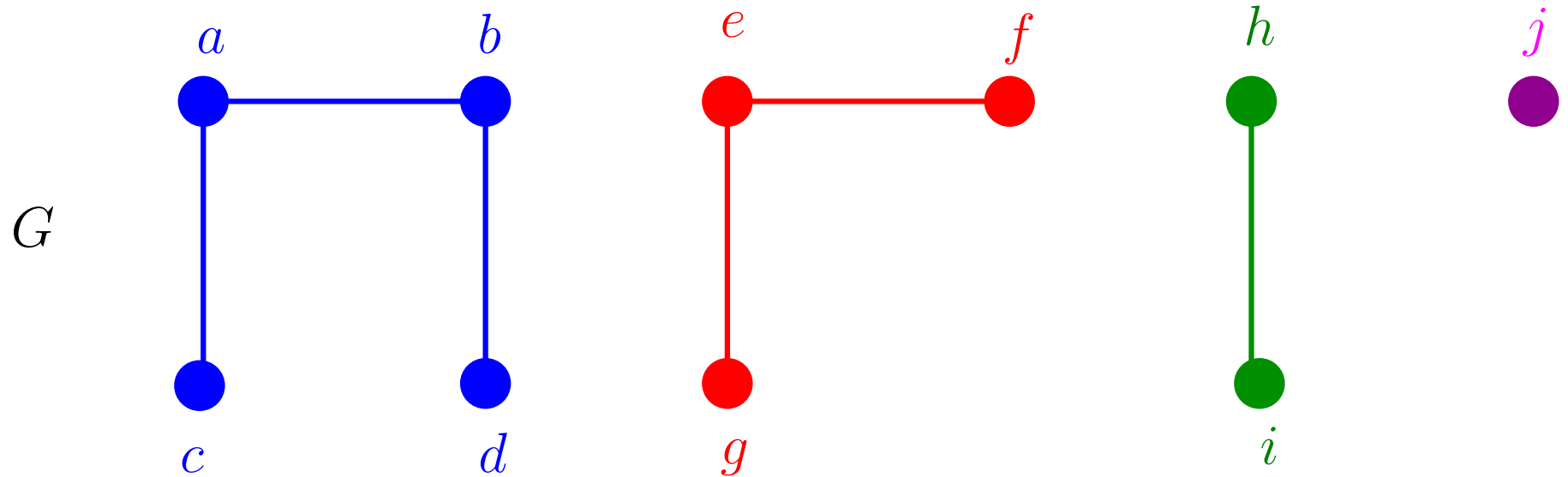
$\{h, i\}$

$\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico



aresta

componentes

(e, f)

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

$\{e, f, g\}$

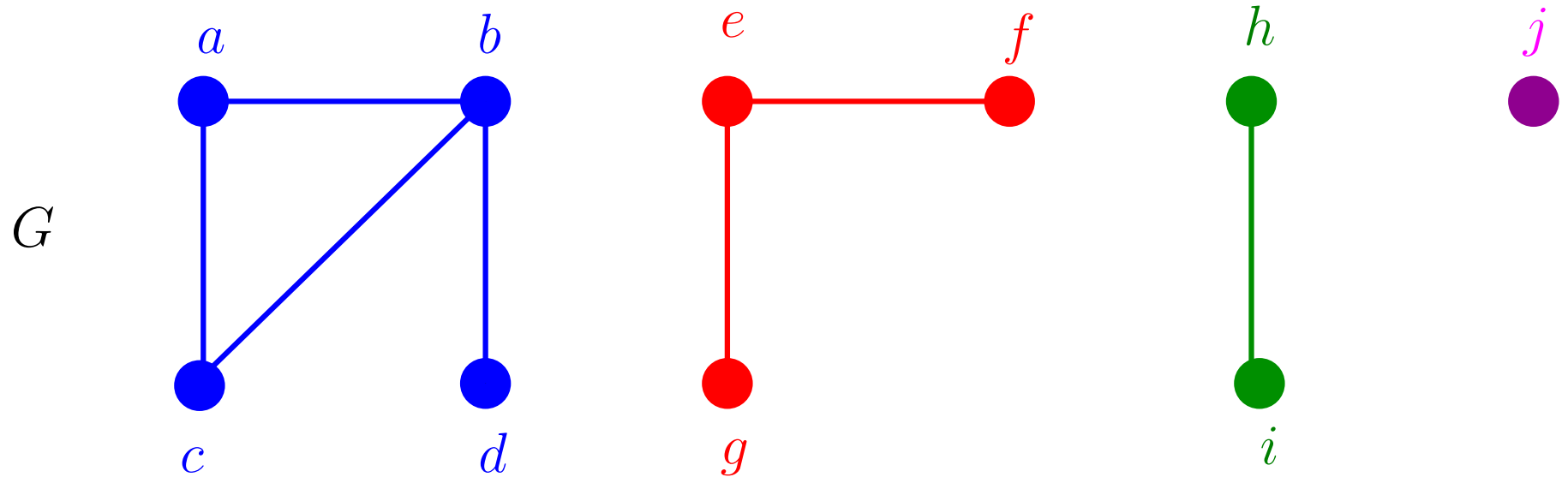
$\{h, i\}$

$\{j\}$

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 co

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)**

Consumo de tempo

n := número de vértices do grafo

m := número de arestas do grafo

linha consumo de **todas** as execuções da linha

$$1 = \Theta(n)$$

$$2 = n \times \text{consumo de tempo MAKESET}$$

$$3 = \Theta(m)$$

$$4 = 2m \times \text{consumo de tempo FINDSET}$$

$$5 \leq n \times \text{consumo de tempo UNION}$$

$$\begin{aligned} \text{total} &\leq \Theta(n + m) + n \times \text{consumo de tempo MAKESET} \\ &\quad + 2m \times \text{consumo de tempo FINDSET} \\ &\quad + n \times \text{consumo de tempo UNION} \end{aligned}$$

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) $\triangleright G$ conexo

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

“Avô” de todos os algoritmos gulosos.

Conjuntos disjuntos dinâmicos

Seqüência de operações MAKESET, UNION, FINDSET

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



n

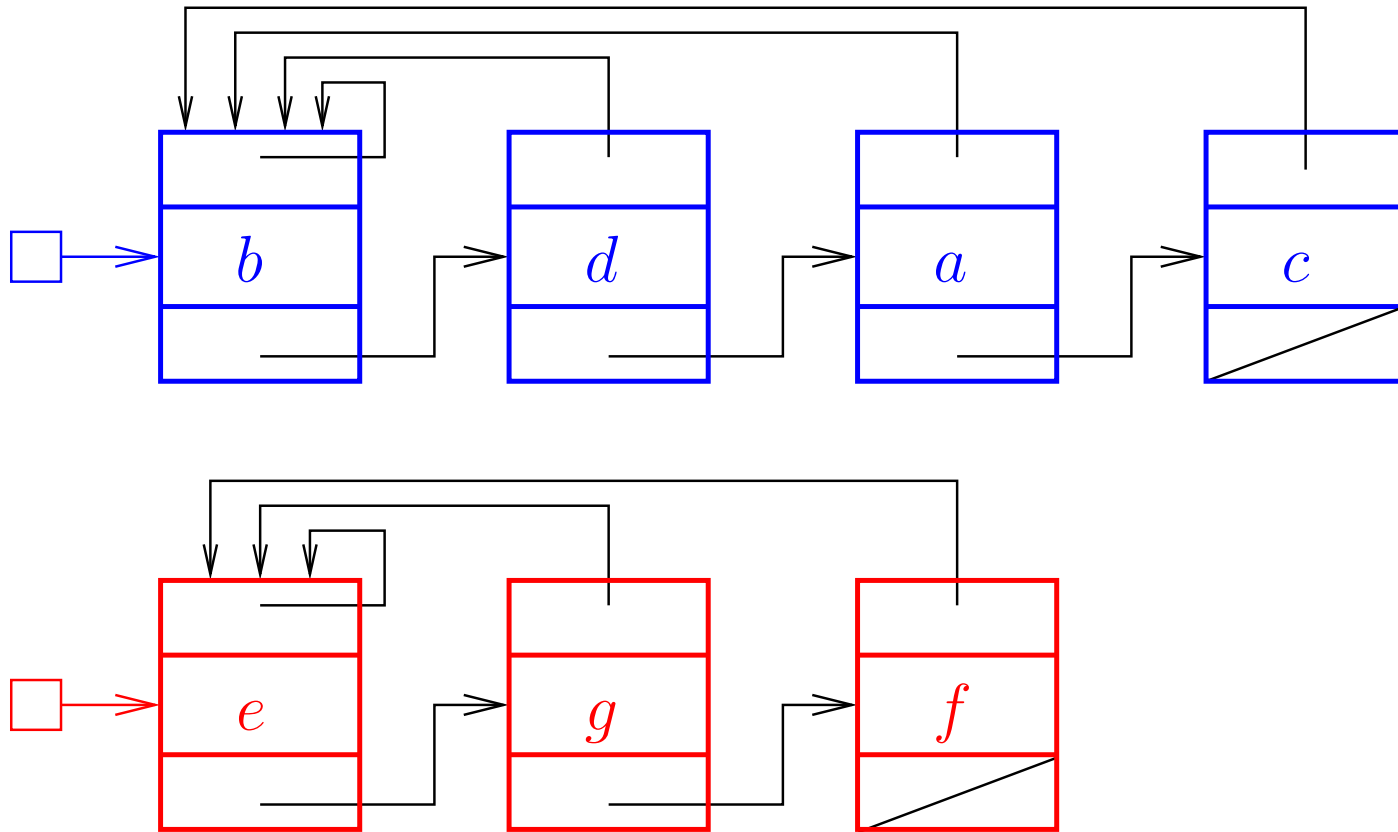


m

Que estrutura de dados usar?

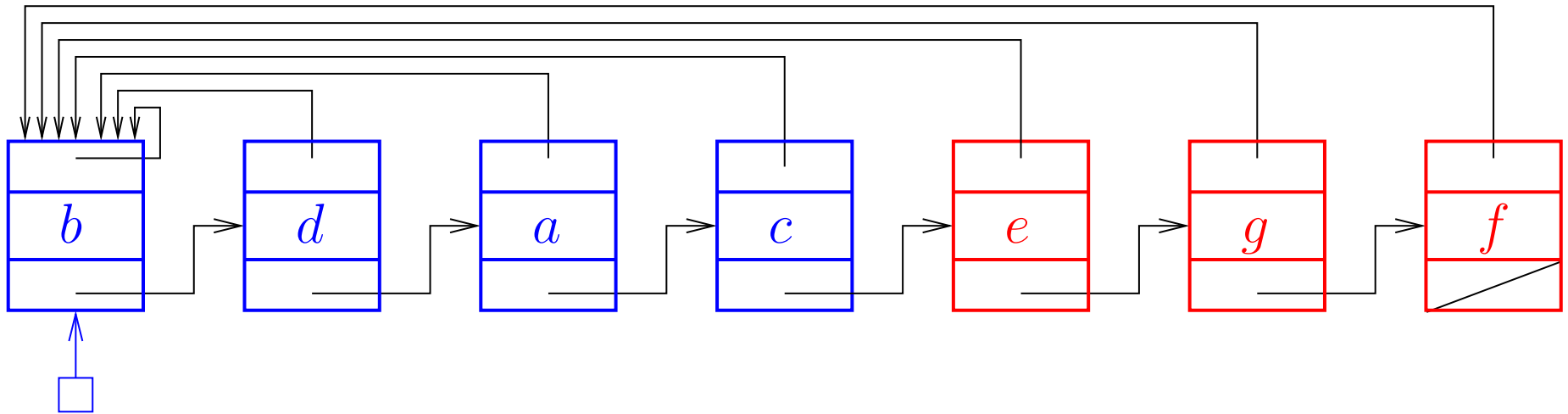
Compromissos (*trade-offs*)

Estrutura lista ligada



- cada conjunto tem um representante (início da lista)
- cada nó x tem um campo $repr$
- $repr[x]$ é o representante do conjunto que contém x

Estrutura lista ligada



UNION (*a*, *e*) : atualiza apontador para o representante.

Consumo de tempo

| Operação | número de objetos atualizados |
|-------------------------|-------------------------------|
| MAKESET(x_1) | 1 |
| MAKESET(x_2) | 1 |
| ⋮ | 1 |
| MAKESET(x_n) | 1 |
| UNION(x_1, x_2) | 1 |
| UNION(x_2, x_3) | 2 |
| ⋮ | ⋮ |
| UNION(x_{n-1}, x_n) | $n - 1$ |

total $= \Theta(n^2) = \Theta(m^2)$

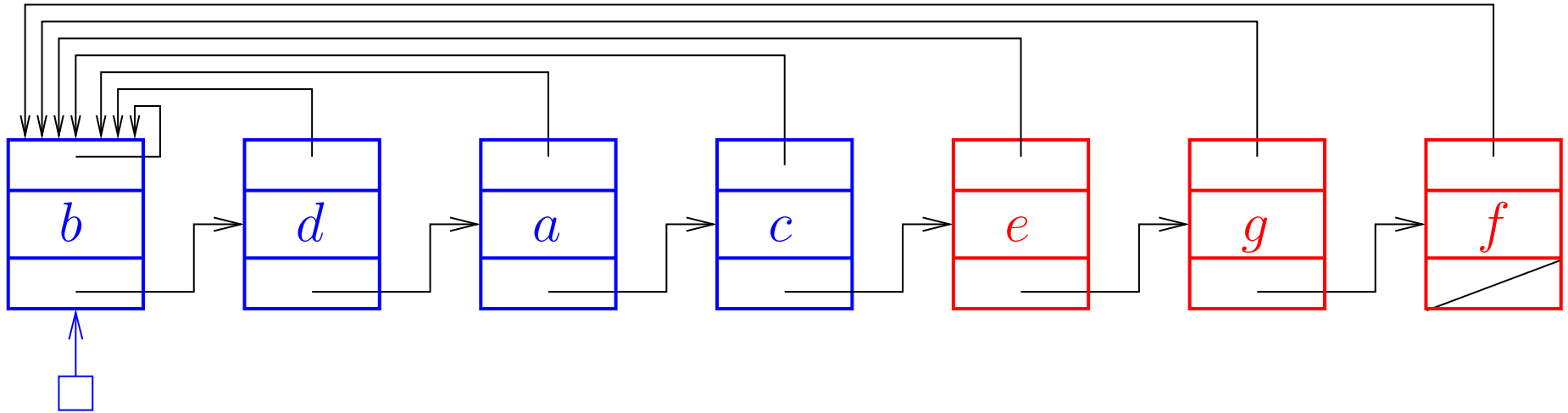
Consumo de tempo

| | |
|---------|-------------|
| MAKESET | $\Theta(1)$ |
| UNION | $O(n)$ |
| FINDSET | $\Theta(1)$ |

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

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

Melhoramento: *weighted-union*



- cada representante armazena o comprimento da lista
- a lista menor é concatenada com a maior

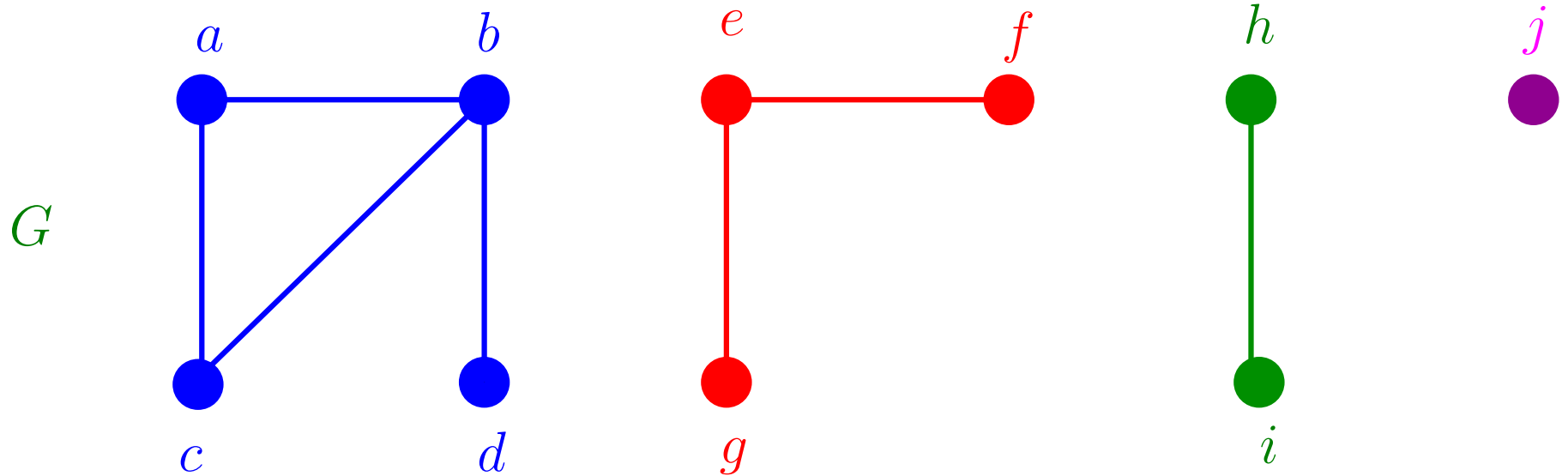
Cada objeto x é atualizado $\leq \lg n$:

cada vez que x é atualizado o tamanho da lista dobra.

Conclusão

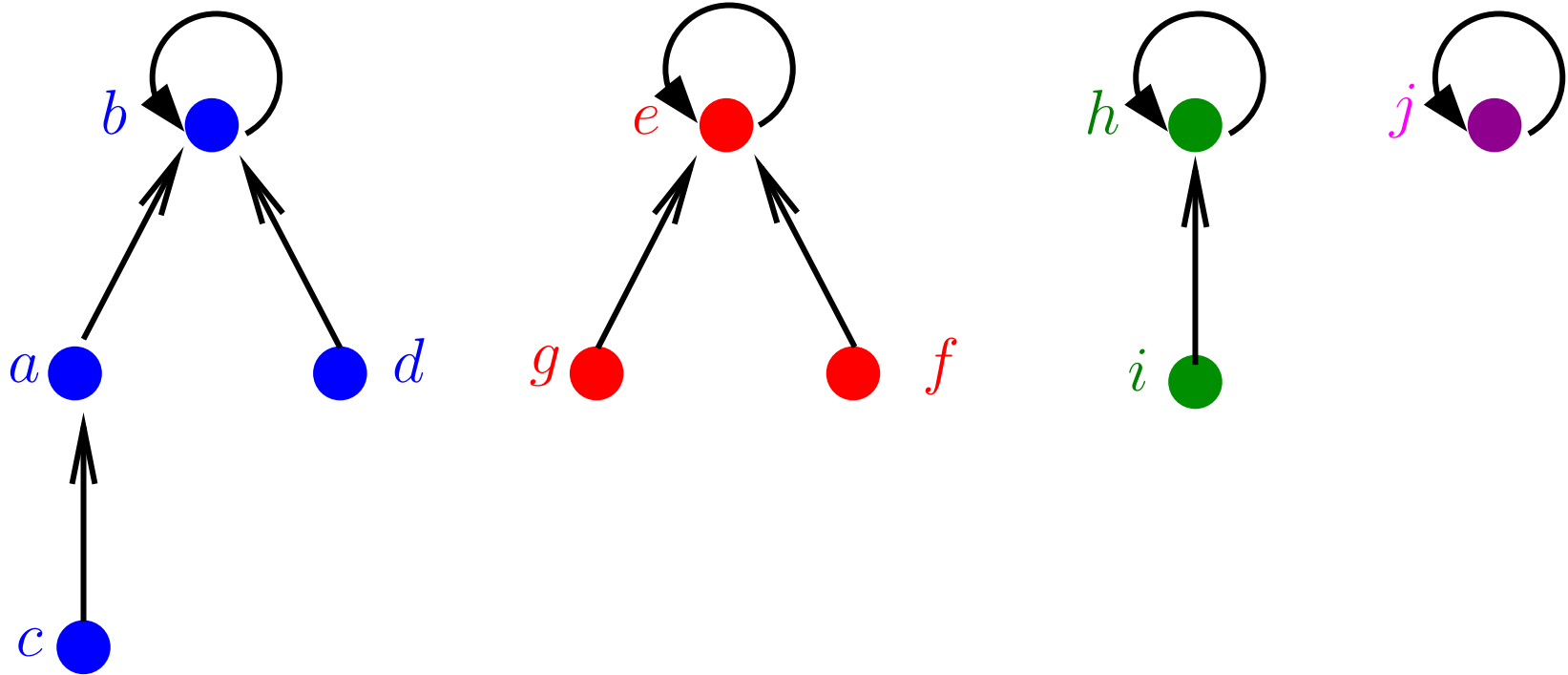
Se conjuntos disjuntos são representados através de *listas ligadas* e *weighted-union* é utilizada, então uma seqüência de m operações MAKESET, UNION e FINDSET, sendo que n são MAKESET, consome tempo $O(m + n \lg n)$.

Estrutura *disjoint-set forest*



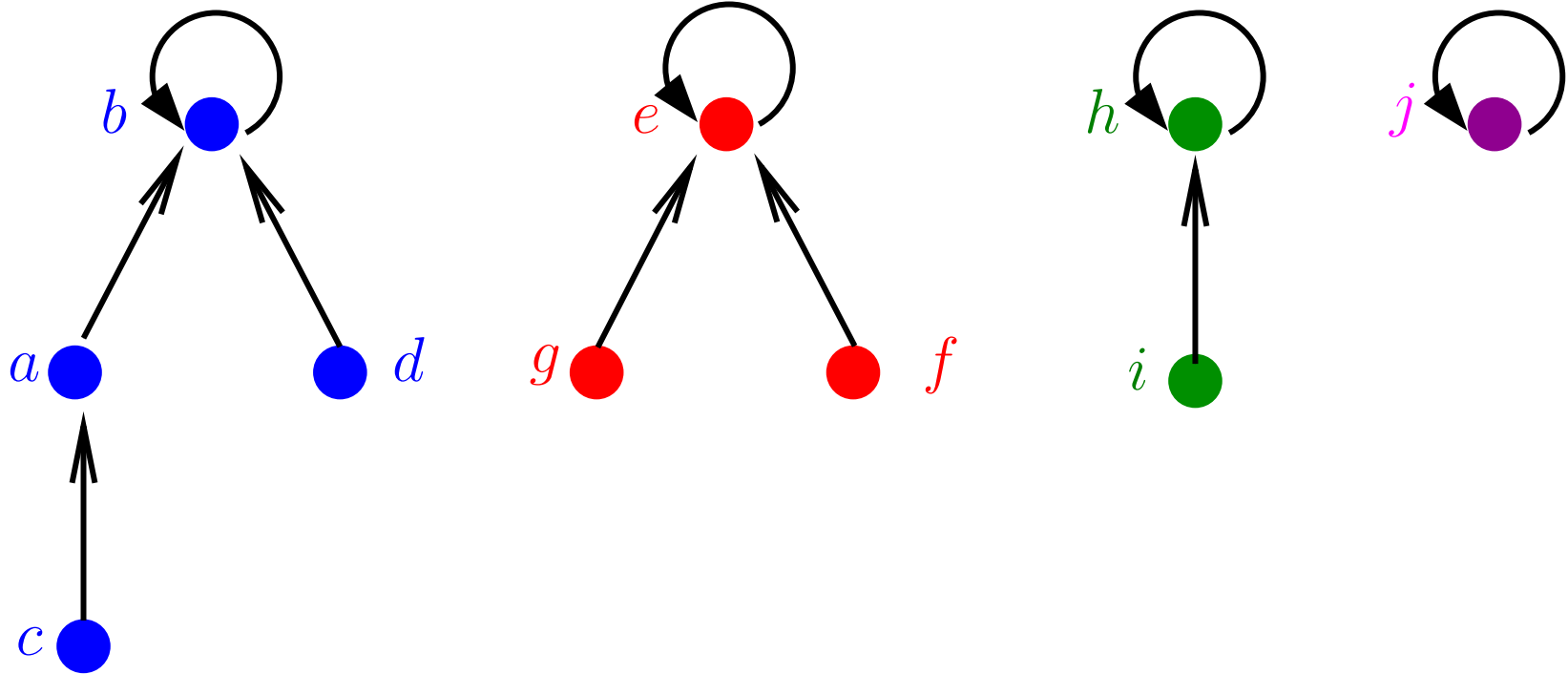
- cada conjunto tem uma *raiz*, que é o seu representante
- cada nó x tem um *pai*
- $\text{pai}[x] = x$ se e só se x é uma raiz

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

MakeSet₀ e FindSet₀



MAKESET₀ (x)

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

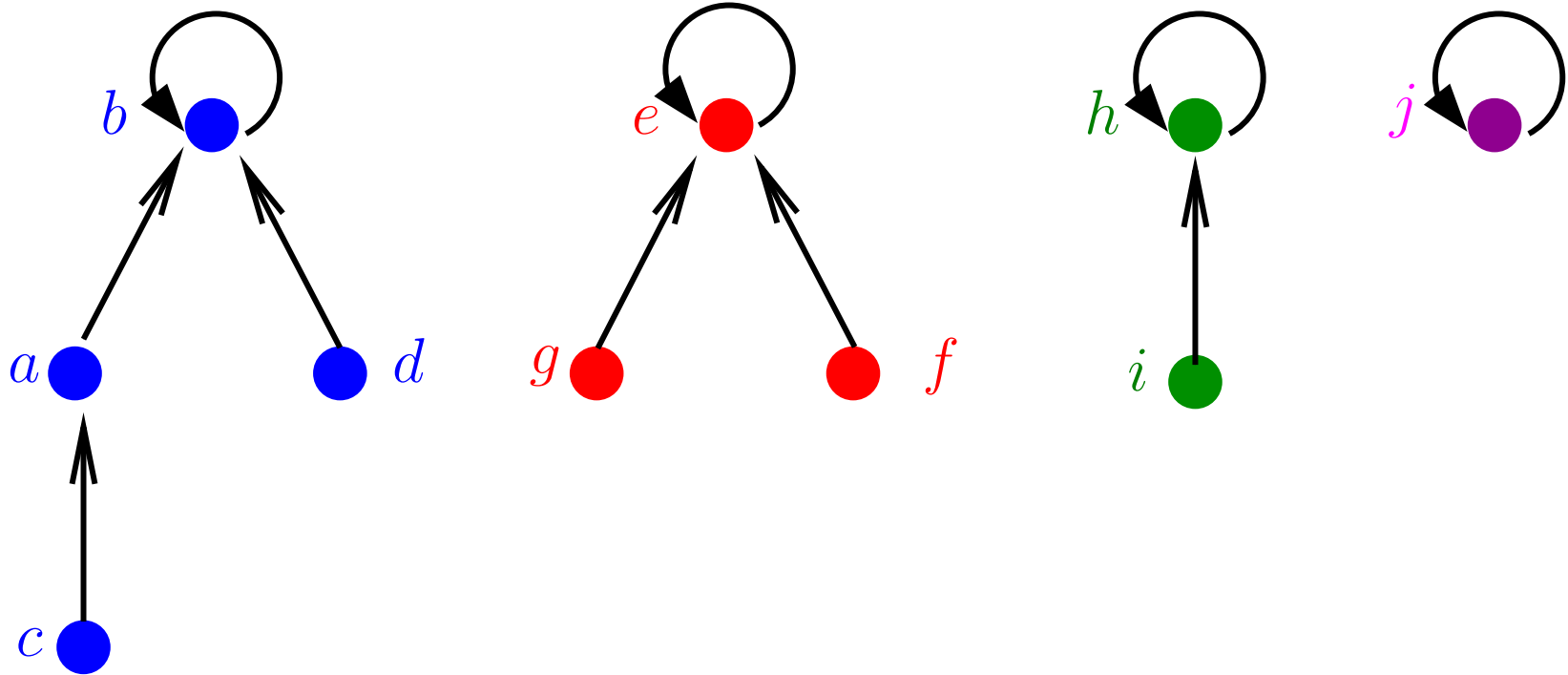
FINDSET₀ (x)

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

2 $x \leftarrow \text{pai}[x]$

3 **devolva** x

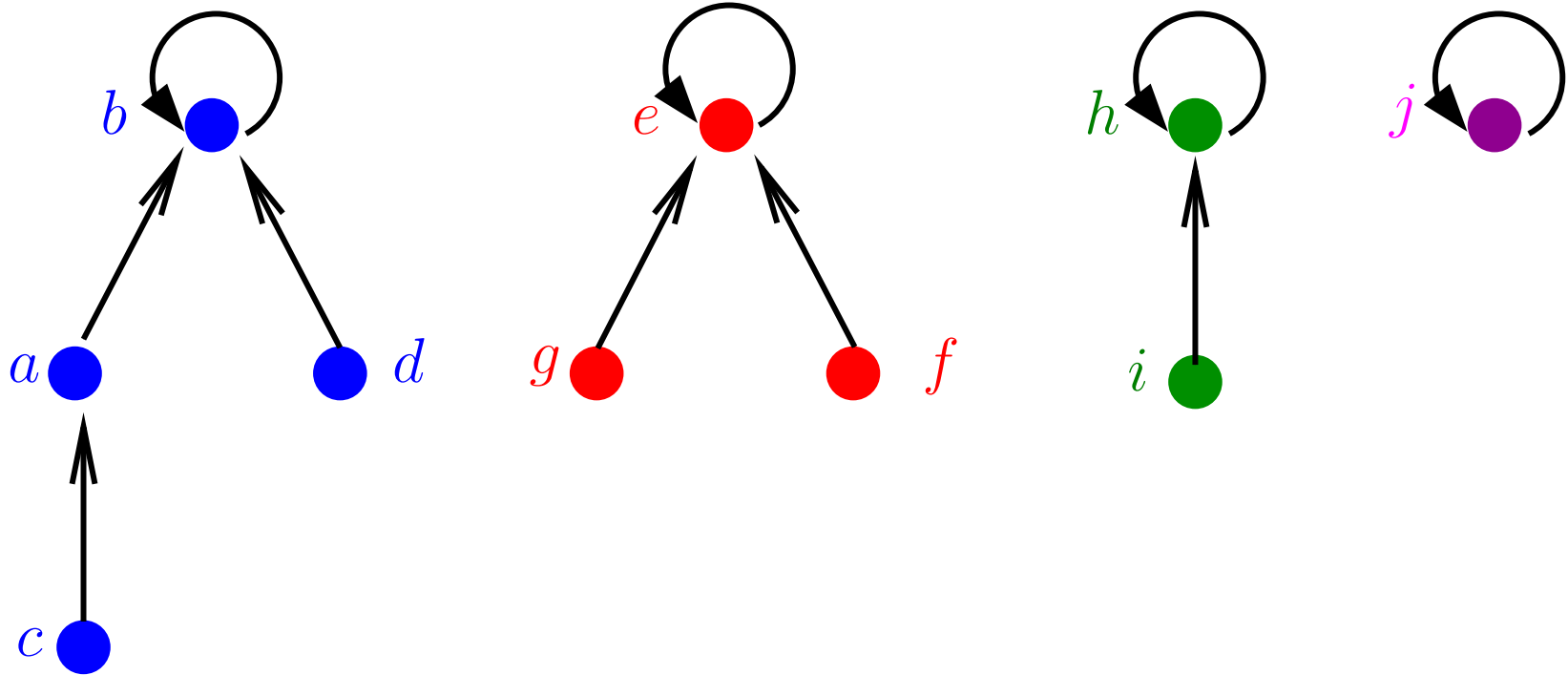
FindSet₁



FINDSET₁ (x)

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

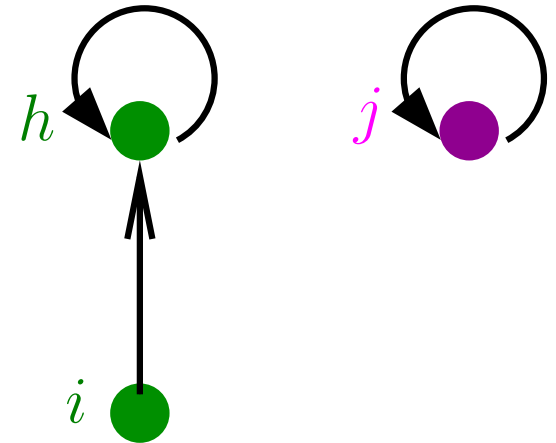
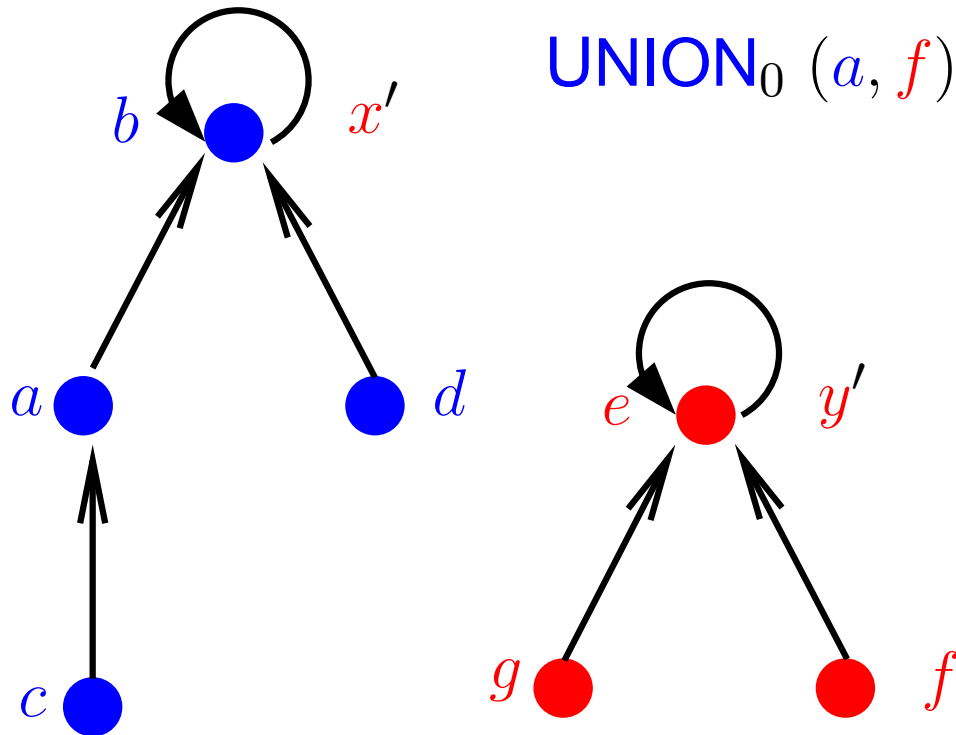
Union₀



UNION₀ (x, y)

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

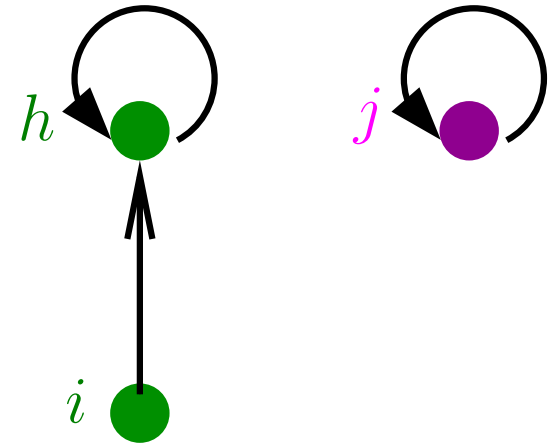
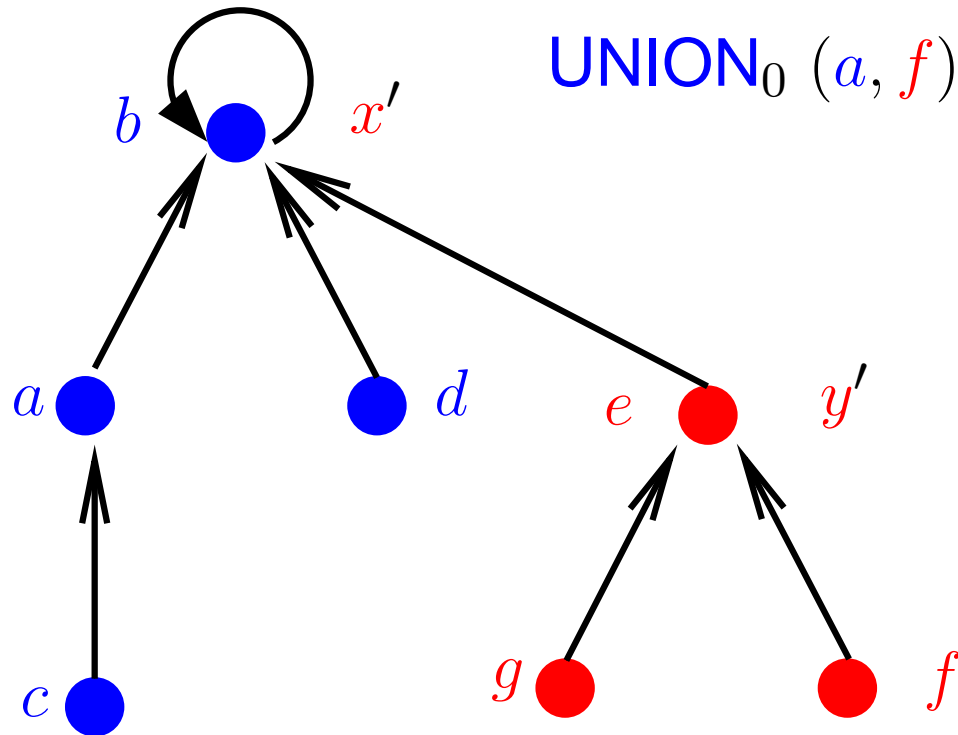
Union₀



UNION₀ (x, y)

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

Union₀



UNION₀ (x, y)

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

MakeSet₀, Union₀ e FindSet₁

MAKESET₀ (x)

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

UNION₀ (x, y)

1 $x' \leftarrow \text{FINDSET}_0(x)$

2 $y' \leftarrow \text{FINDSET}_0(y)$

3 $\text{pai}[y'] \leftarrow x'$

FINDSET₁ (x)

1 **se** $\text{pai}[x] = x$

2 **então devolva** x

3 **senão devolva** FINDSET₁ ($\text{pai}[x]$)

Consumo de tempo

| | |
|----------------------|-------------|
| MAKESET ₀ | $\Theta(1)$ |
| UNION ₀ | $O(n)$ |
| FINDSET ₀ | $O(n)$ |

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



n

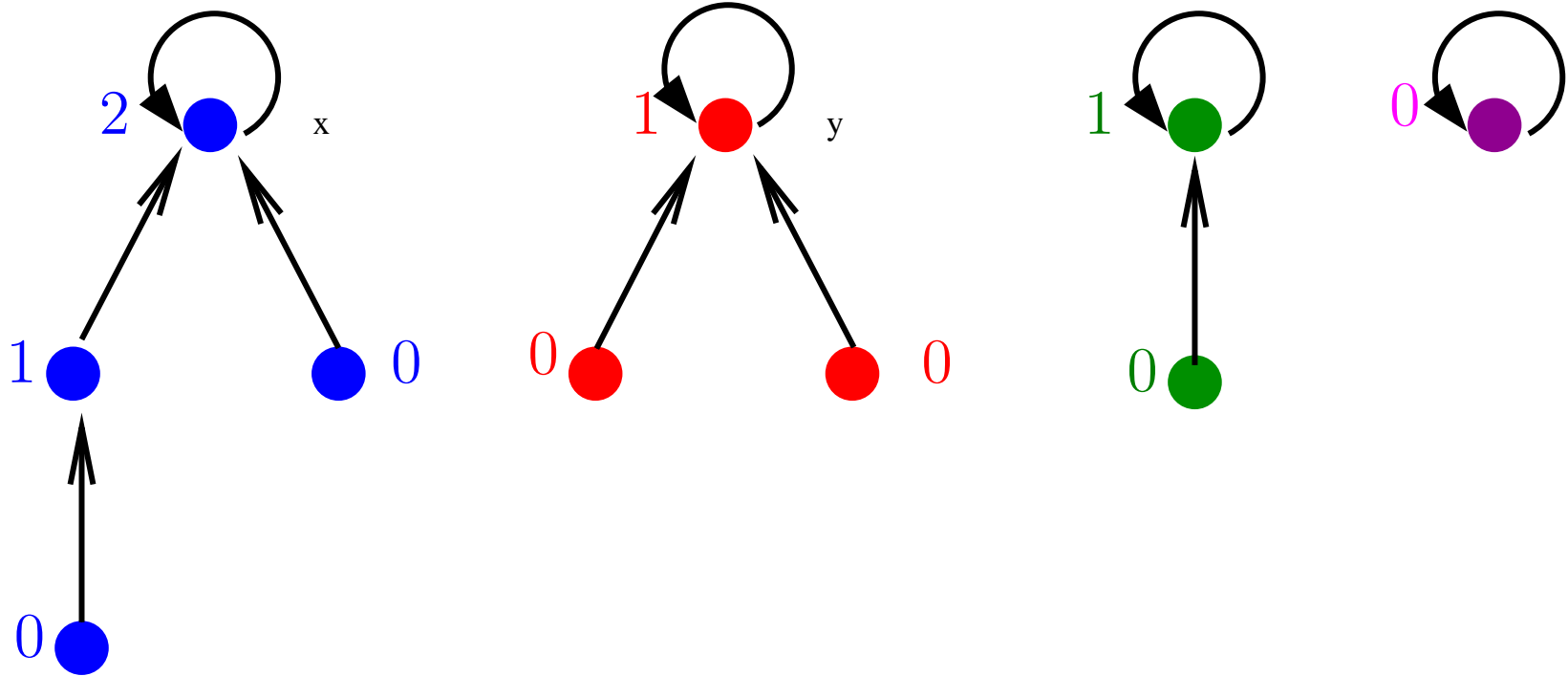


m

Custo total da seqüência:

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

Melhoramento 1: *union by rank*

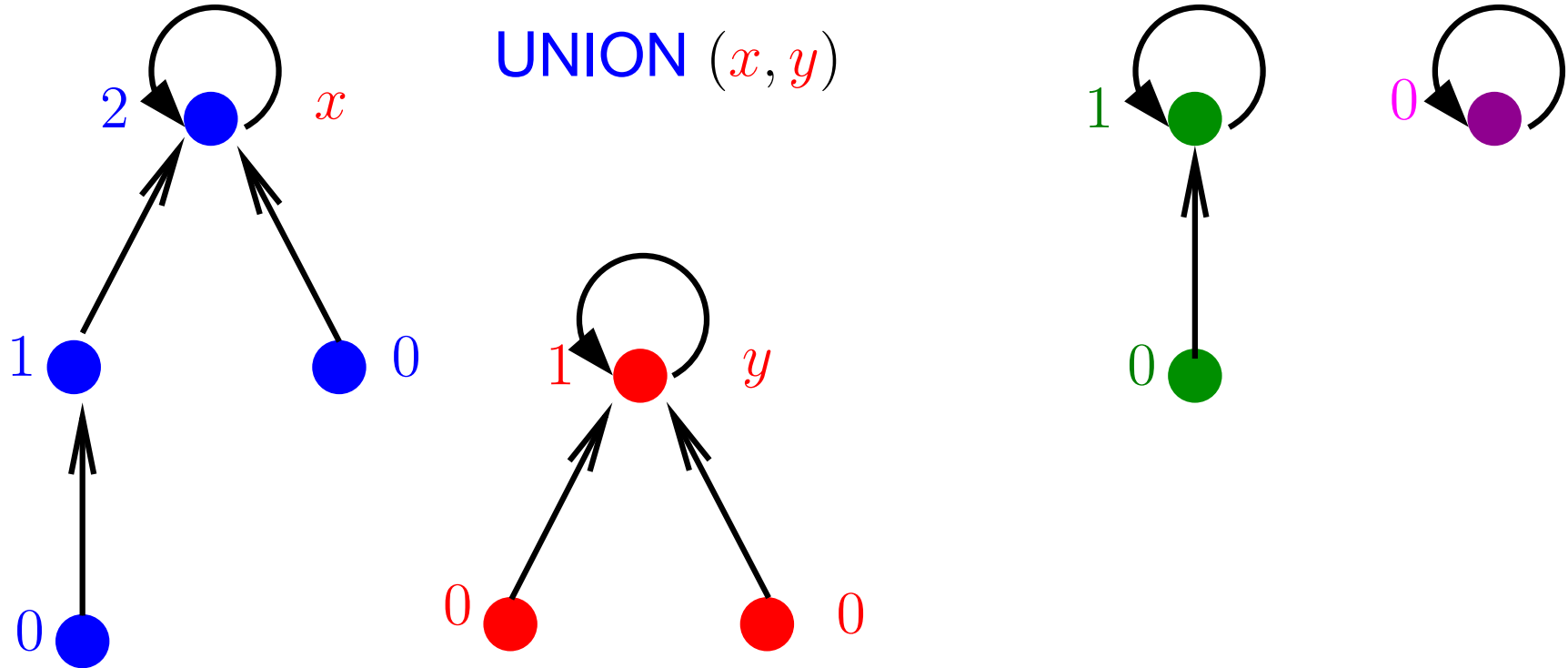


$rank[x]$ = posto do nó x

MAKESET (x)

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

Melhoramento 1: *union by rank*

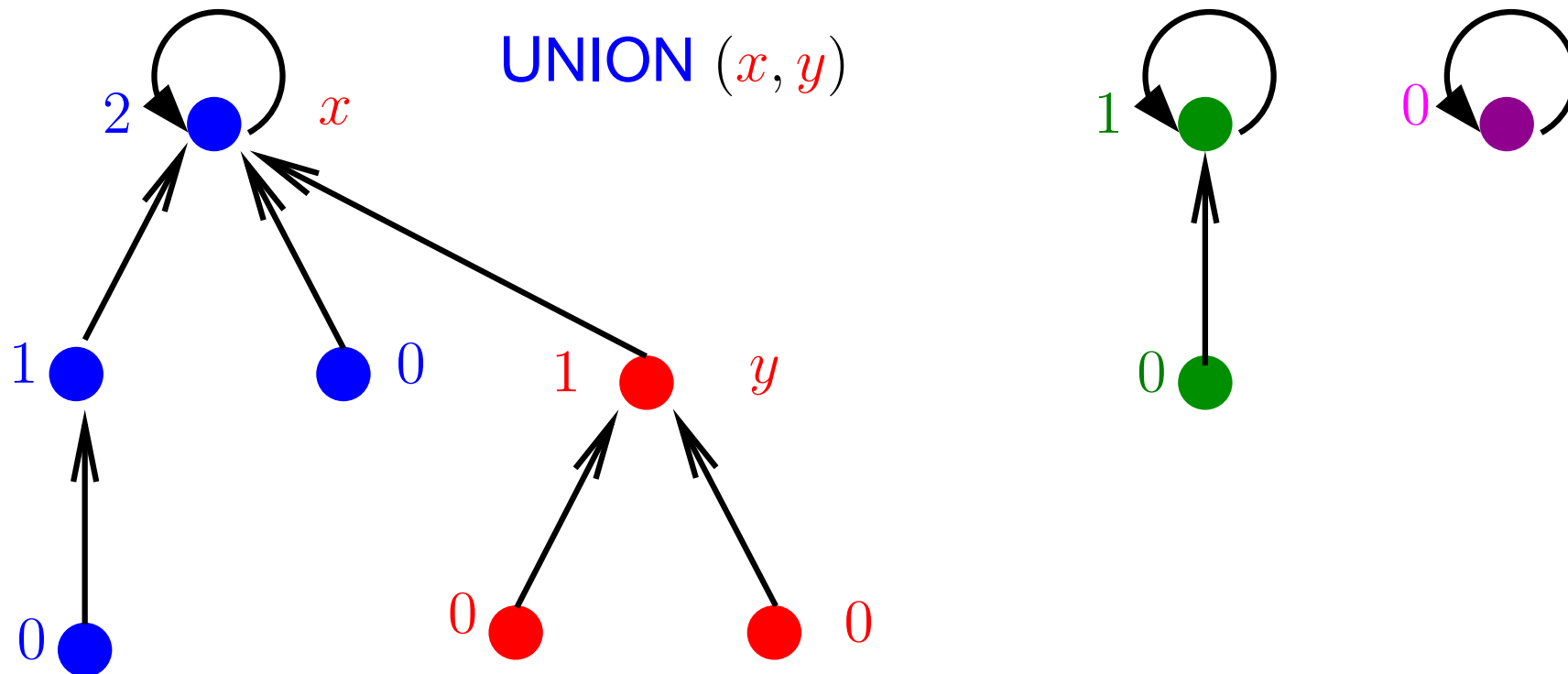


$rank[x]$ = posto do nó x

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 $pai[x] \leftarrow x$

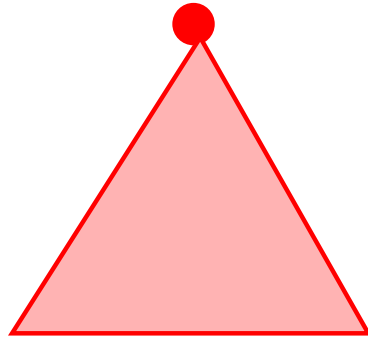
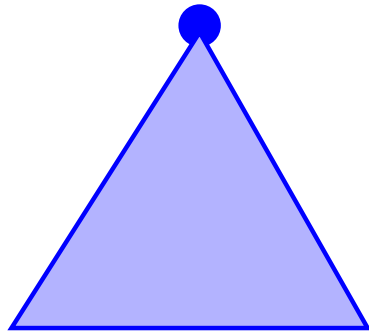
2 $rank[x] \leftarrow 0$

Melhoramento 1: *union by rank*

$rank[x]$

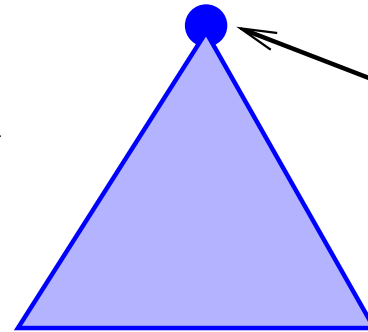
$>$

$rank[y]$

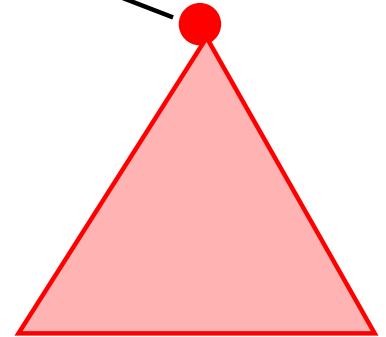


\Rightarrow

$rank[x]$



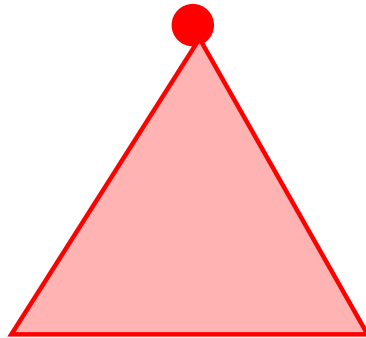
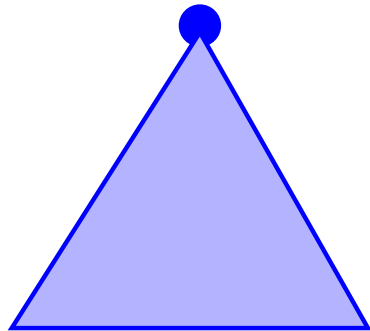
$rank[y]$



$rank[x]$

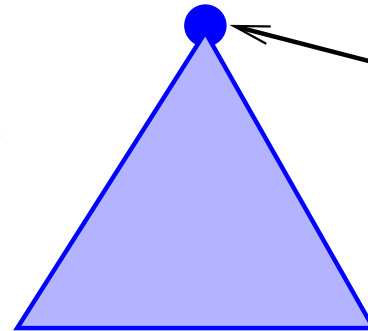
$=$

$rank[y]$

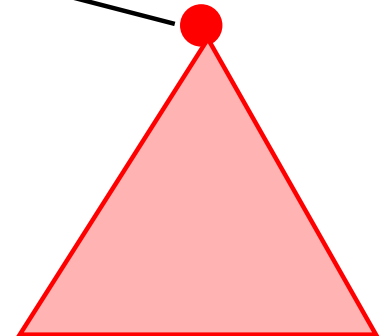


\Rightarrow

$rank[x] + 1$



$rank[y]$



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



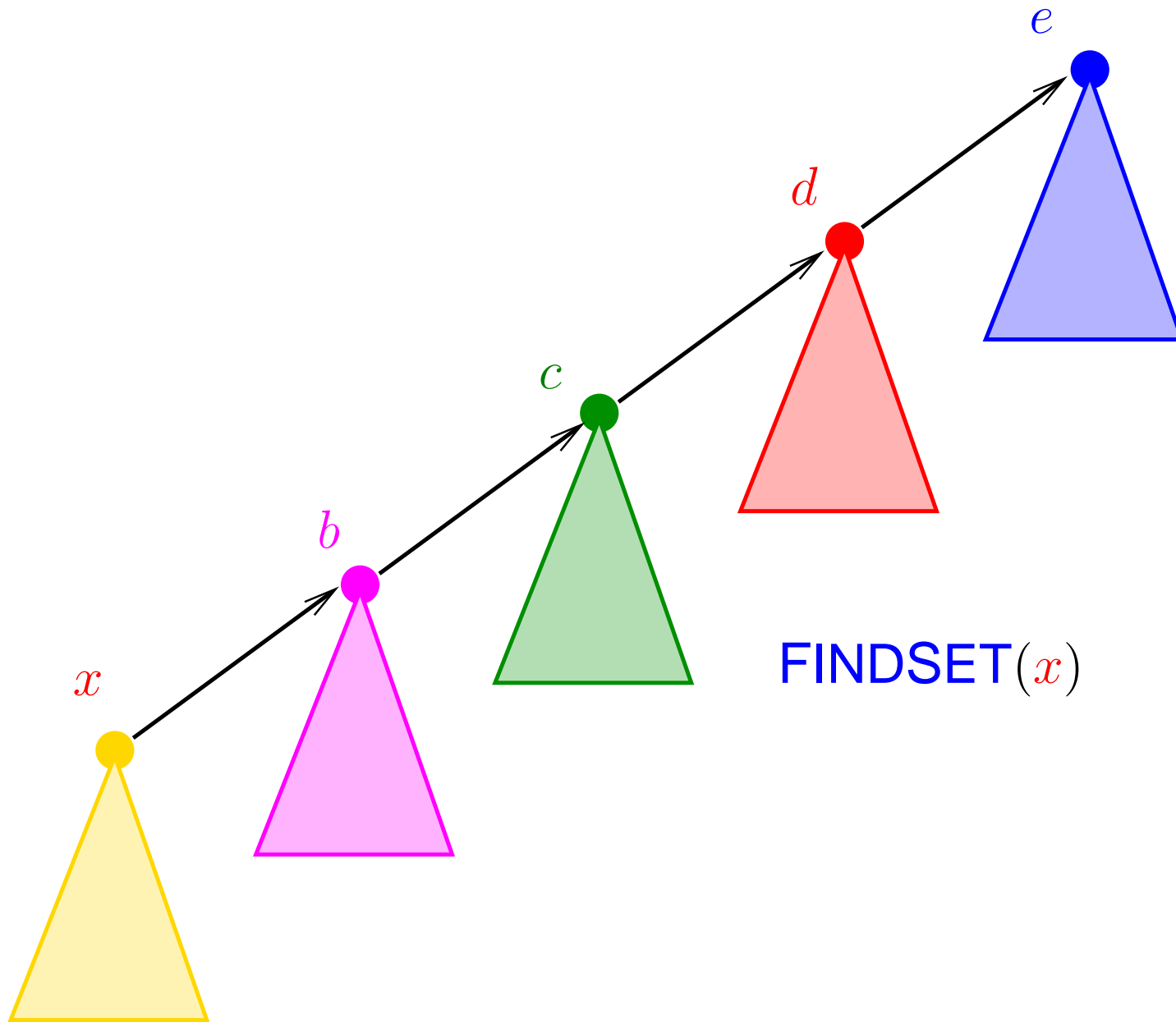
m

$altura(x) \leq \lg n$ para cada nó x

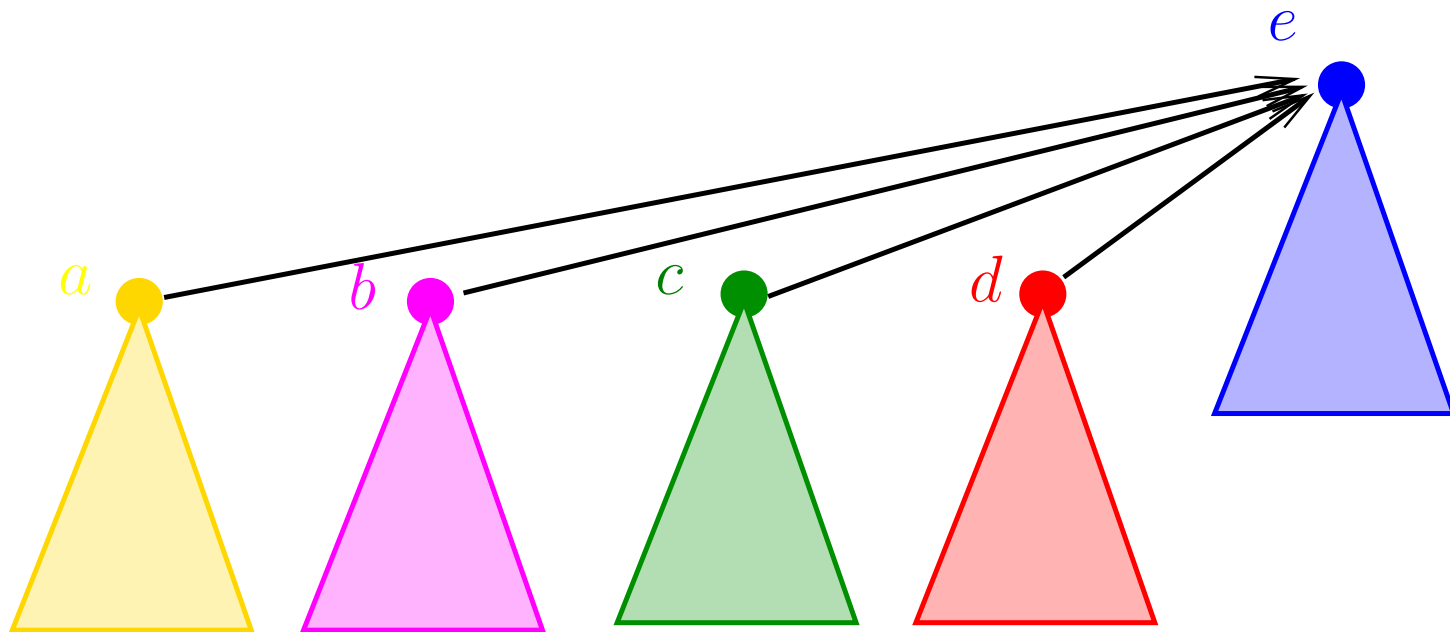
| | | |
|--------------------|---------|-------------|
| Consumos de tempo: | MAKESET | $\Theta(1)$ |
| | UNION | $O(\lg n)$ |
| | FINDSET | $O(\lg n)$ |

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

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