Conjuntos disjuntos dinâmicos

CLRS 21

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 $i \neq j$.

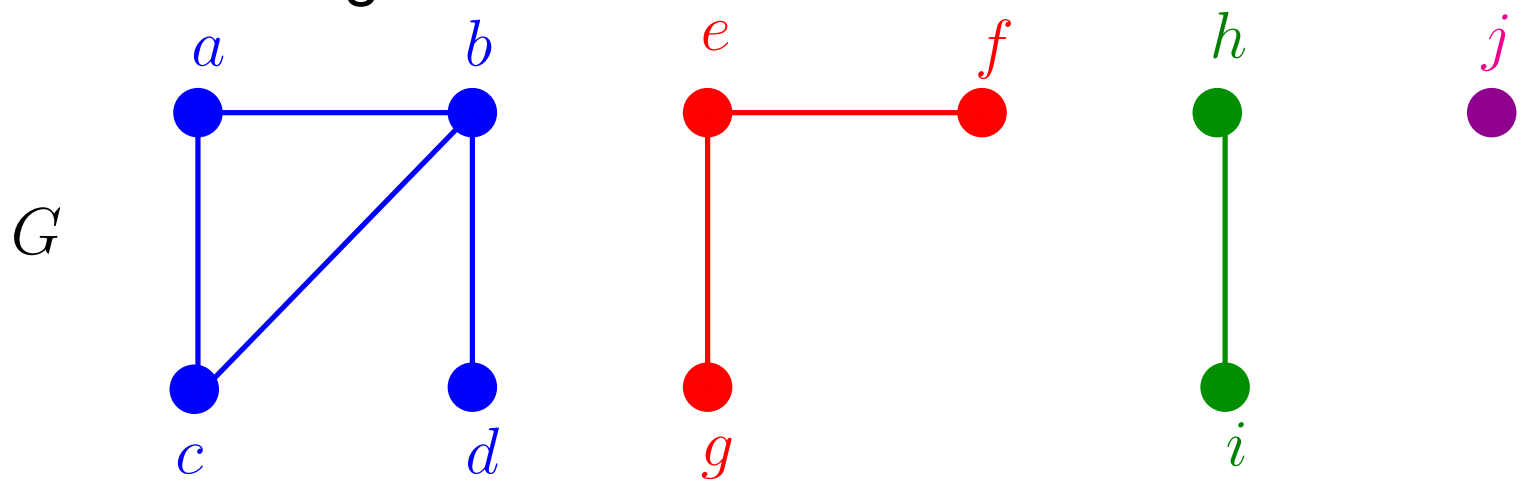
Conjuntos disjuntos

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 $i \neq j$.

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



componentes = conjuntos disjuntos de vértices

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

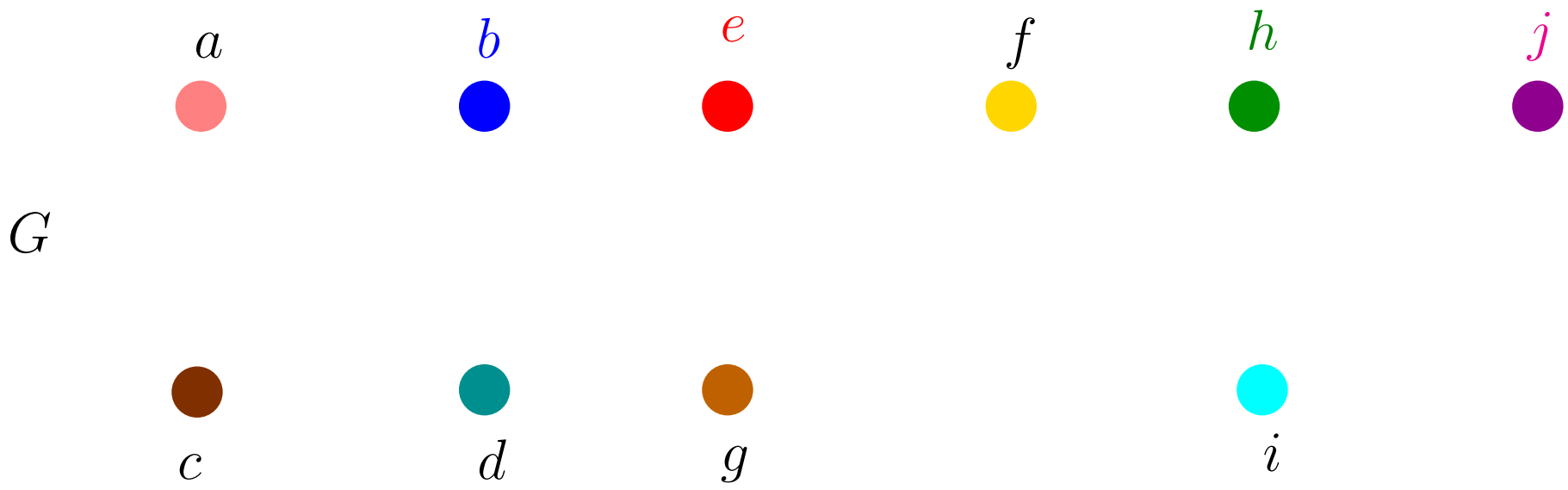
Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

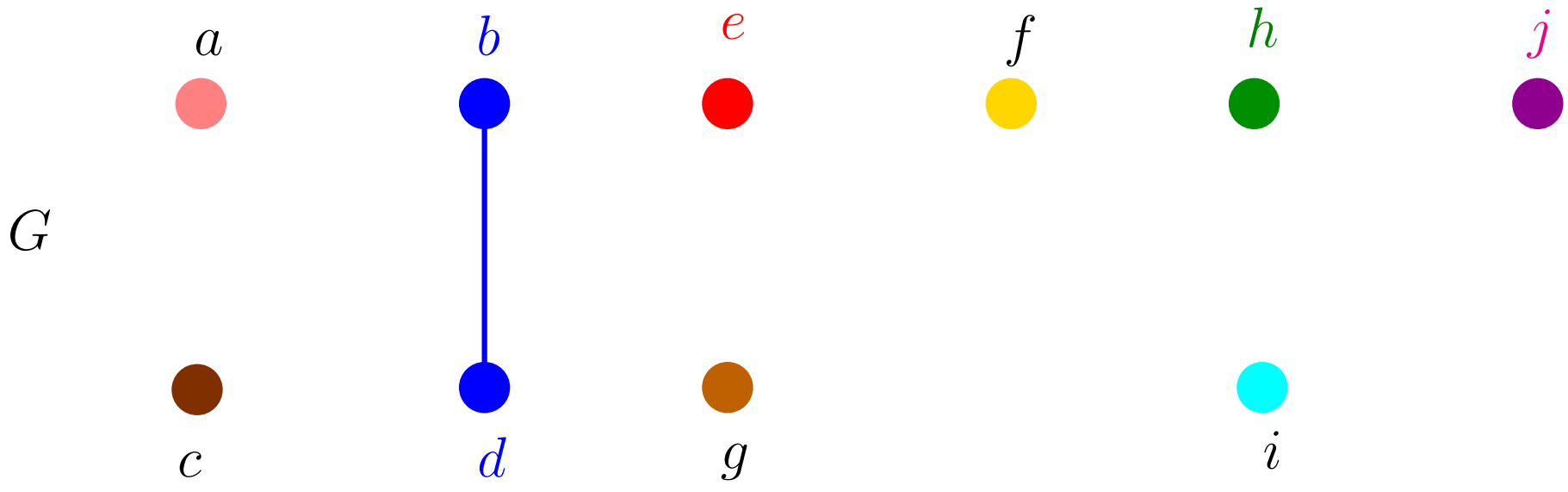


aresta	componentes
	$\{a\}$ $\{b\}$ $\{c\}$ $\{d\}$ $\{e\}$ $\{f\}$ $\{g\}$ $\{h\}$ $\{i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

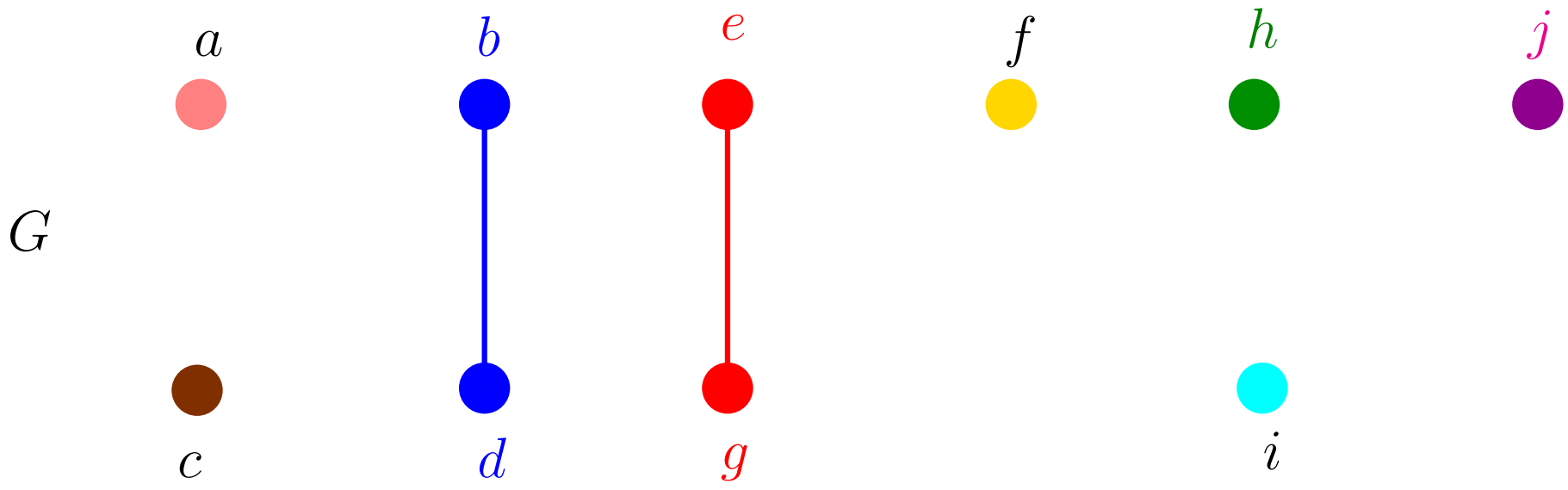


aresta	componentes								
(b, d)	$\{a\}$	$\{b, d\}$	$\{c\}$	$\{e\}$	$\{f\}$	$\{g\}$	$\{h\}$	$\{i\}$	$\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

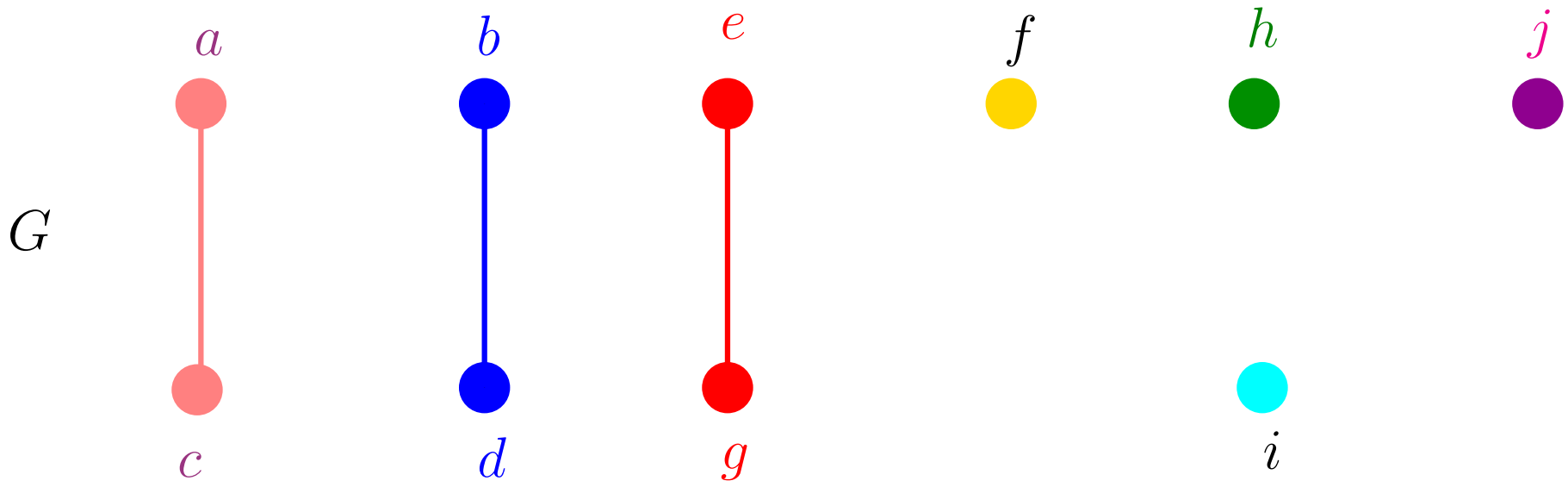


aresta	componentes							
(e, g)	$\{a\}$	$\{b, d\}$	$\{c\}$	$\{e, g\}$	$\{f\}$	$\{h\}$	$\{i\}$	$\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

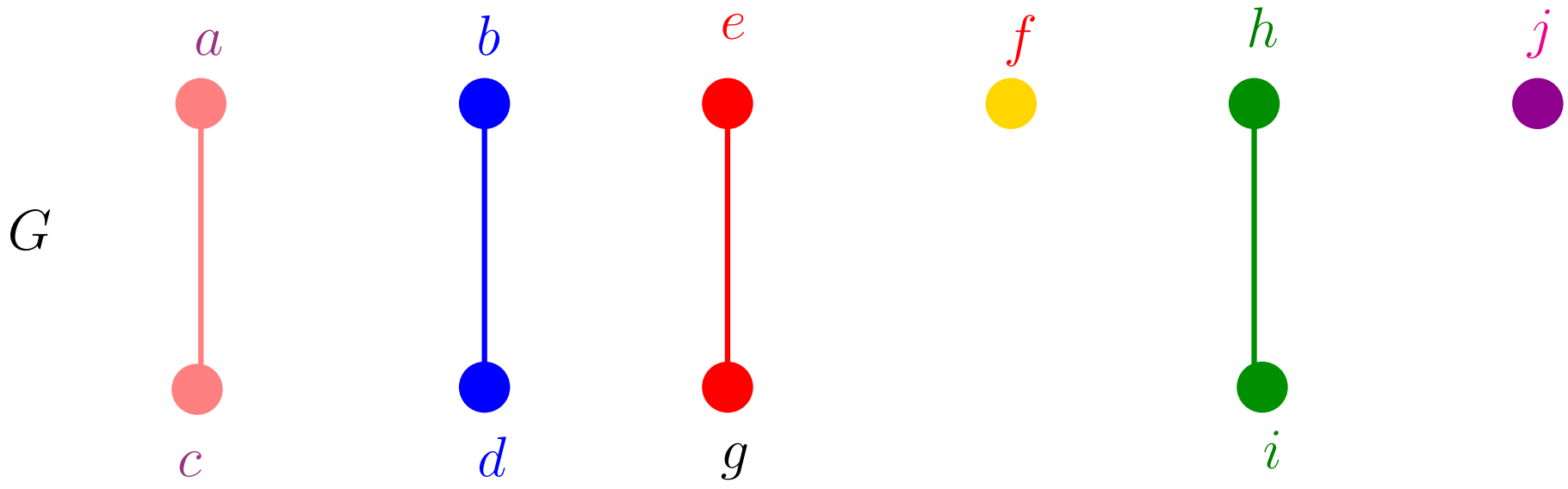


aresta	componentes
(a, c)	$\{a, c\}$ $\{b, d\}$ $\{e, g\}$ $\{f\}$ $\{h\}$ $\{i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

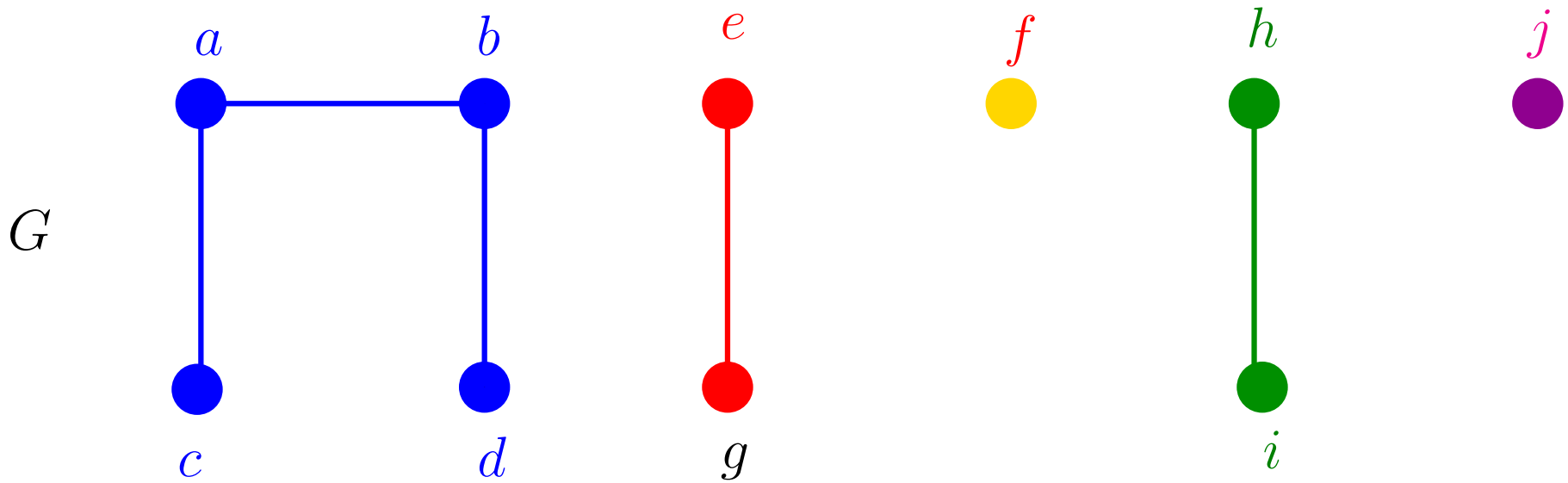


aresta	componentes					
(h, i)	$\{a, c\}$	$\{b, d\}$	$\{e, g\}$	$\{f\}$	$\{h, i\}$	$\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

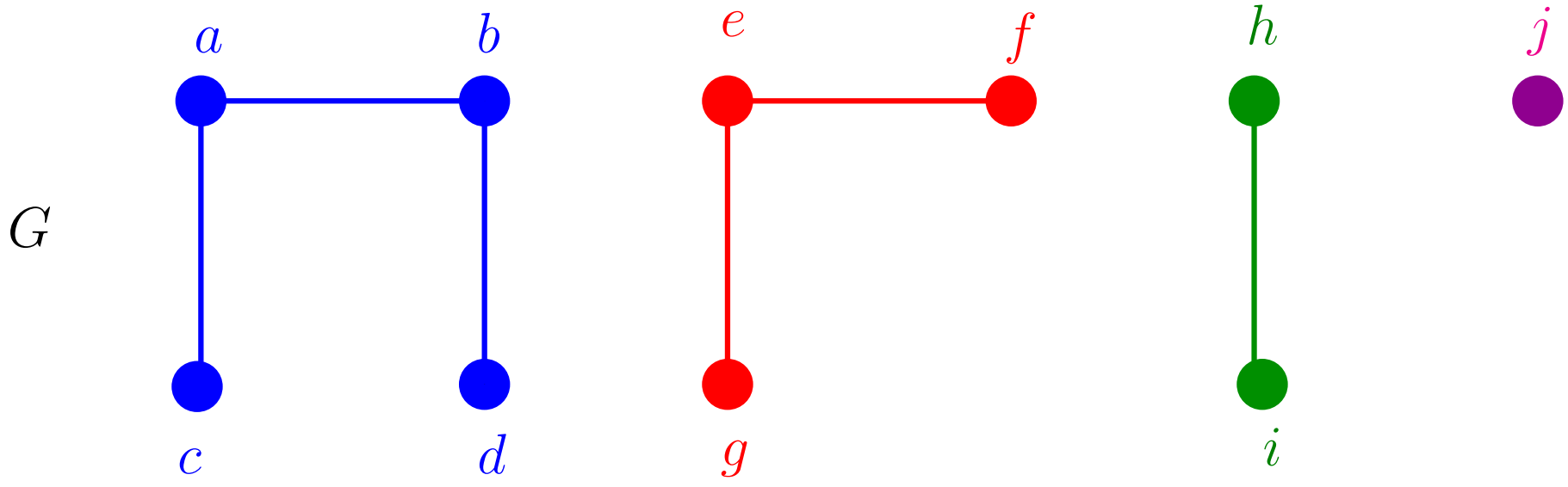


aresta	componentes				
(a, b)	$\{a, b, c, d\}$	$\{e, g\}$	$\{f\}$	$\{h, i\}$	$\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

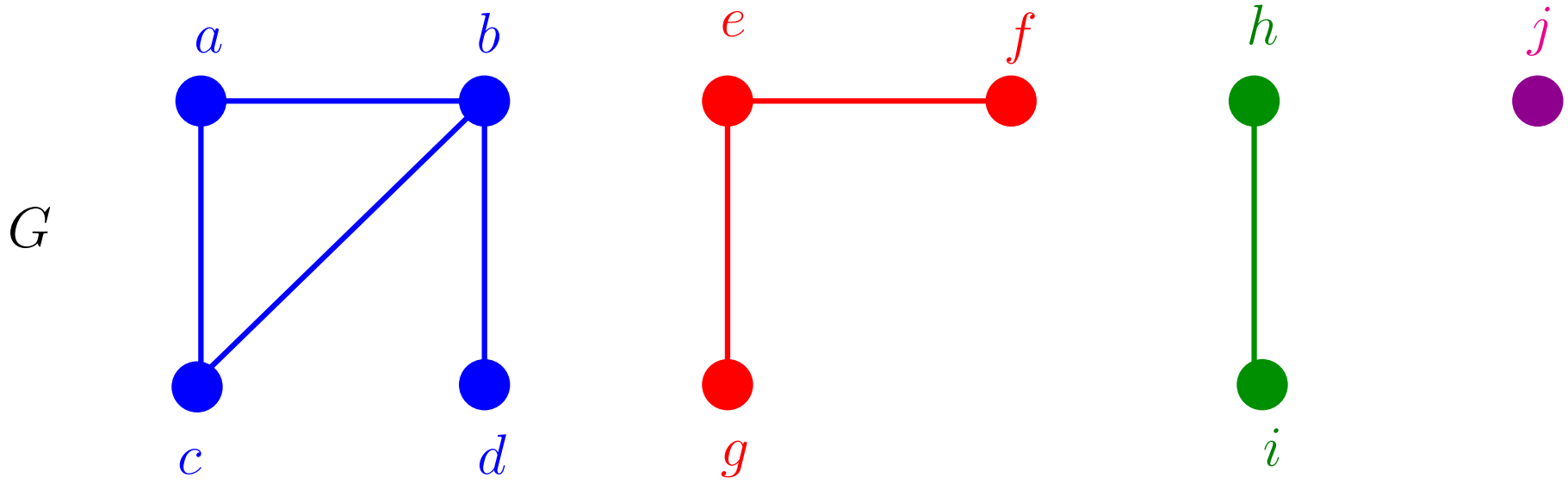


aresta	componentes			
(e, f)	$\{a, b, c, d\}$	$\{e, f, g\}$	$\{h, i\}$	$\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico



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

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 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)
para cada vértice v de G faça
    MAKESET (v)
para cada aresta (u, v) de G faça
    se FINDSET (u) ≠ FINDSET (v)
        então UNION (u, v)
```

Consumo de tempo

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

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) $\quad \triangleright$ G conexo

```
A ← ∅
para cada vértice v faça
    MAKESET (v)
coloque arestas em ordem crescente de w
para cada aresta uv em ordem crescente de w faça
    se FINDSET (u) ≠ FINDSET (v)
        então A ← A ∪ {uv}
                UNION (u, v)
devolva A
```

Conjuntos disjuntos dinâmicos

Seqüência de operações MAKESET, UNION, FINDSET

$$\underbrace{\underbrace{\mathsf{M}\quad \mathsf{M}\quad \mathsf{M}}_{n}\quad \mathsf{U}\quad \mathsf{F}\quad \mathsf{U}\quad \mathsf{U}\quad \mathsf{F}\quad \mathsf{U}\quad \mathsf{F}\quad \mathsf{F}\quad \mathsf{F}\quad \mathsf{U}\quad \mathsf{F}}_{m}$$

Que estrutura de dados usar?

Compromissos (*trade-offs*)

Algumas possibilidades

(a) cada conjunto é representado por uma lista ligada circular

(b) cada conjunto é representado por uma árvore enraizada de altura 1.

Algumas possibilidades

(a) cada conjunto é representado por uma lista ligada circular

(b) cada conjunto é representado por uma árvore enraizada de altura 1.

Consumo de tempo	(a)	(b)
MAKESET	$\Theta(1)$	$\Theta(1)$
UNION/LINK	$\mathrm{O}(n)/O(1)$	$\mathrm{O}(n)$
FINDSET	$\mathrm{O}(n)$	$\Theta(1)$

Algumas possibilidades

(a) cada conjunto é representado por uma lista ligada circular

(b) cada conjunto é representado por uma árvore enraizada de altura 1.

Consumo de tempo	(a)	(b)
MAKESET	$\Theta(1)$	$\Theta(1)$
UNION/LINK	$\mathrm{O}(n)/O(1)$	$\mathrm{O}(n)$
FINDSET	$\mathrm{O}(n)$	$\Theta(1)$

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

Consumo de tempo amortizado por operação é $\mathrm{O}(m)$.

Melhoramento: *weighted-union*

Na segunda opção, armazene adicionalmente o tamanho de cada conjunto.

Melhoramento: *weighted-union*

Na segunda opção, armazene adicionalmente
o tamanho de cada conjunto.

Na união, altera-se os apontadores para
o representante de cada elemento do conjunto menor.

Melhoramento: *weighted-union*

Na segunda opção, armazene adicionalmente
o tamanho de cada conjunto.

Na união, altera-se os apontadores para
o representante de cada elemento do conjunto menor.

O custo de pior caso de cada operação não se altera,
mas o custo amortizado melhora.

Melhoramento: *weighted-union*

Na segunda opção, armazene adicionalmente
o tamanho de cada conjunto.

Na união, altera-se os apontadores para
o representante de cada elemento do conjunto menor.

O custo de pior caso de cada operação não se altera,
mas o custo amortizado melhora.

Como fazer esta análise?

Melhoramento: *weighted-union*

Na segunda opção, armazene adicionalmente
o tamanho de cada conjunto.

Na união, altera-se os apontadores para
o representante de cada elemento do conjunto menor.

O custo de pior caso de cada operação não se altera,
mas o custo amortizado melhora.

O apontador para o representante de cada elemento x
pode ser alterado no máximo $\lg n$ vezes, pois o tamanho do
conjunto que contém x dobra a cada alteração.

Melhoramento: *weighted-union*

Na segunda opção, armazene adicionalmente o tamanho de cada conjunto.

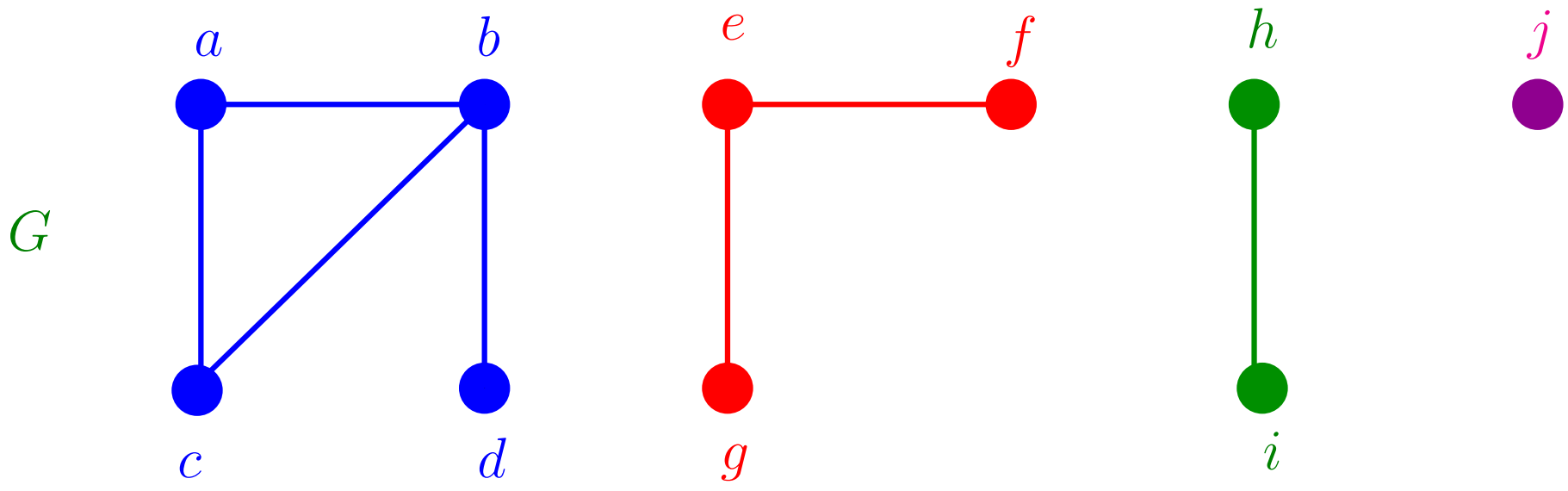
Na união, altera-se os apontadores para o representante de cada elemento do conjunto menor.

O custo de pior caso de cada operação não se altera, mas o custo amortizado melhora.

O apontador para o representante de cada elemento x pode ser alterado no máximo $\lg n$ vezes, pois o tamanho do conjunto que contém x dobra a cada alteração.

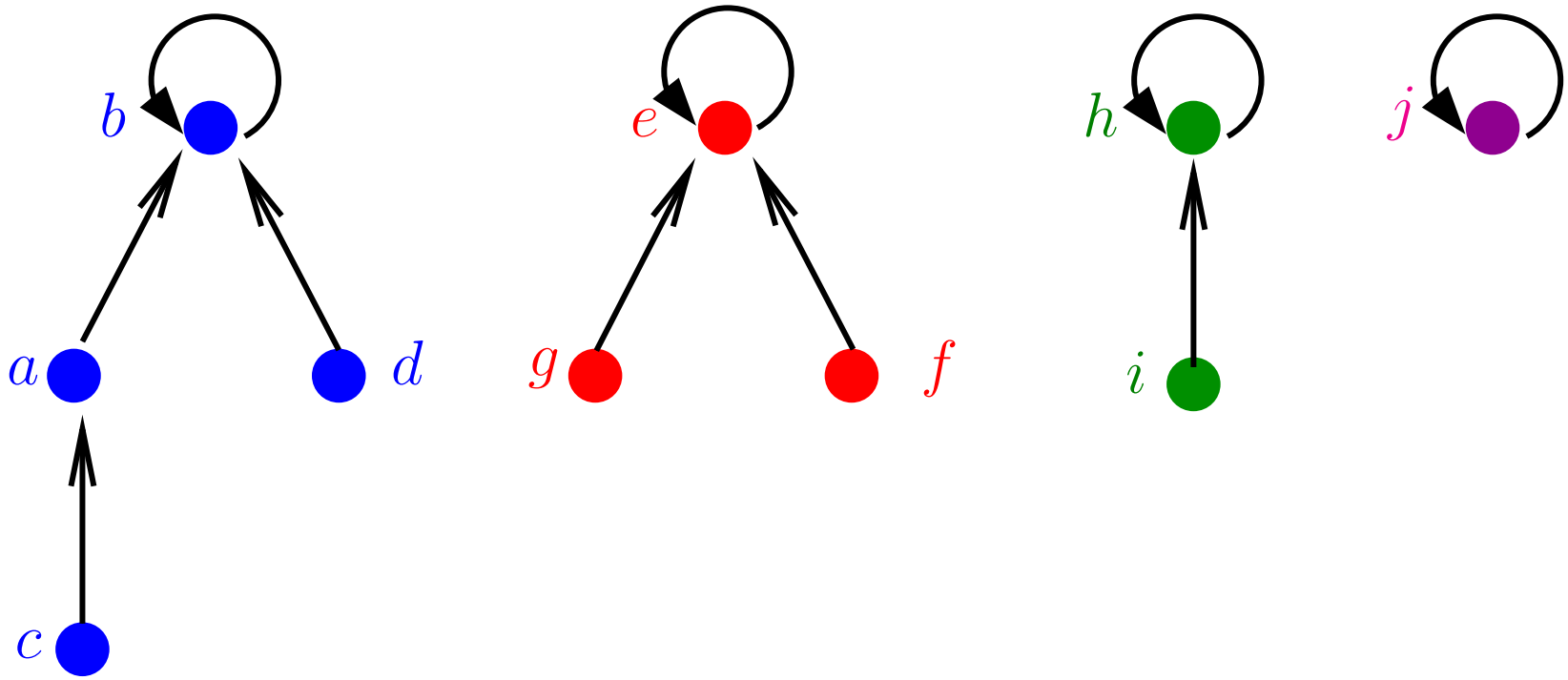
Conclusão: uma seqüência de m operações MAKESET, UNION e FINDSET, sendo que n são MAKESET, consome tempo $\mathrm{O}(m + n \lg n)$.

Estrutura *disjoint-set forest*



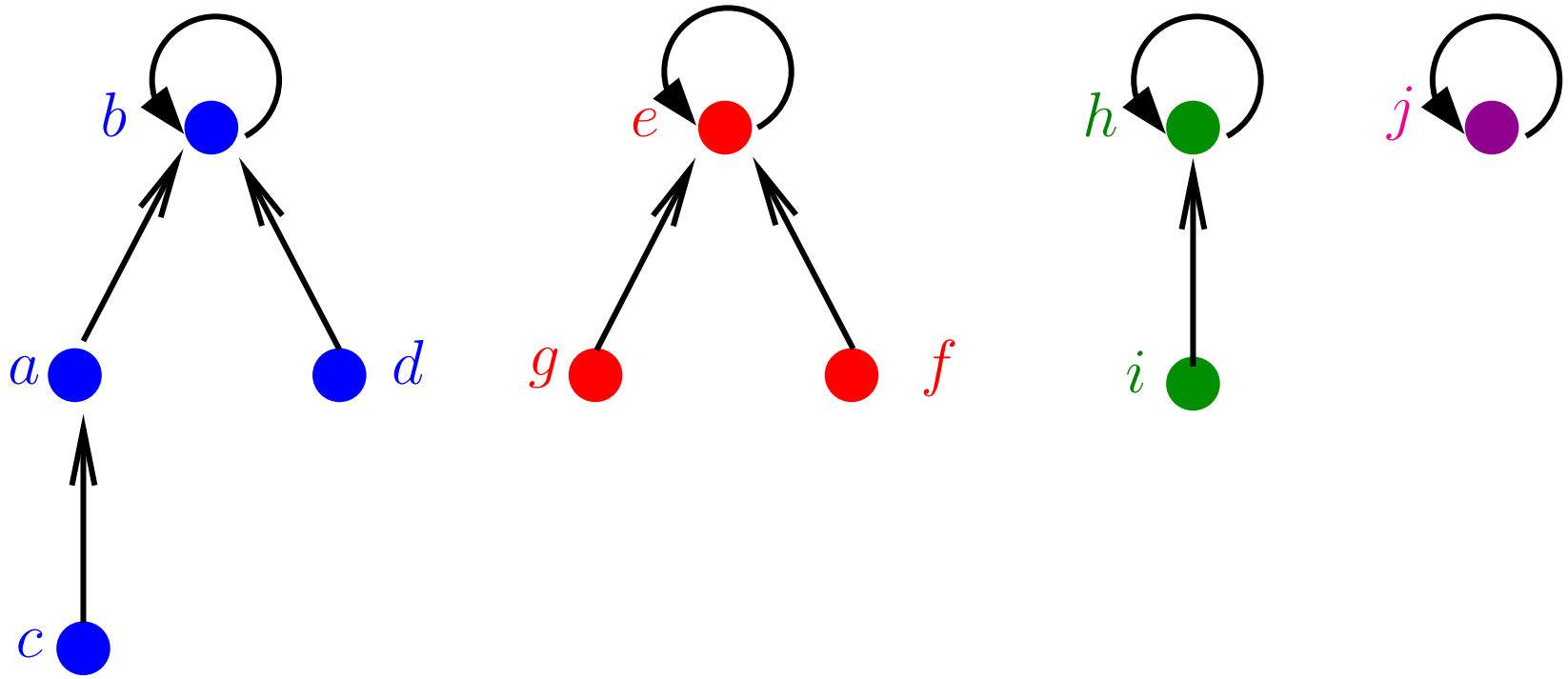
- cada conjunto tem uma $raiz$, que é o seu representate
- cada nó x tem um pai
- $pai[x] = x$ se e só se x é uma raiz

Estrutura *disjoint-set forest*



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

MakeSet$_0$ e FindSet$_0$



MAKESET$_0$ (x)

1 $\quad pai[x] \leftarrow x$

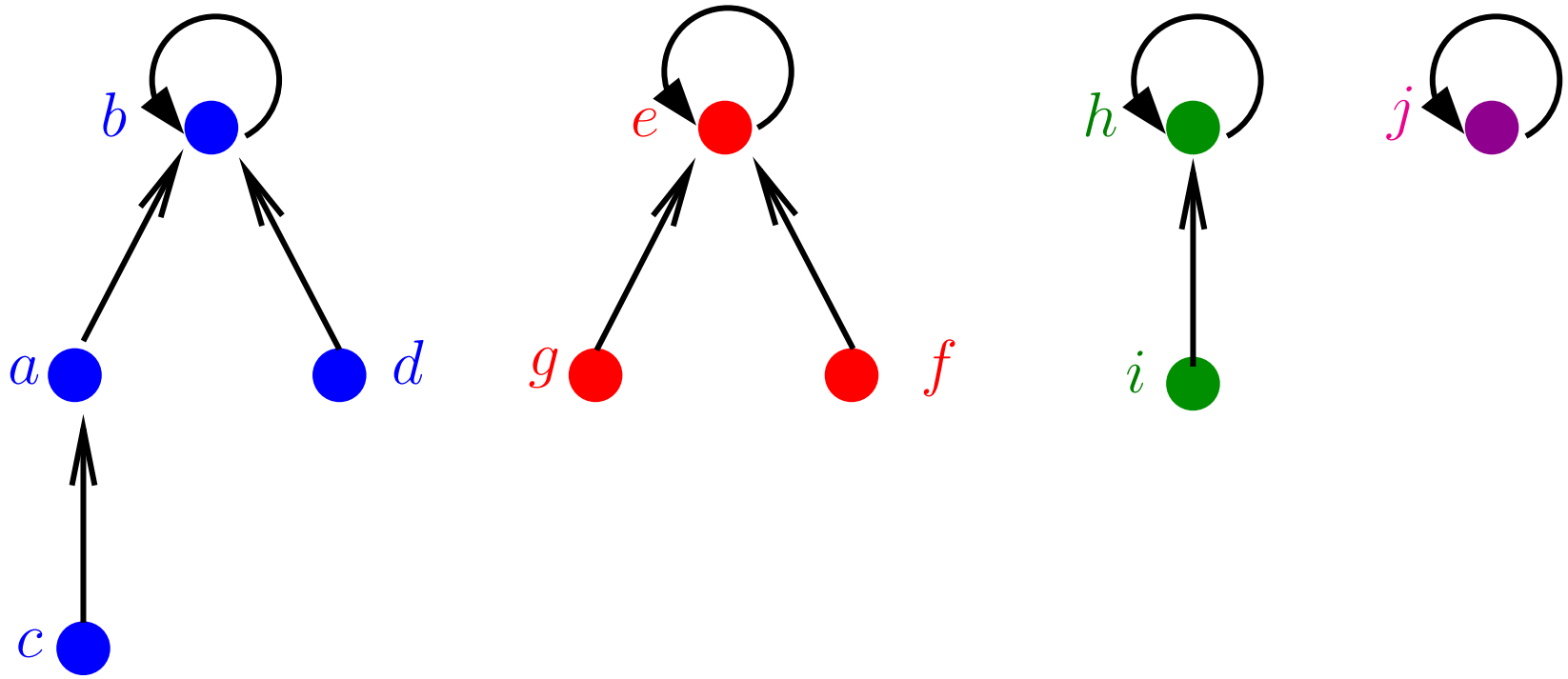
FINDSET$_0$ (x)

1 **enquanto** $pai[x] \neq x$ **faça**

2 $\quad\quad x \leftarrow pai[x]$

3 **devolva** x

FindSet$_1$



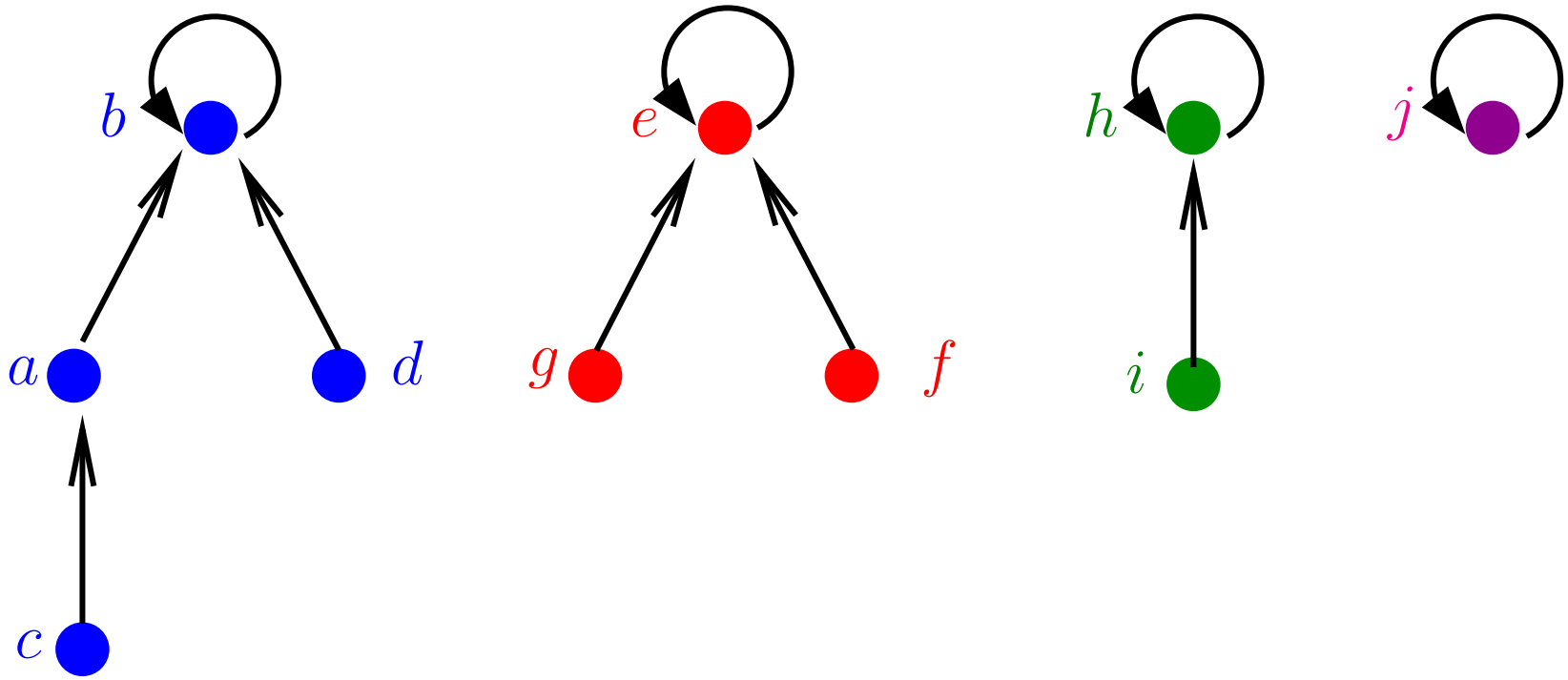
FINDSET$_1$ (x)

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

2 **então devolva** x

3 **senão devolva** FINDSET$_1$ $(pai[x])$

Union$_0$



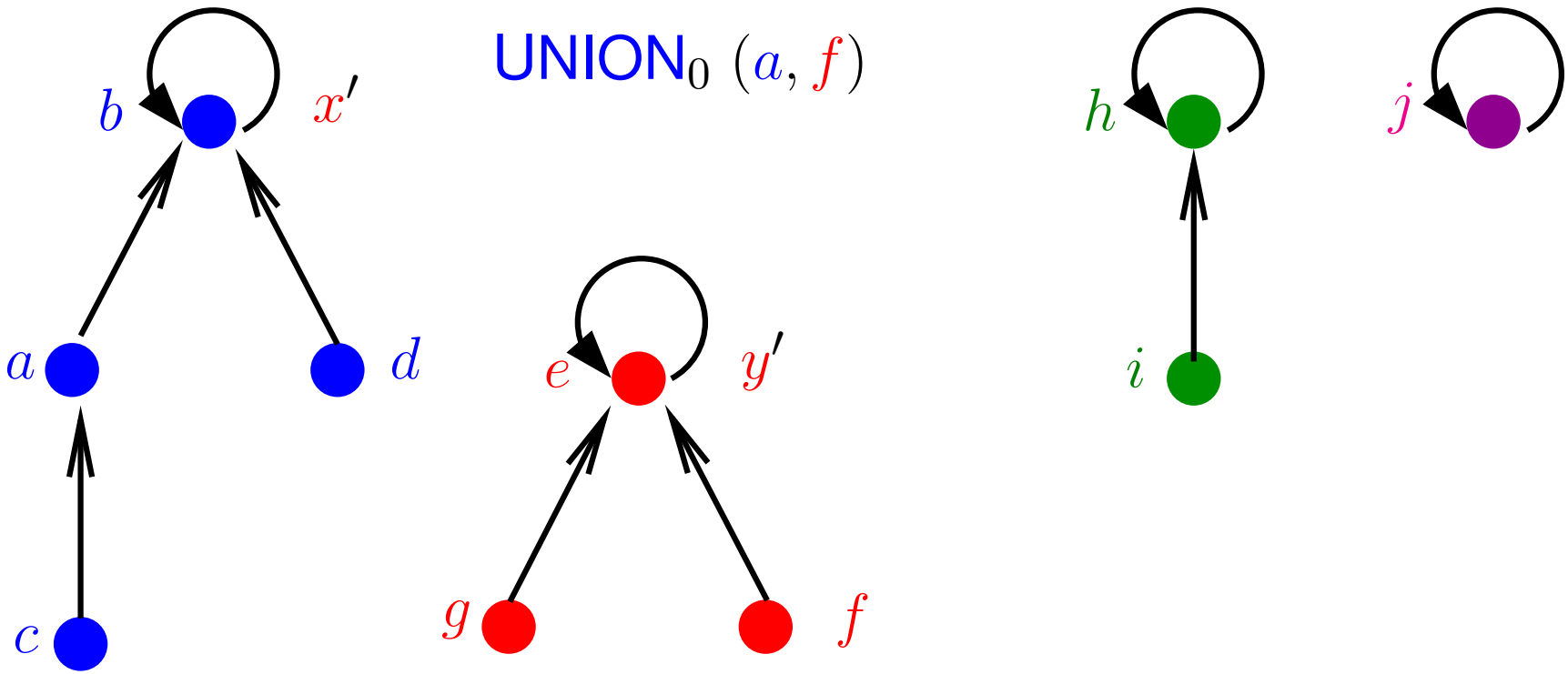
UNION$_0$ (x, y)

1 $x' \leftarrow$ FINDSET$_0$ (x)

2 $y' \leftarrow$ FINDSET$_0$ (y)

3 $pai[y'] \leftarrow x'$

Union$_0$



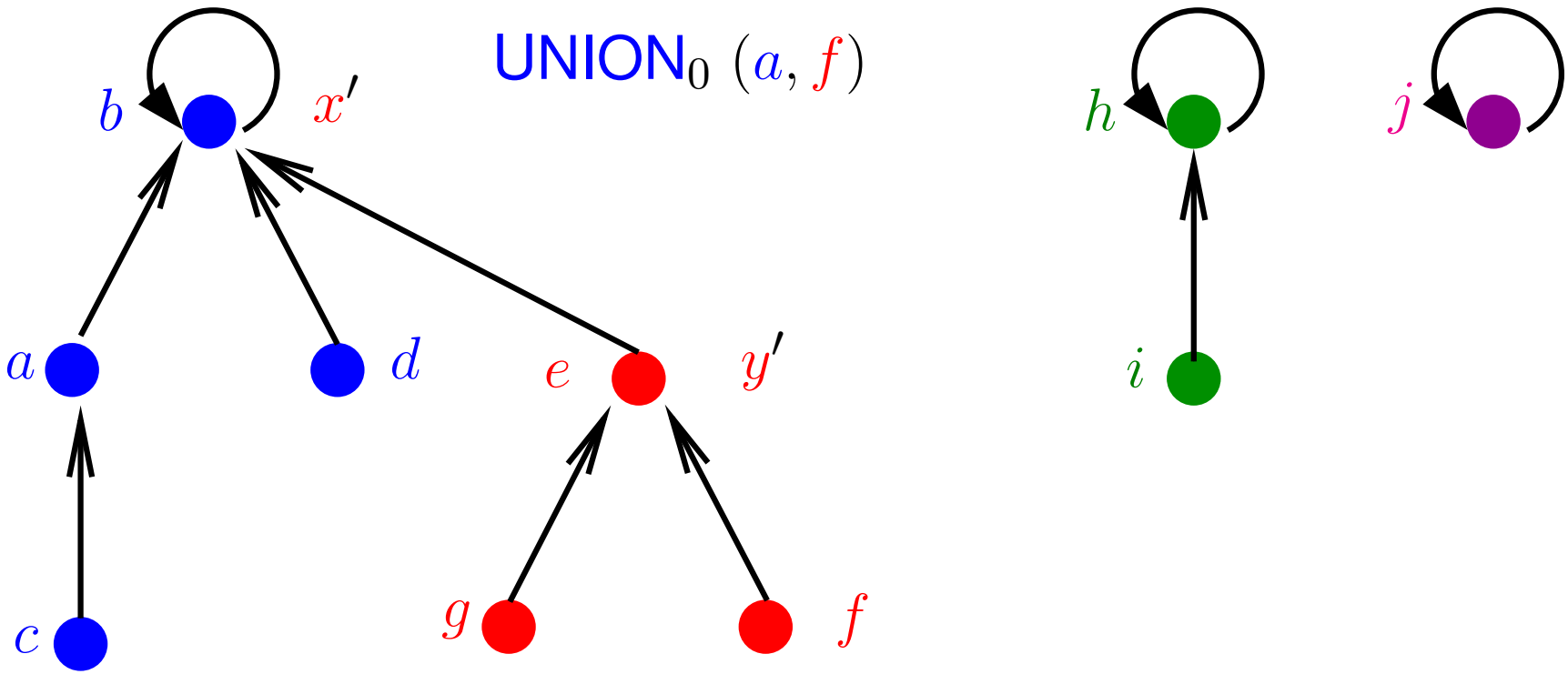
UNION$_0$ (x, y)

1 $x' \leftarrow$ FINDSET$_0$ (x)

2 $y' \leftarrow$ FINDSET$_0$ (y)

3 $pai[y'] \leftarrow x'$

Union$_0$



UNION$_0$ (x, y)

1 $x' \leftarrow$ FINDSET$_0$ (x)

2 $y' \leftarrow$ FINDSET$_0$ (y)

3 $pai[y'] \leftarrow x'$

MakeSet$_0$, Union$_0$ e FindSet$_1$

MAKESET$_0$ (x)

1 $pai[x] \leftarrow x$

UNION$_0$ (x, y)

1 $x' \leftarrow$ FINDSET$_0$ (x)

2 $y' \leftarrow$ FINDSET$_0$ (y)

3 $pai[y'] \leftarrow x'$

FINDSET$_1$ (x)

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

2 **então devolva** x

3 **senão devolva** FINDSET$_1$ $(pai[x])$

Consumo de tempo

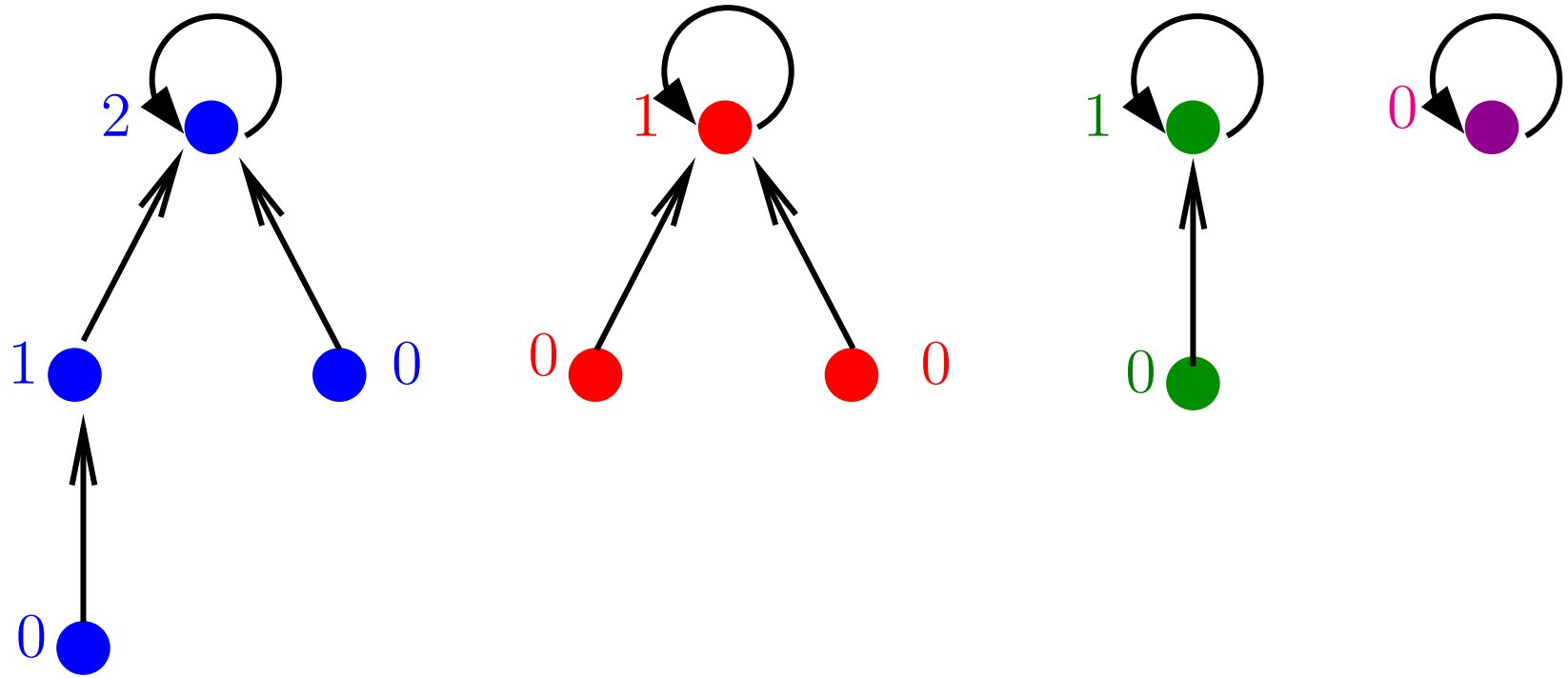
MAKESET$_0$	$\Theta(1)$
UNION$_0$	$\mathrm{O}(n)$
FINDSET$_0$	$\mathrm{O}(n)$

$$\underbrace{\underbrace{\mathsf{M}\quad \mathsf{M}\quad \mathsf{M}}_{n}\quad \mathsf{U}\quad \mathsf{F}\quad \mathsf{U}\quad \mathsf{U}\quad \mathsf{F}\quad \mathsf{U}\quad \mathsf{F}\quad \mathsf{F}\quad \mathsf{F}\quad \mathsf{U}\quad \mathsf{F}}_{m}$$

Custo total da seqüência:

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

Melhoramento 1: *union by rank*



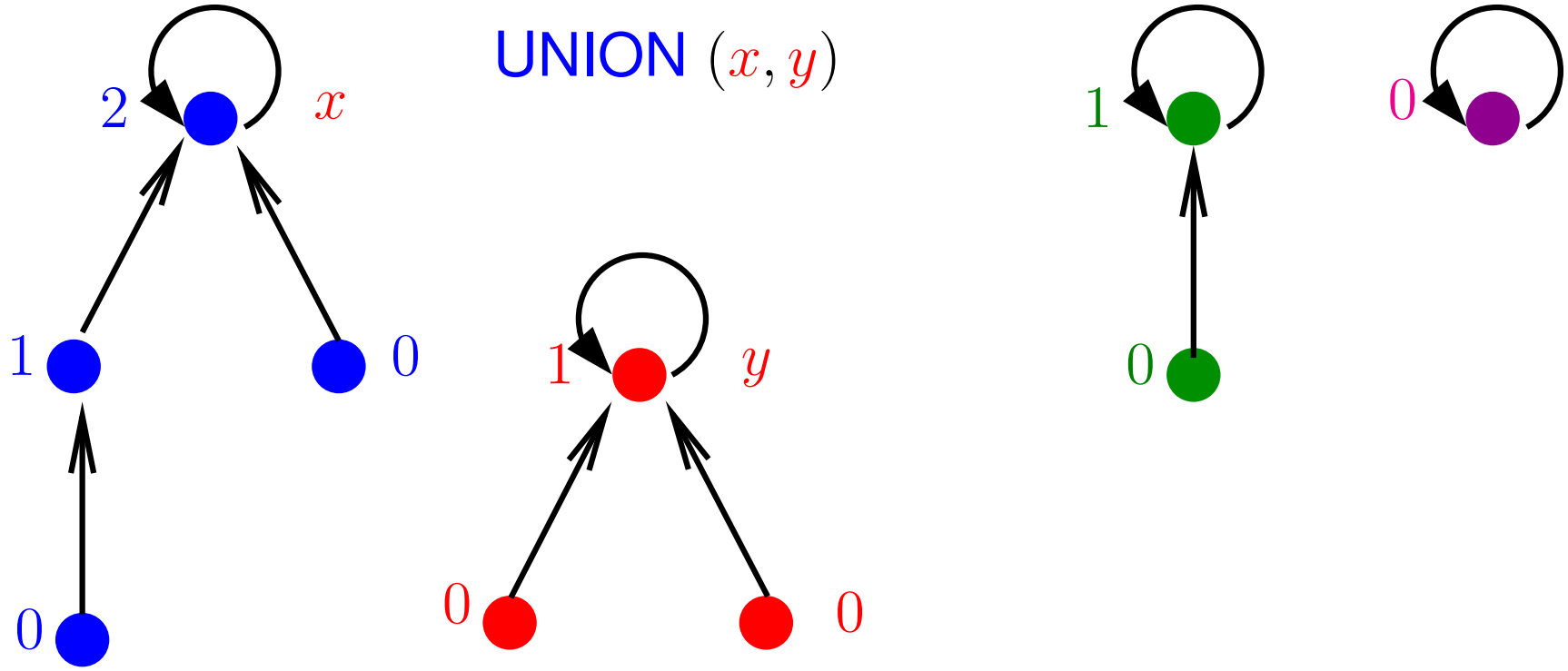
$rank[x]$ = posto do nó x

MAKESET (x)

1 $pai[x] \leftarrow x$

2 $rank[x] \leftarrow 0$

Melhoramento 1: *union by rank*



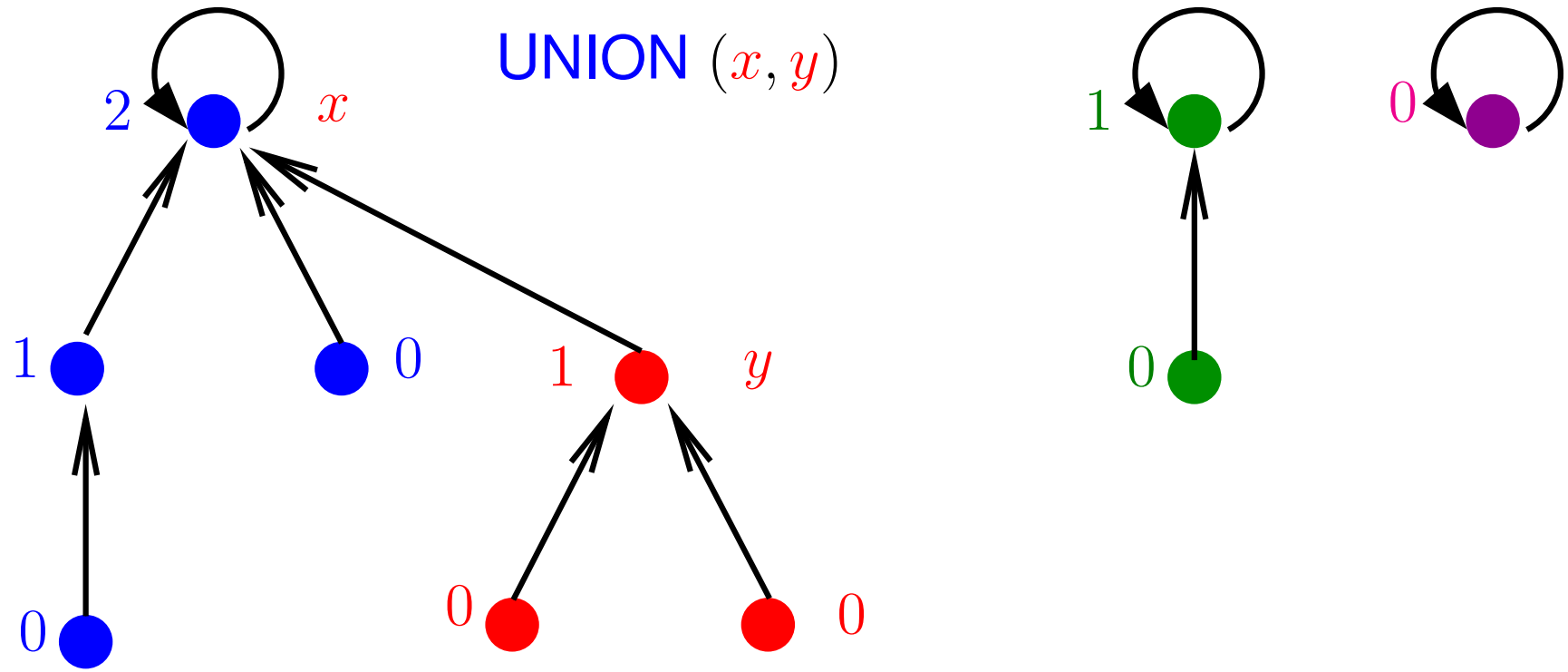
$rank[x]$ = posto do nó x

MAKESET (x)

1 $pai[x] \leftarrow x$

2 $rank[x] \leftarrow 0$

Melhoramento 1: *union by rank*



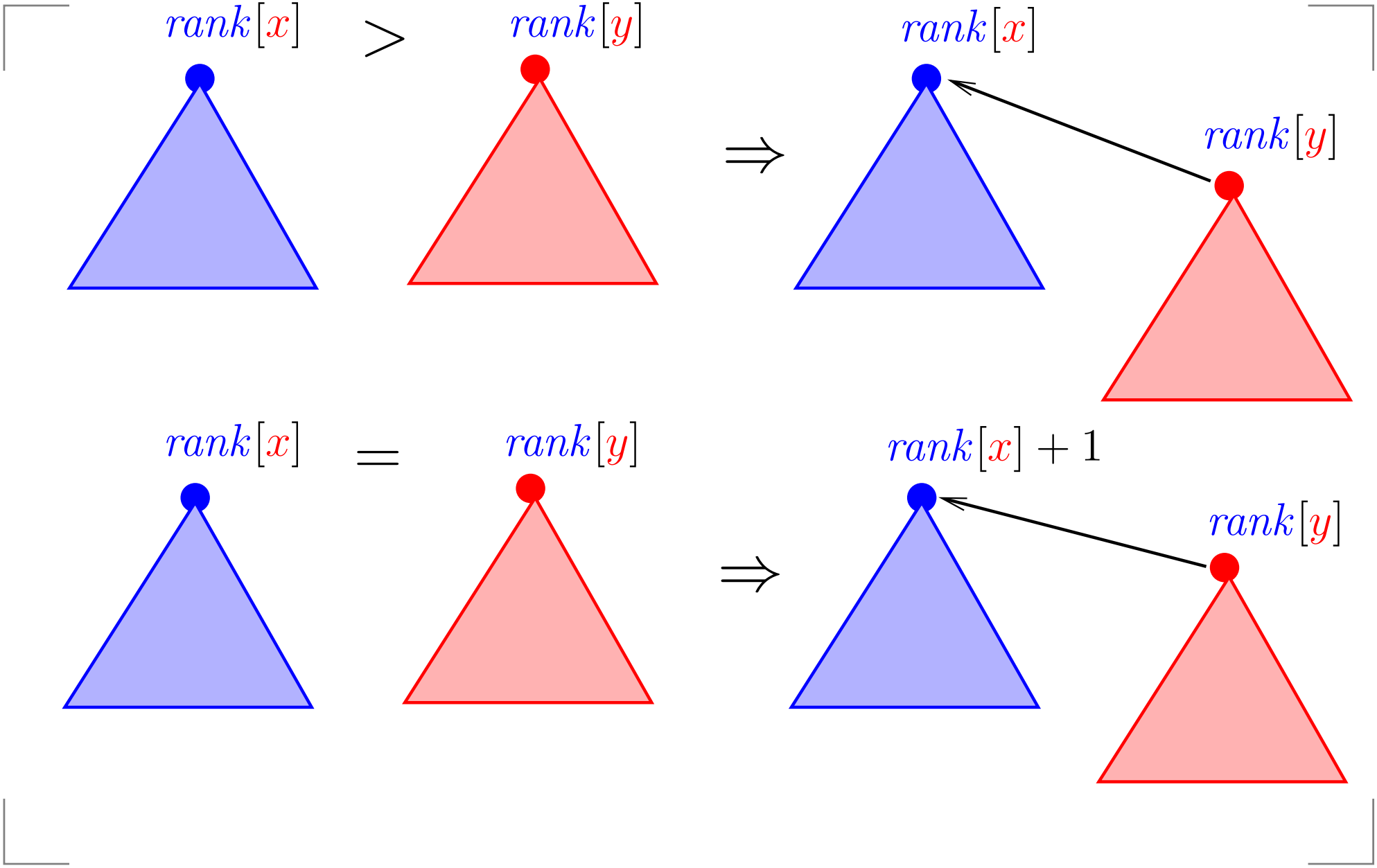
$rank[x]$ = posto do nó x

MAKESET (x)

1 $pai[x] \leftarrow x$

2 $rank[x] \leftarrow 0$

Melhoramento 1: *union by rank*



Melhoramento 1: *union by rank*

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

$x' \leftarrow$ FINDSET (x)

$y' \leftarrow$ FINDSET (y) $\triangleright$ supõe que $x' \neq y'$

se $rank[x'] > rank[y']$

  então $pai[y'] \leftarrow x'$

  senão $pai[x'] \leftarrow y'$

    se $rank[x'] = rank[y']$

      então $rank[y'] \leftarrow rank[y'] + 1$

Melhoramento 1: estrutura

- $rank[x] < rank[pai[x]]$ 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

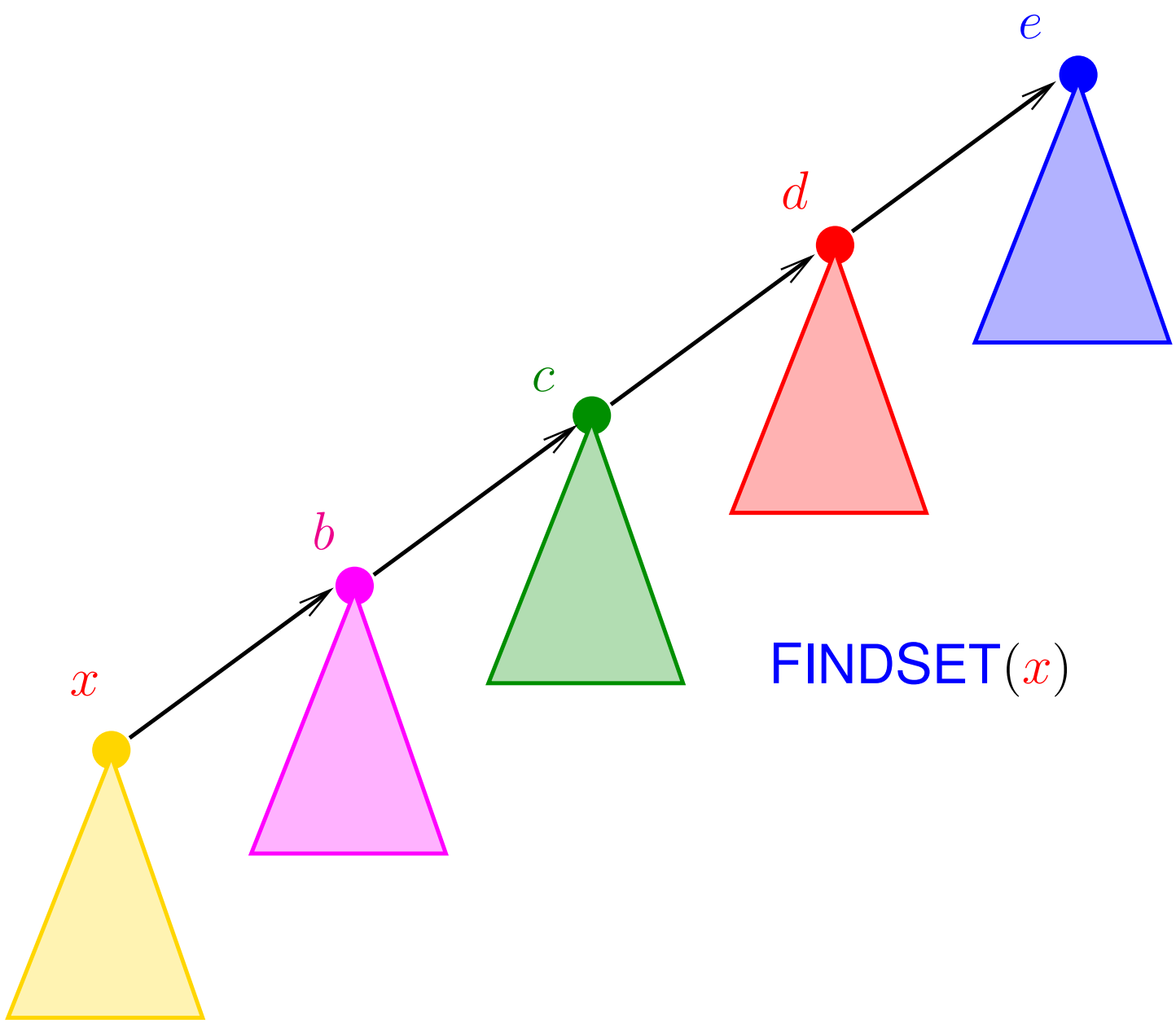
$$\underbrace{\underbrace{\text{M}\quad\text{M}\quad\text{M}}_{n}\quad\text{U}\quad\text{F}\quad\text{U}\quad\text{U}\quad\text{F}\quad\text{U}\quad\text{F}\quad\text{F}\quad\text{F}\quad\text{U}\quad\text{F}}_{m}$$

$altura(x) \leq \lg n$ para cada nó x

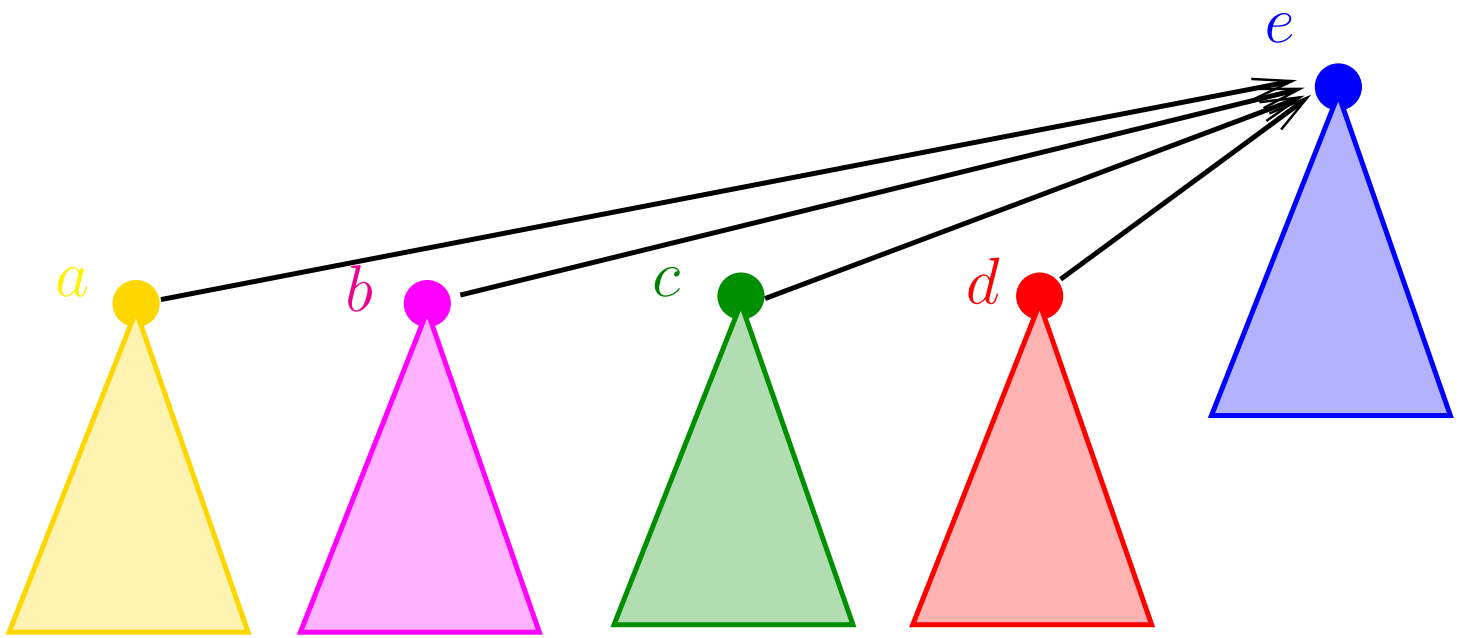
Consumos de tempo:	MAKESET	$\Theta(1)$
	UNION	$\mathrm{O}(\lg n)$
	FINDSET	$\mathrm{O}(\lg n)$

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

Melhoramento 2: *path compression*



Melhoramento 2: *path compression*



FINDSET(x)

Melhoramento 2: *path compression*

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

- $rank[x] < rank[pai[x]]$ 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 $\leq n/2^k$
- $altura(x) \leq rank[x] \leq \lg n$ para cada nó x

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
$\vdots$	$\vdots$
15	3
16	3
$\vdots$	$\vdots$
65535	4
65536	4
$\vdots$	$\vdots$
$\underbrace{100000000000000000 \cdots 000000000000}_{80}$	5
$\vdots$	$\vdots$

Tempo amortizado

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 $\mathrm{O}(m \lg^* n)$.

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 operacões:

```
para i ← 1 até 16
    faça MAKESET (x_i)
para i ← 1 até 15 em passos de 2
    faça UNION (x_i, x_{i+1})
para i ← 1 até 13 em passos de 4
    faça UNION (x_i, x_{i+2})
UNION (x_1, x_5)
UNION (x_11, x_13)
UNION (x_1, x_10)
FINDSET (x_2)
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$_0$, 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$_0$.) Mostre que essa simplificação consome tempo

$$\mathrm{O}(m \lg n) \ .$$

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.)