Melhores momentos

AULA PASSADA

Conjuntos disjuntos dinâmicos

CLR 22   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 $i \neq j$.

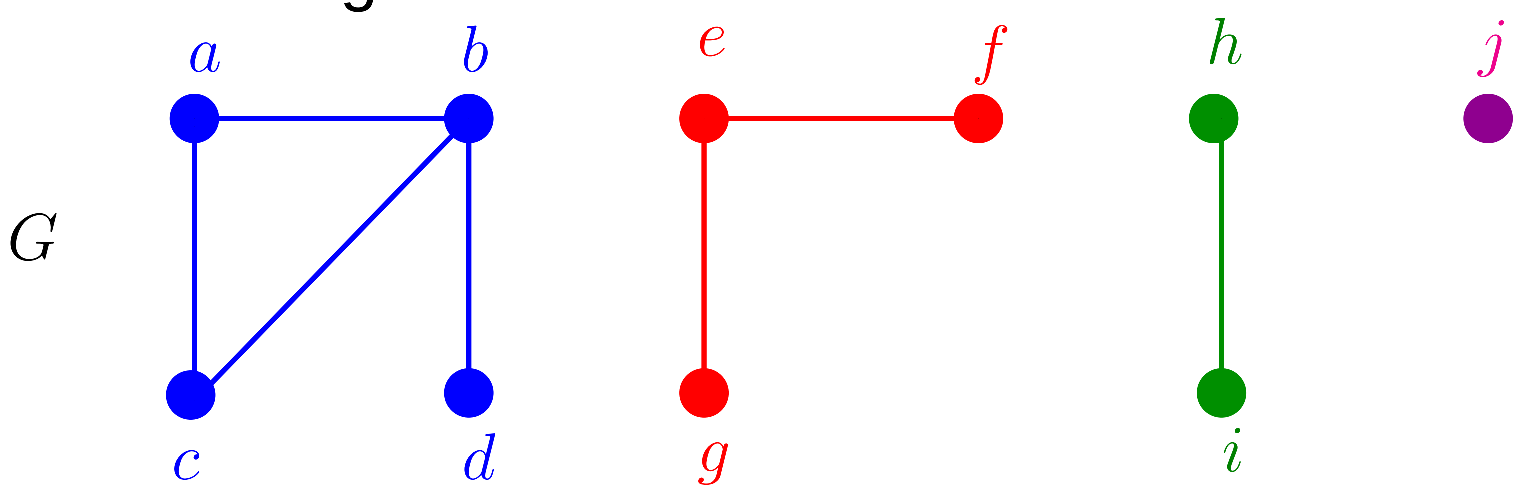
Conjuntos disjuntos

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

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

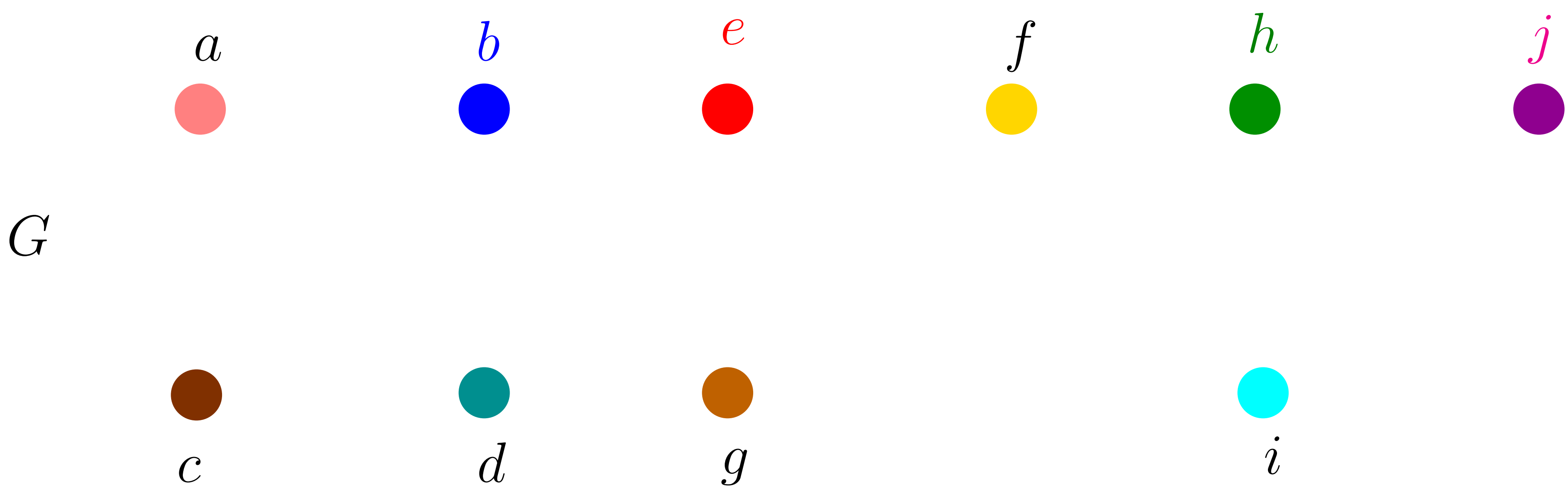
Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

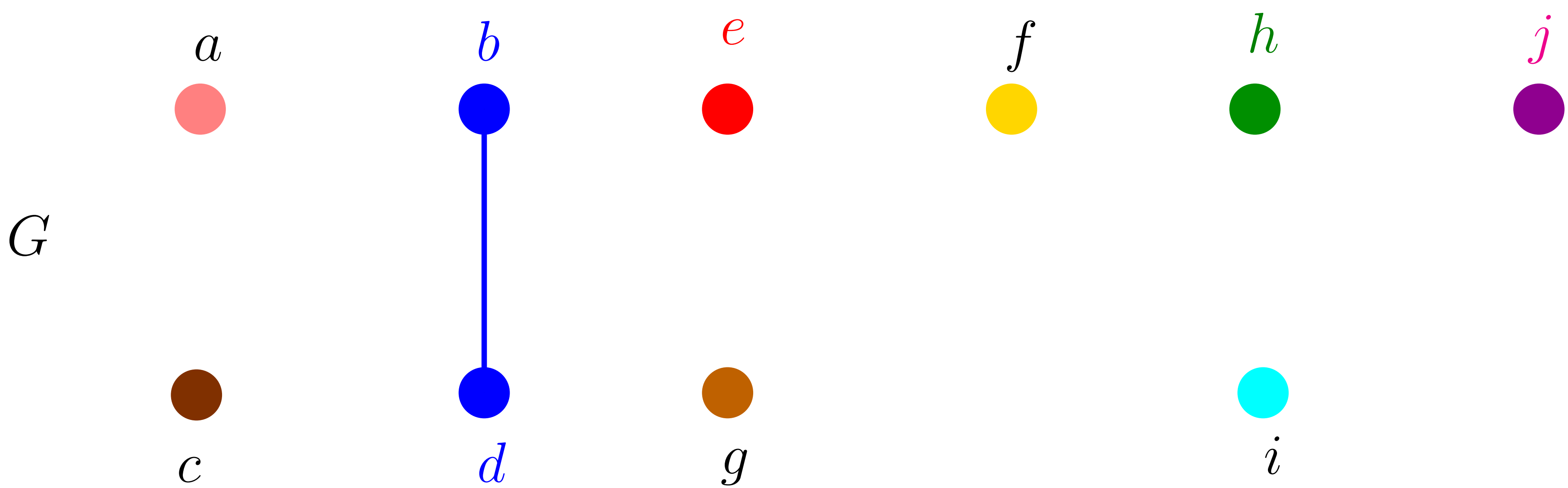


aresta	componentes
	$\{a\}$ $\{b\}$ $\{c\}$ $\{d\}$ $\{e\}$ $\{f\}$ $\{g\}$ $\{h\}$ $\{i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

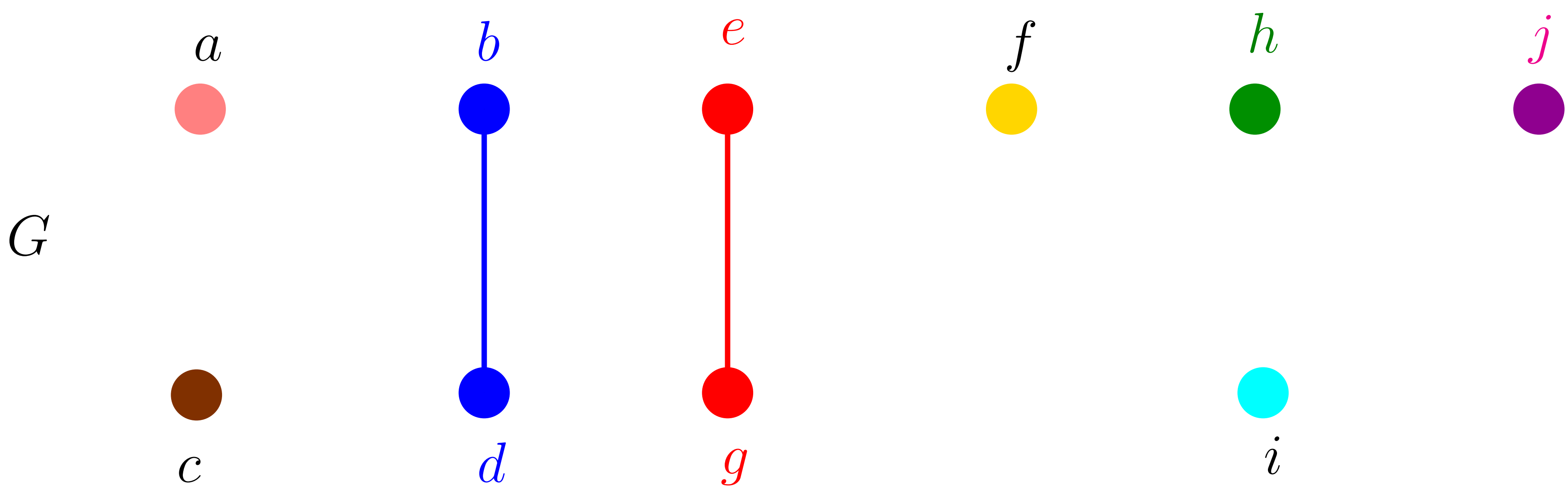


aresta	componentes
(b, d)	$\{a\}$ $\{b, d\}$ $\{c\}$ $\{e\}$ $\{f\}$ $\{g\}$ $\{h\}$ $\{i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

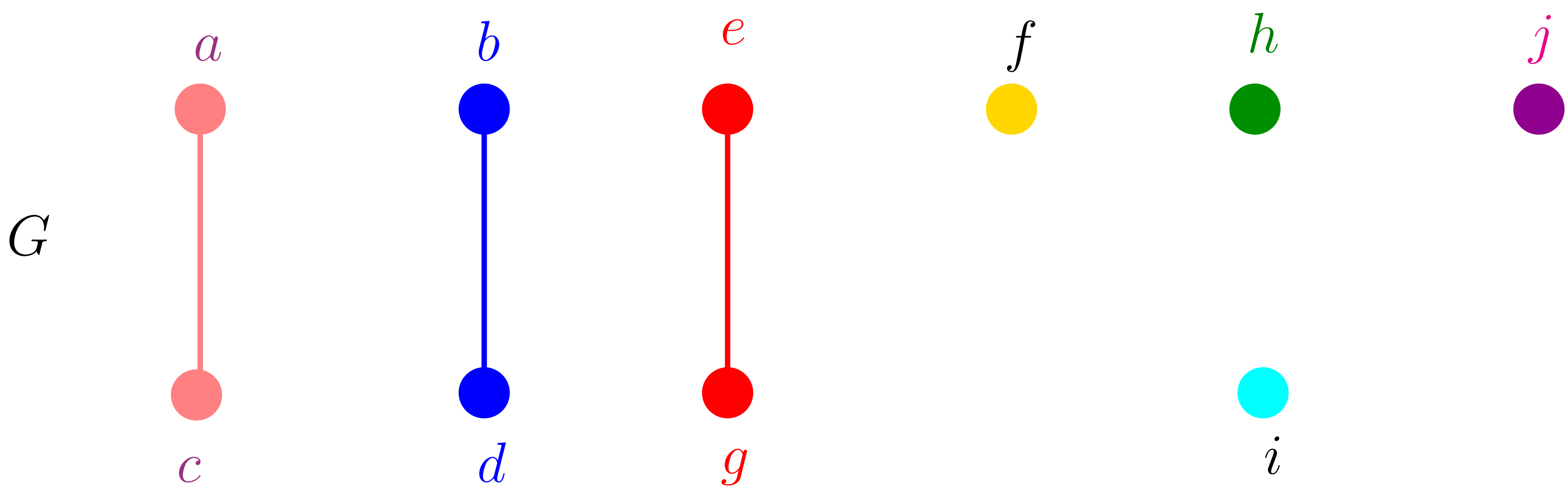


aresta	componentes
(e, g)	$\{a\}$ $\{b, d\}$ $\{c\}$ $\{e, g\}$ $\{f\}$ $\{h\}$ $\{i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

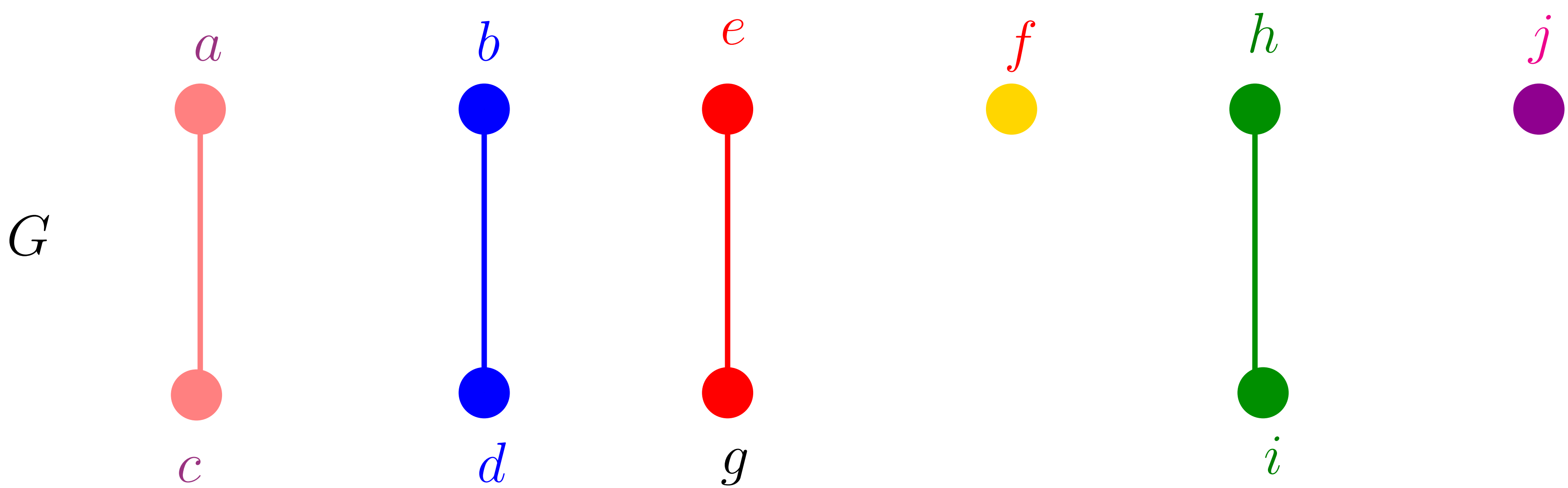


aresta	componentes
(a, c)	$\{a, c\}$ $\{b, d\}$ $\{e, g\}$ $\{f\}$ $\{h\}$ $\{i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

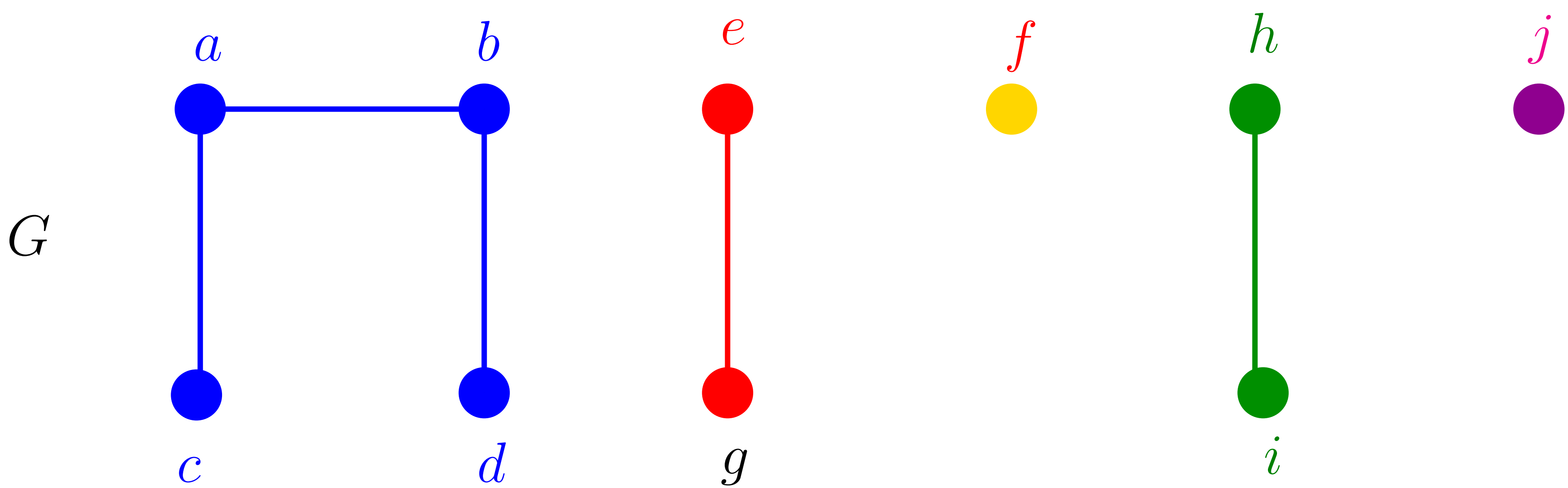


aresta	componentes
(h,i)	$\{a,c\}$ $\{b,d\}$ $\{e,g\}$ $\{f\}$ $\{h,i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico



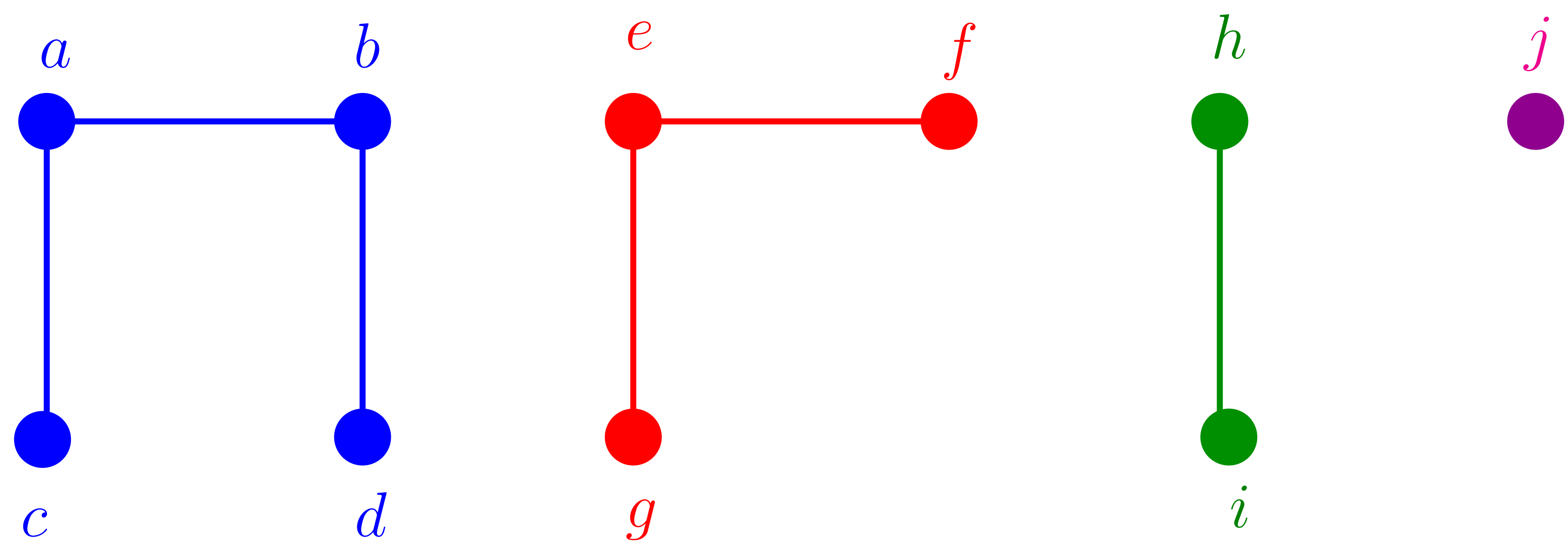
aresta	componentes
(a,b)	$\{a,b,c,d\}$ $\{e,g\}$ $\{f\}$ $\{h,i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

G

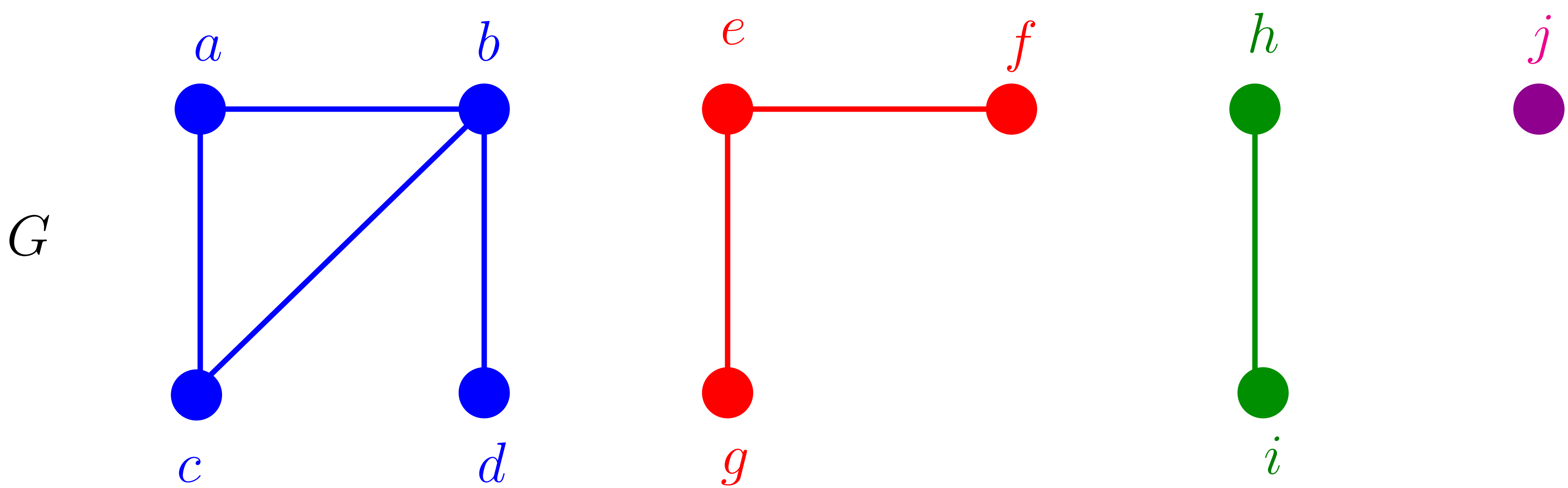


aresta	componentes
(e, f)	$\{a, b, c, d\}$ $\{e, f, g\}$ $\{h, i\}$ $\{j\}$

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 o representante do conjunto que contém x

Connected-Componets

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	$= n \times$ 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

Algoritmo de Kruskal

Encontra uma árvore geradora mínima (CLRS 23).

MST-KRUSKAL (G, w) $\quad \triangleright$ G conexo

coloque arestas em ordem crescente de w
$A \leftarrow \emptyset$
para cada vértice v **faça**
    MAKESET (v)
para cada aresta uv em ordem crescente de w **faça**
    se FINDSET $(u) \neq$ FINDSET (v)
        então UNION (u, v)
            $A \leftarrow A \cup \{uv\}$
devolva A

“Avô” de todos os algoritmos gulosos.

Conjuntos disjuntos dinâmicos

Seqüência de operações MAKESET, UNION, FINDSET

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

Que estrutura de dados usar?

Compromissos (*trade-off* s)

Conclusões

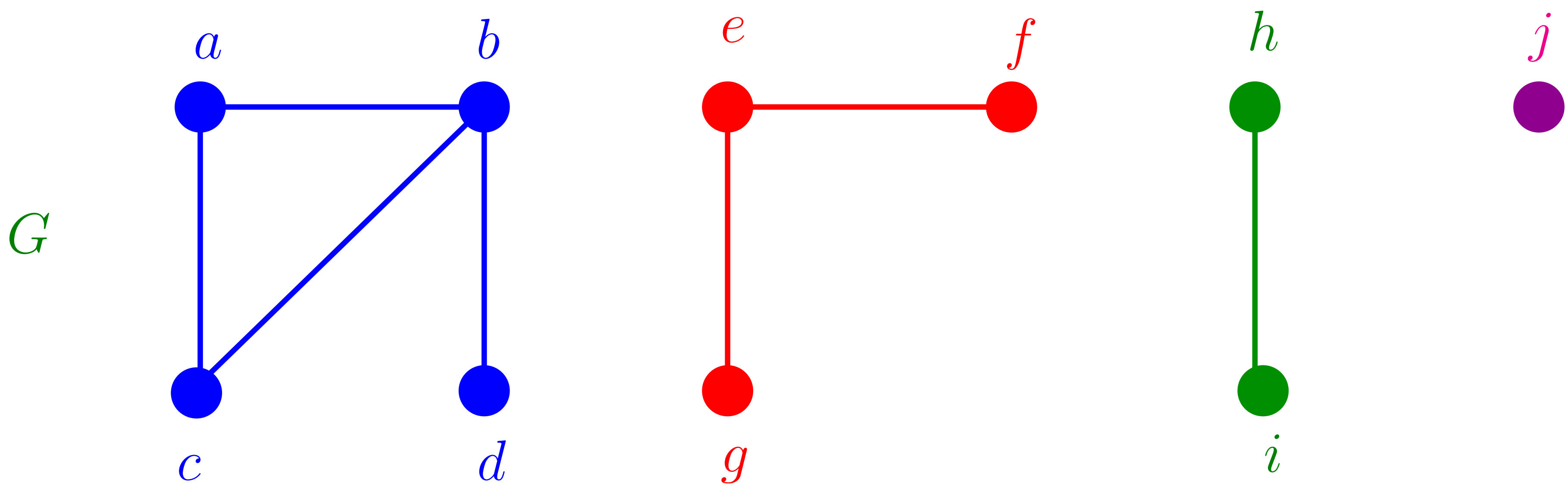
Se conjuntos disjuntos são representados através de listas ligadas e *weighted-union* é utilizada, então uma seqüência de m operações MAKESET, UNION e FINDSET, sendo que n são MAKESET, consome tempo $\mathrm{O}(m + n \lg n)$.

Se conjuntos disjuntos são representados através de listas ligadas e *weighted-union* é utilizada, então o algoritmos CONNECTED-COMPONENTS consome tempo $\mathrm{O}(m + n \lg n)$ e o algoritmo MST-KRUSKAL consome tempo $\mathrm{O}((n + m) \lg n)$.

No que se refere ao algoritmo MST-KRUSKAL, estamos supondo que $m = \mathrm{O}(n^2)$.

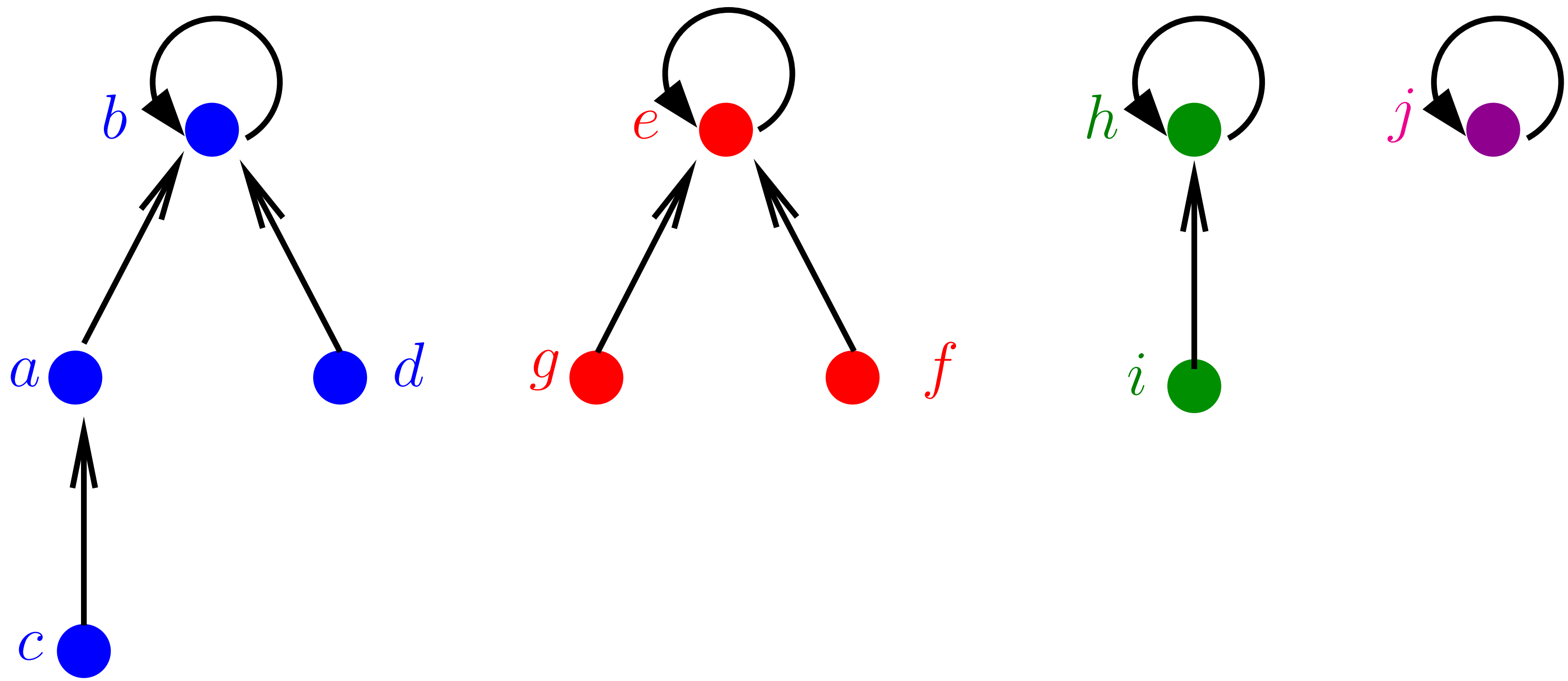
AULA 22

Estrutura *disjoint-set forest*



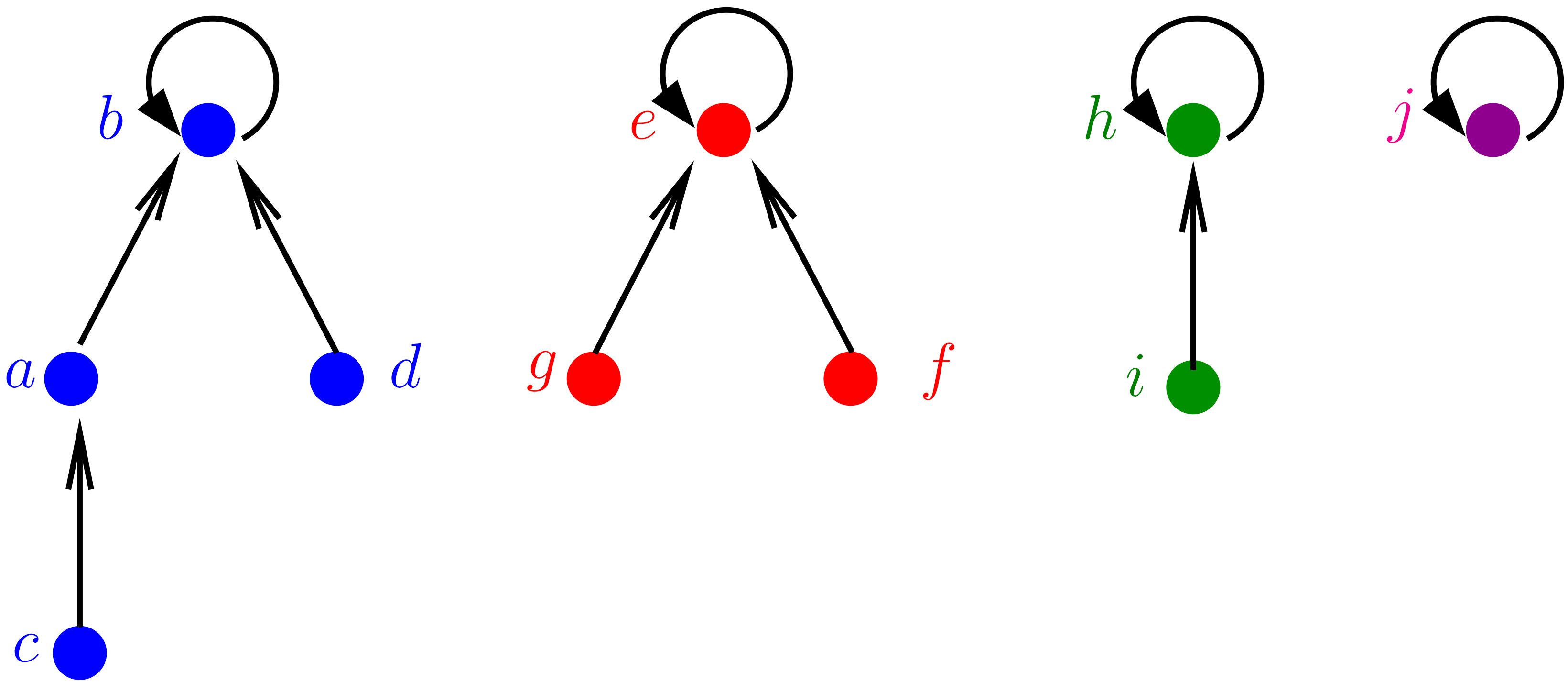
- cada conjunto tem uma $raiz$, que é o seu representate
- cada nó x tem um pai
- $pai[x] = x$ se e só se x é uma raiz

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 $\quad pai[x] \leftarrow x$

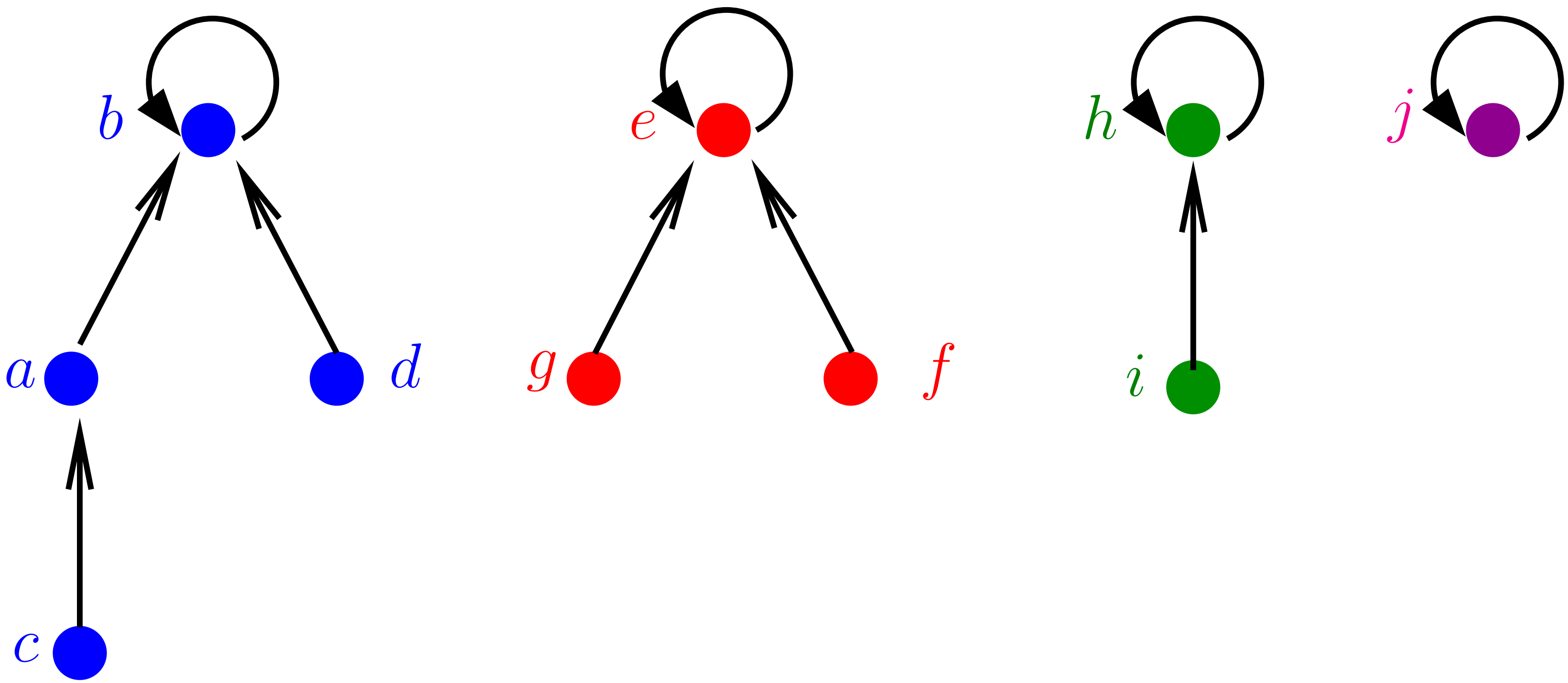
FINDSET$_0$ (x)

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

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

3 $\quad$ **devolva** x

FindSet$_1$



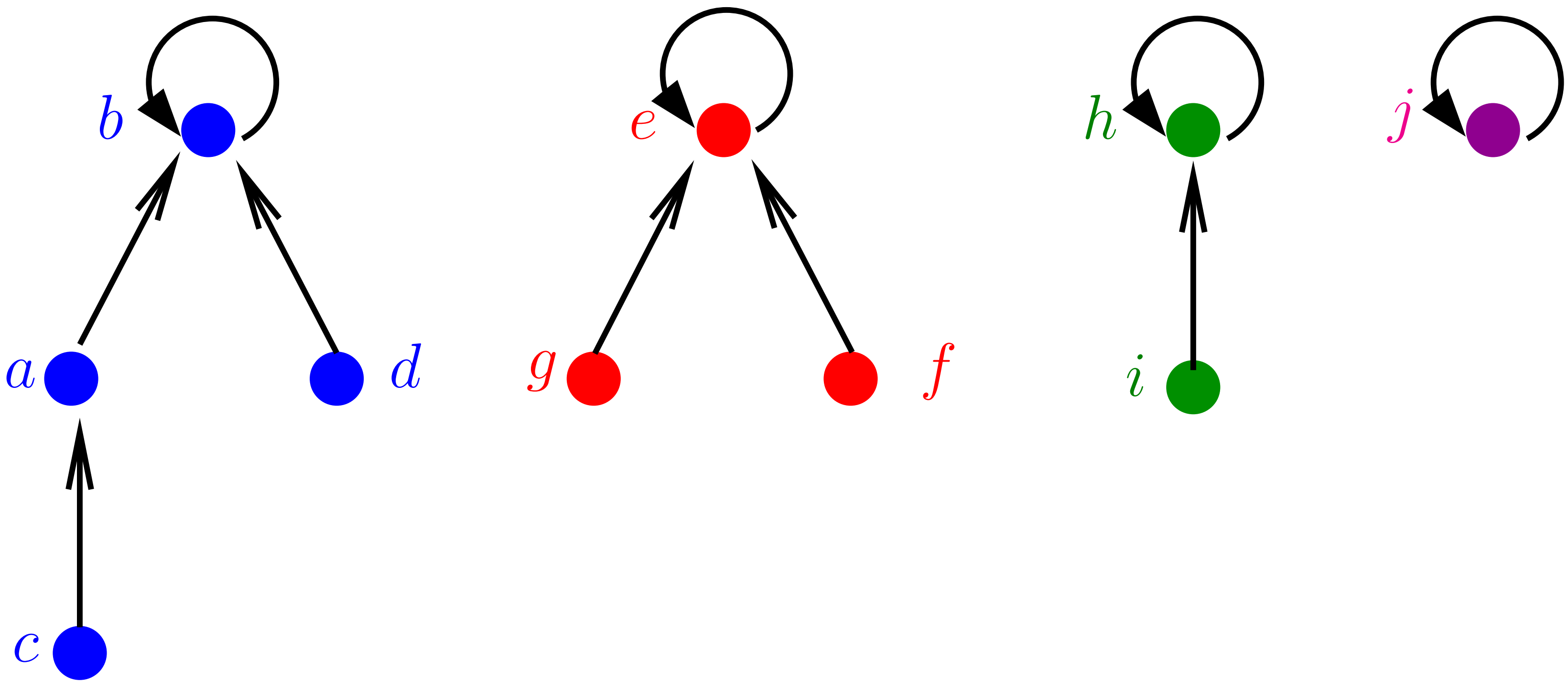
FINDSET$_1$ (x)

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

2 **então devolva** x

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

Union$_0$



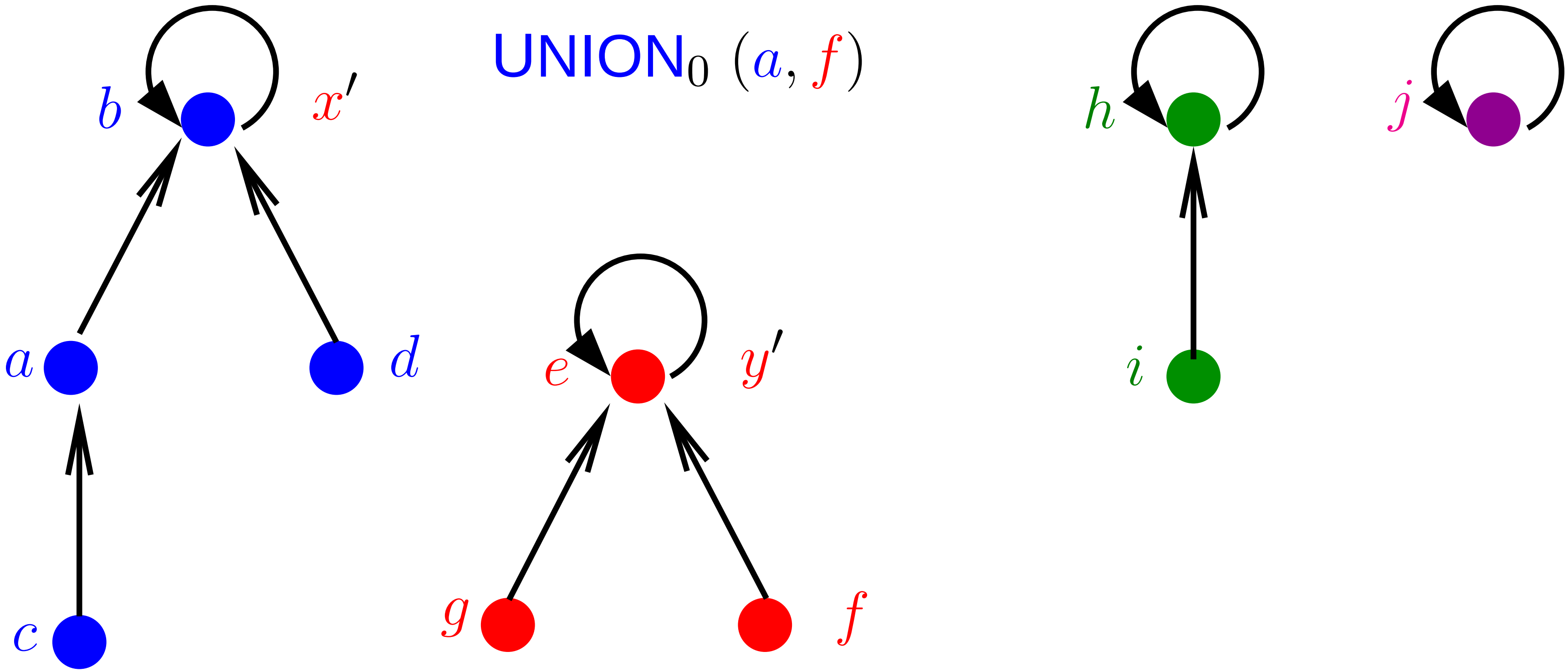
UNION$_0$ (x, y)

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

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

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

Union$_0$



UNION$_0$ (a, f)

UNION$_0$ (x, y)

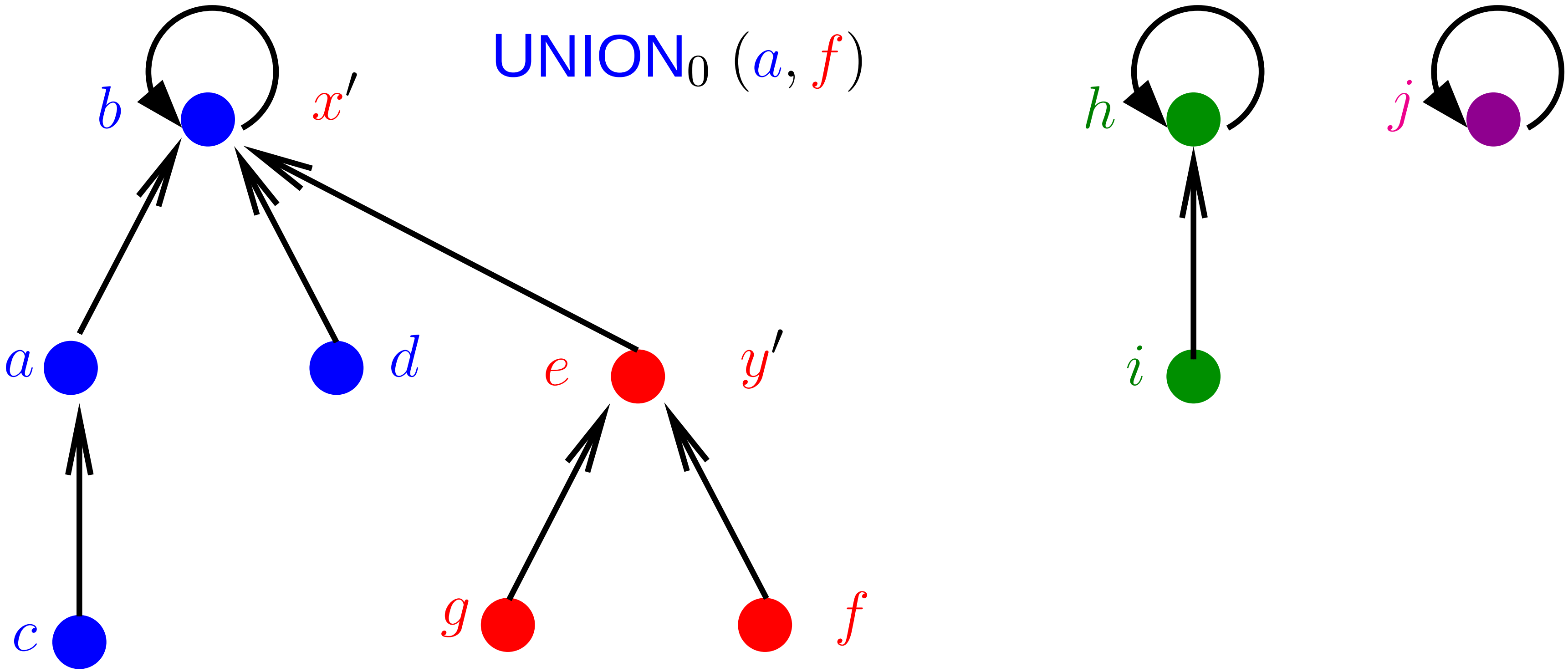
1 $\quad x' \leftarrow$ FINDSET$_0$ (x)

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

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

$\textbf{Union}_0$

$\text{UNION}_0\,(a, f)$



$\text{UNION}_0\,(x, y)$

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

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

$pai[y'] \leftarrow x'$

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

MAKESET$_0$ (x)

$pai[x] \leftarrow x$

UNION$_0$ (x, y)

$x' \leftarrow$ FINDSET$_0$ (x)
$y' \leftarrow$ FINDSET$_0$ (y)
$pai[y'] \leftarrow x'$

FINDSET$_1$ (x)

se $pai[x] = x$
    então devolva x
    senão devolva FINDSET$_1$ $(pai[x])$

Consumo de tempo

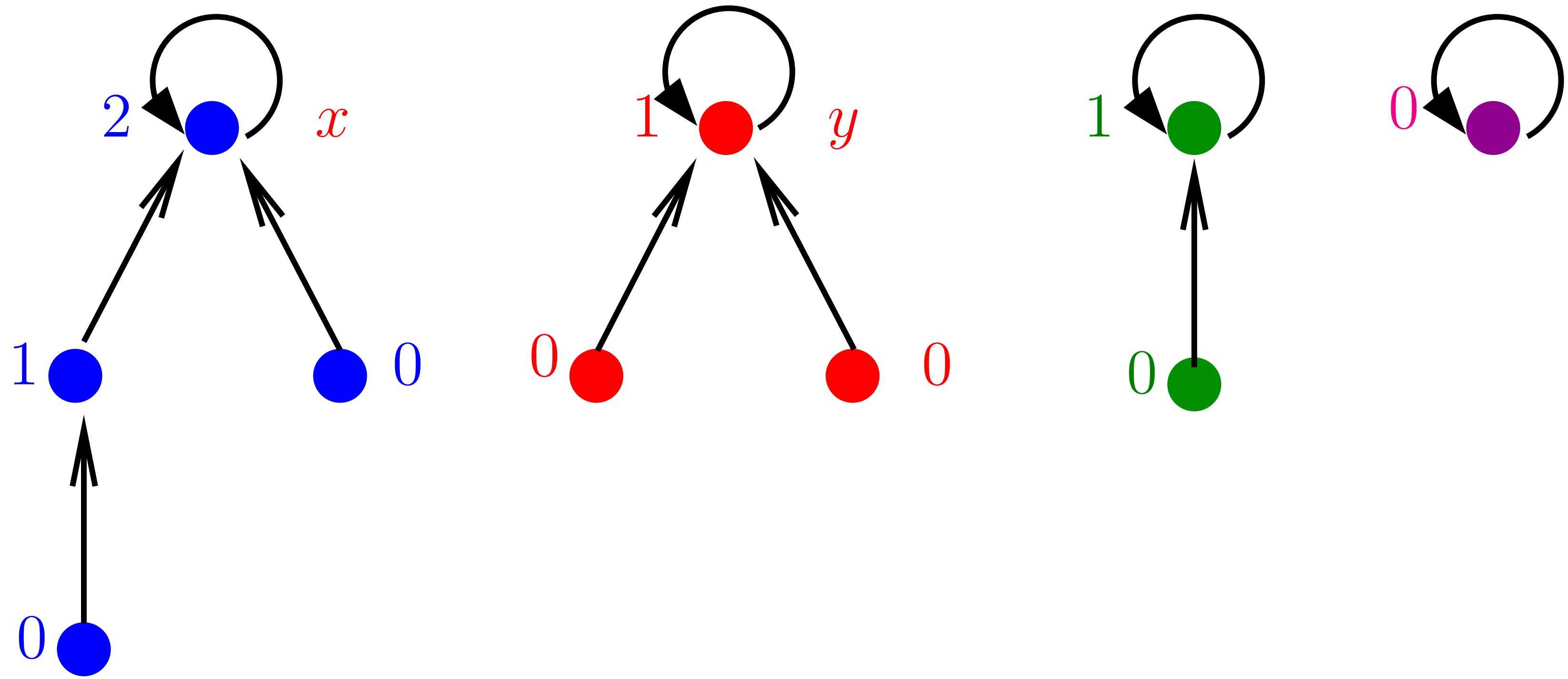
MAKESET$_0$	$\Theta(1)$
UNION$_0$	$\mathrm{O}(n)$
FINDSET$_0$	$\mathrm{O}(n)$

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

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

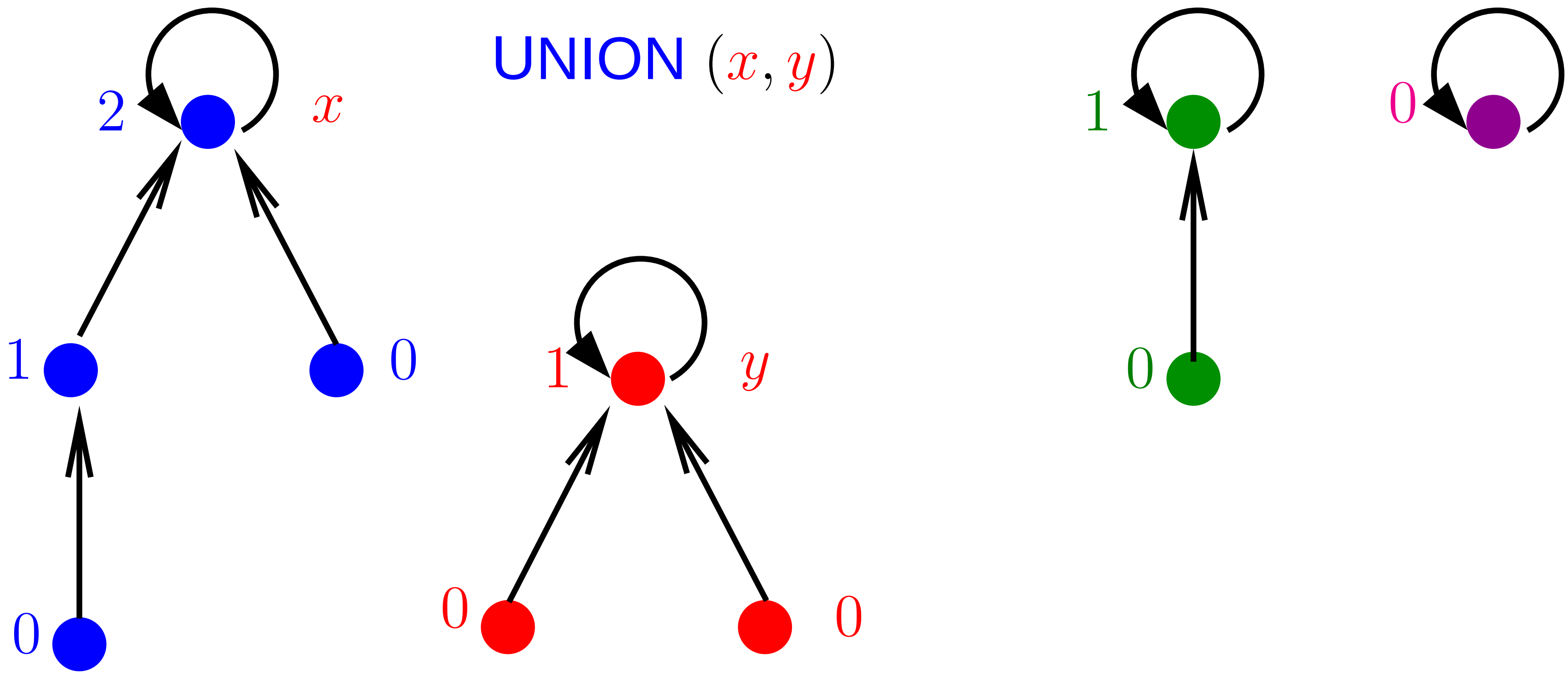
MAKESET (x)

1 $pai[x] \leftarrow x$

2 $rank[x] \leftarrow 0$

Melhoramento 1: *union by rank*

UNION (x, y)



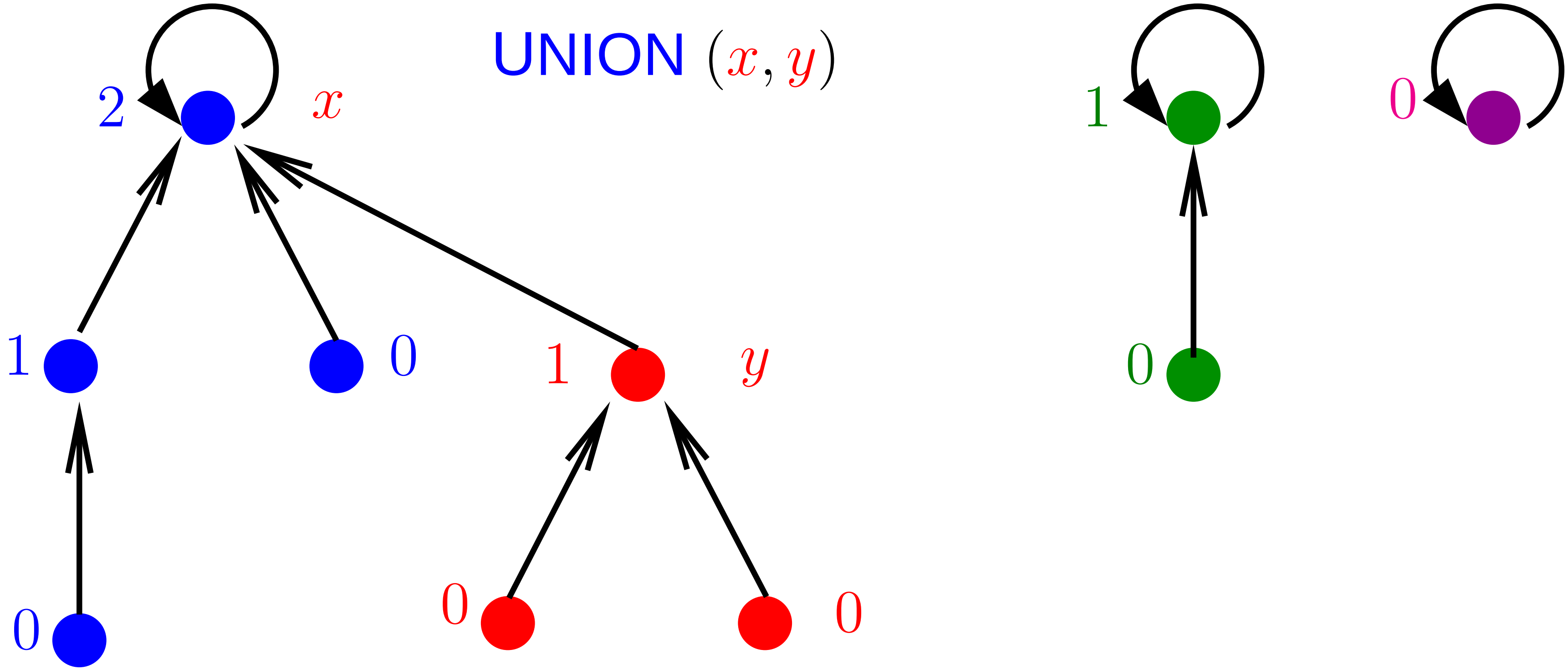
$rank[x]$ = posto do nó x

MAKESET (x)

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

2 $\quad rank[x] \leftarrow 0$

Melhoramento 1: *union by rank*



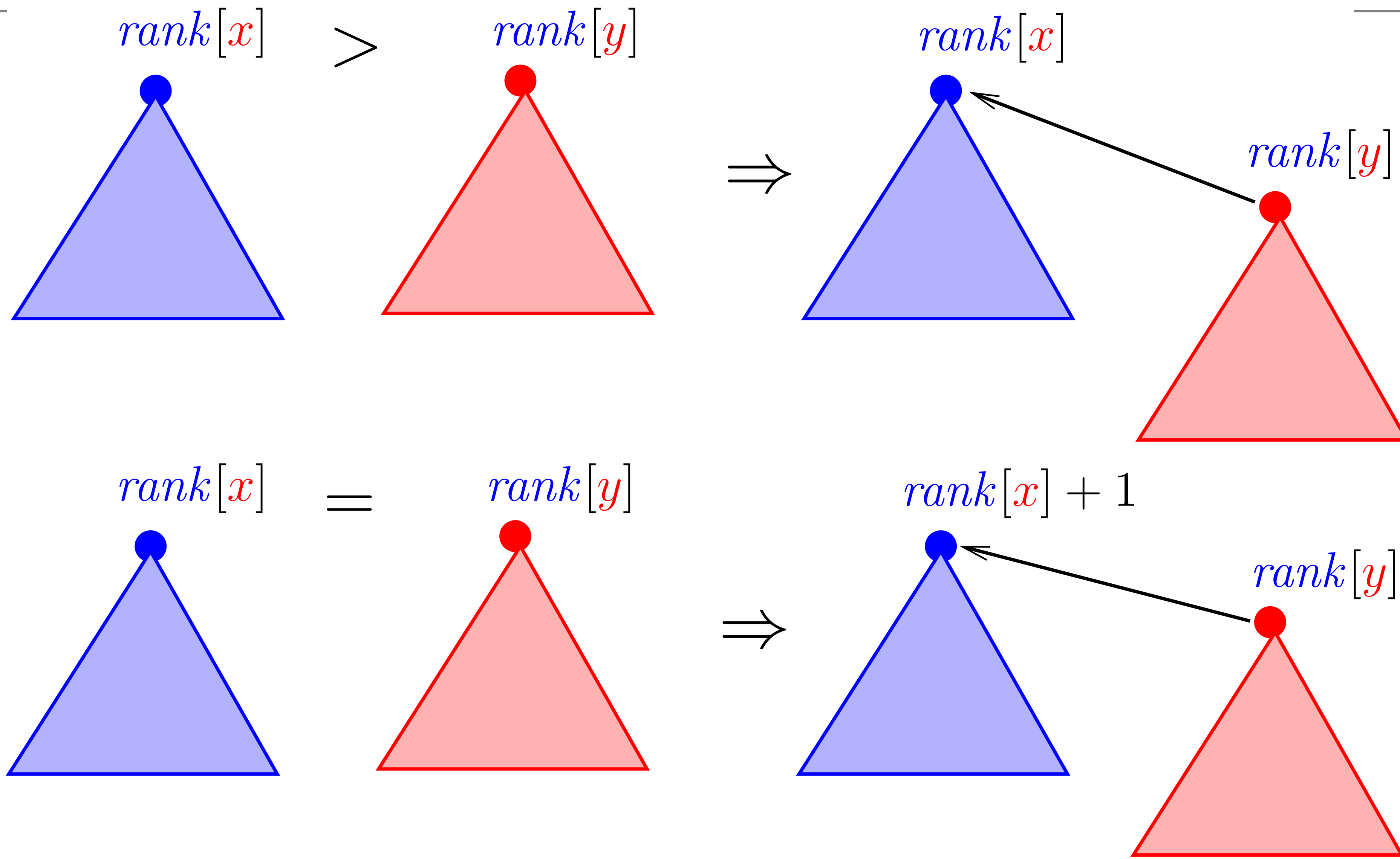
$rank[x]$ = posto do nó x

MAKESET (x)

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

2 $\quad rank[x] \leftarrow 0$

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: invariantes

- $rank[x] \leq rank[pai[x]]$ para cada nó x
- $rank[x] = rank[pai[x]]$ se e só se x é raiz
- $rank[pai[x]]$ é uma função não-descrente 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: $\mathrm{O}(m \lg n)$

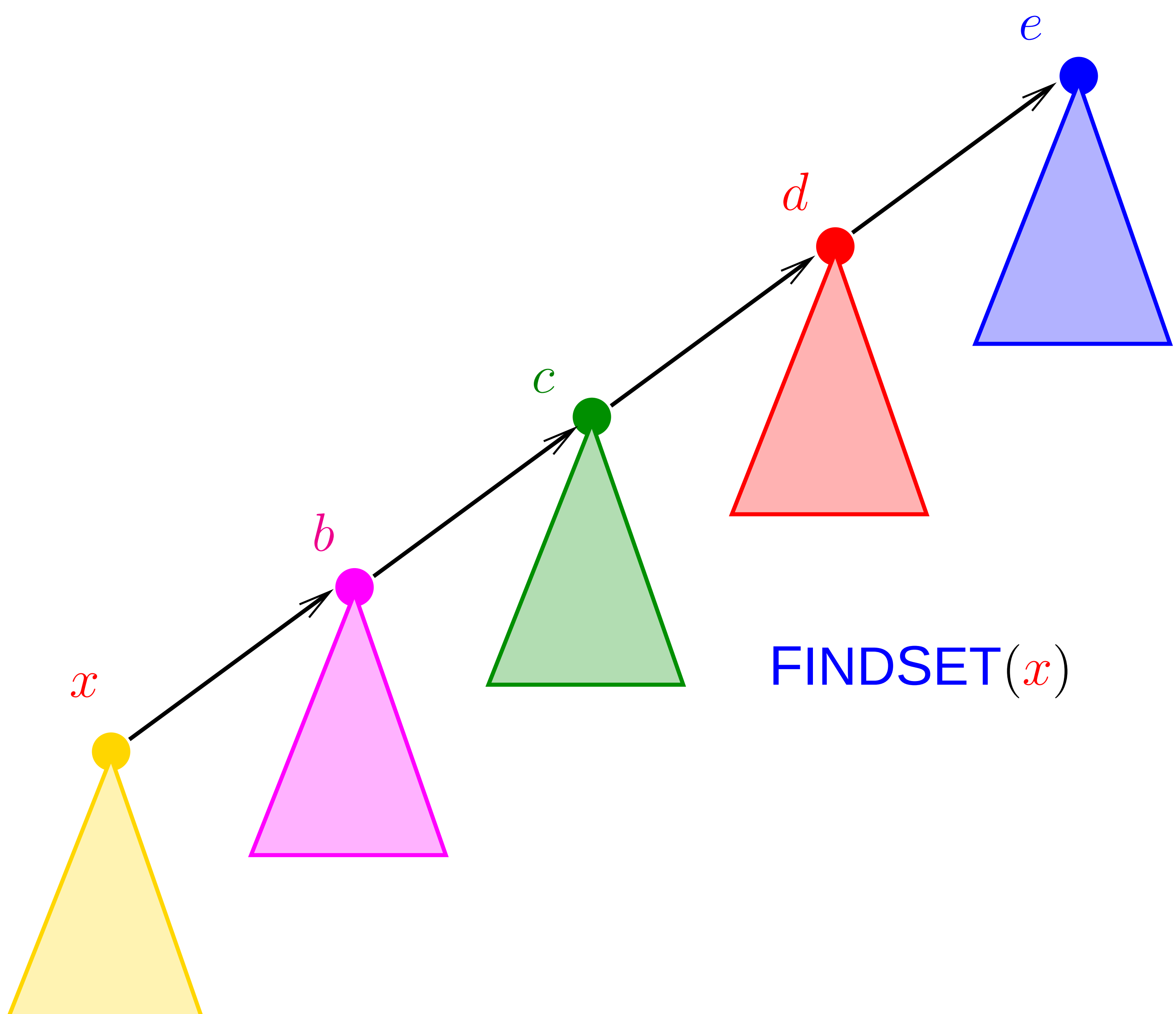
Conclusões

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

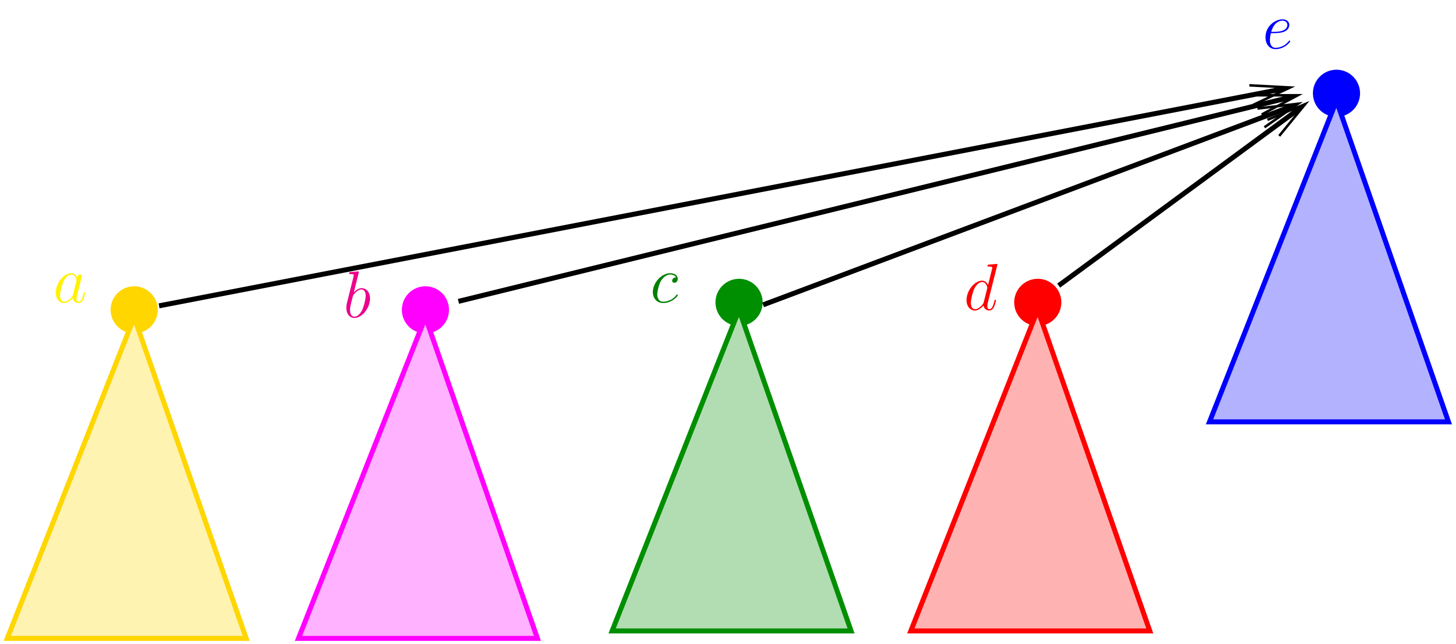
Se conjuntos disjuntos são representados através de disjoint-set forest com *union by rank*, então os algoritmos CONNECTED-COMPONENTS e MST-KRUSKAL consomem tempo $\mathrm{O}((n+m) \lg n)$.

No que se refere ao algoritmo MST-KRUSKAL, estamos supondo que $m = \mathrm{O}(n^2)$.

Melhoramento 2: *path compression*



Melhoramento 2: *path compression*



FINDSET(x)

Melhoramento 2: *path compression*

FINDSET (x) $\triangleright$ com "path compression"

1 **se** $x \neq pai[x]$

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

3 **devolva** $pai[x]$

- $rank[x] \leq rank[pai[x]]$ para cada nó x
- $rank[x] = rank[pai[x]]$ se e só se x é raiz
- $rank[pai[x]]$ é uma função não-descrente 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

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

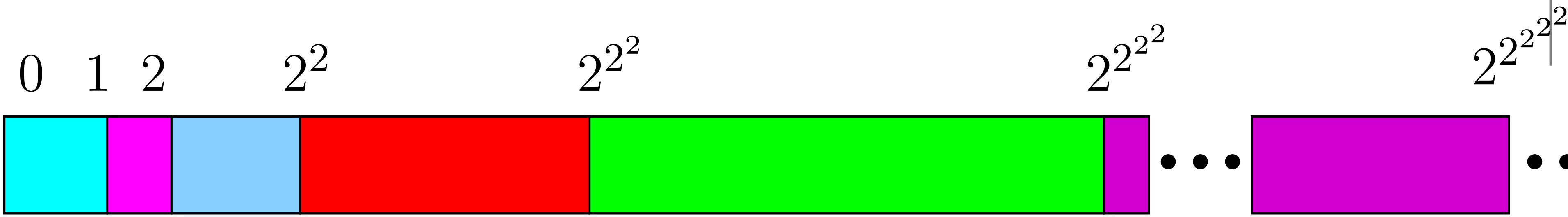
n	$\lg^* n$
1	0
2	1
3	2
4	2
5	3
$\vdots$	$\vdots$
15	3
16	3
$\vdots$	$\vdots$
65535	4
65536	4
$\vdots$	$\vdots$
$\underbrace{10000000000000000 \cdots 000000000000}_{80}$	5
$\vdots$	$\vdots$

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	$2^{2^{2^{2^2}}} > \underbrace{100000000000000000 \cdots 000000000000}_{80}$
$\vdots$	$\vdots$

Blocos



$$bloco[0] = [0\,..\,1]$$

$$bloco[1] = [2\,..\,2]$$

$$bloco[2] = [3\,..\,4]$$

$$bloco[3] = [5\,..\,16]$$

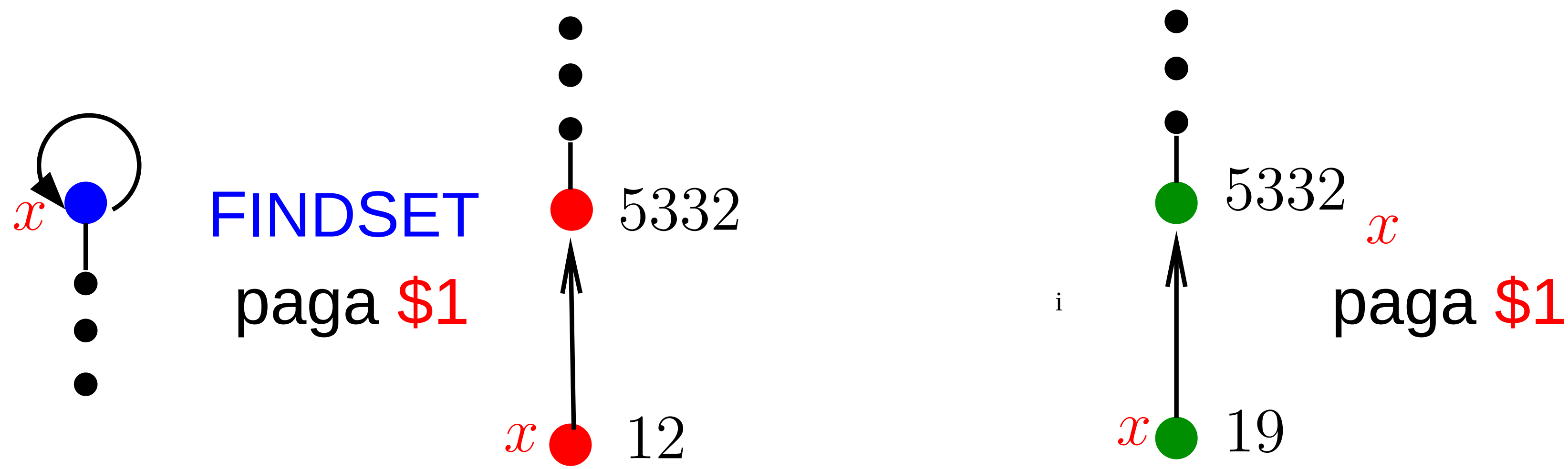
$$bloco[i] = [t(i-1)+1\,..\,t(i)]$$

Contabilidade

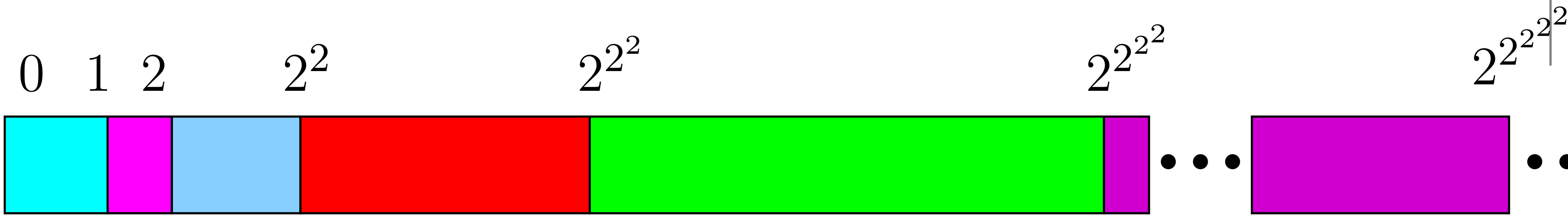
Custo de cada operação FINDSET(y) será contabilizado da seguinte maneira.

Para cada nó x no caminho de y até raiz:

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



Pagamento de cada FindSet



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

- $rank[x] \leq \lg n$ para cada nó x
- há nós em $< \lg^* n$ blocos:

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

$\Rightarrow i = \lg^* \lg n = \lg^* n - 1$

Pagamento de cada vértice

Suponha que x não é raiz.

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

Seja $N(i)$ o número de nós em $bloco[i]$, $i > 0$.
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 nós

Para cada i o valor pago por nós 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 nós em $< \lg^* n$ blocos

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

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

Conclusões

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

Se conjuntos disjuntos são representados através de disjoint-set forest com *union by rank* e *path compression*, então o algoritmo CONNECTED-COMPONENTS consome tempo $\mathrm{O}((n+m)\lg^* n)$ e o algoritmo MST-KRUSKAL consome tempo $\mathrm{O}(n \lg^* n + m \lg n)$.

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

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

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.)