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 $i \neq j$.

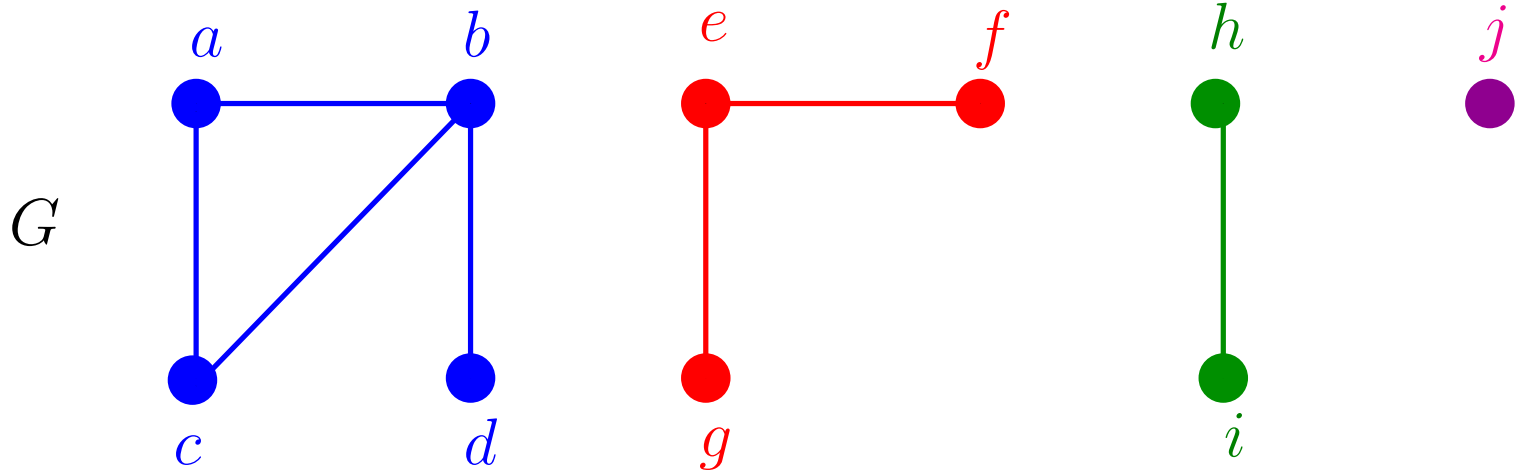
Conjuntos disjuntos

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

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

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

G

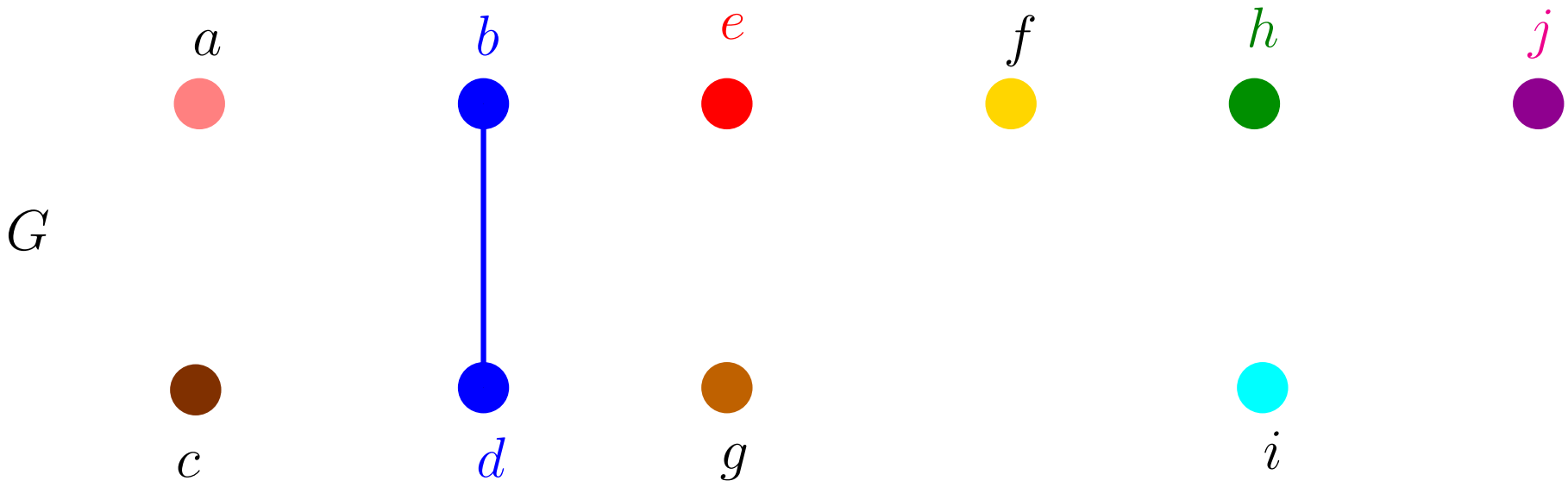
c d g i

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

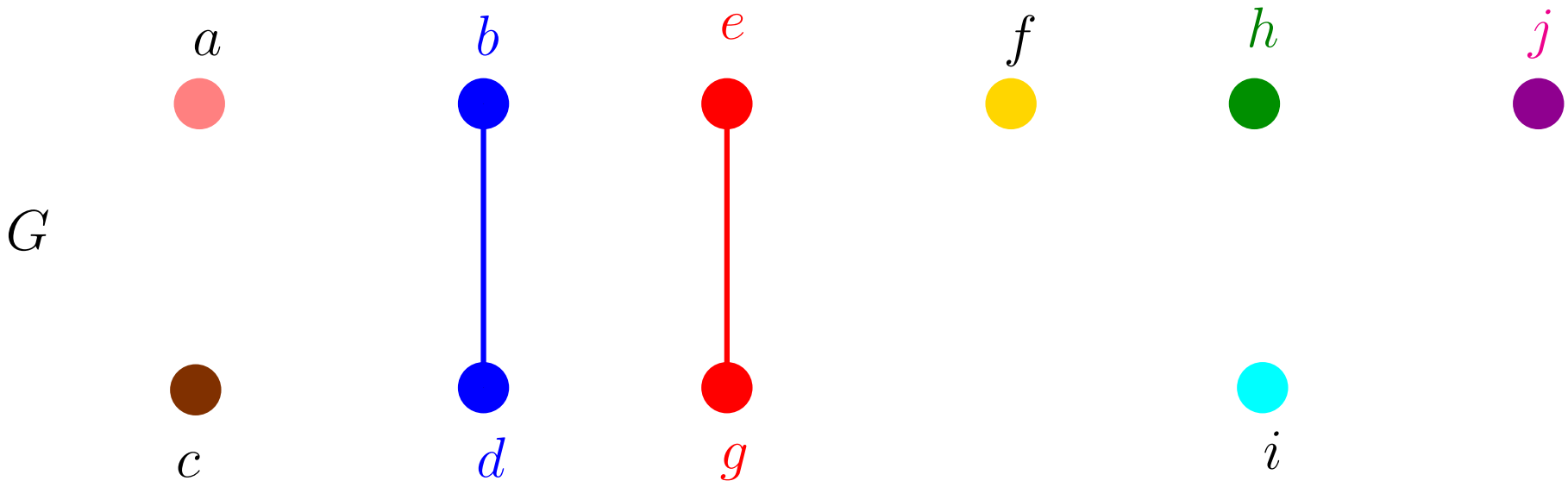


aresta	componentes
(b,d)	$\{a\}$ $\{b,d\}$ $\{c\}$ $\{e\}$ $\{f\}$ $\{g\}$ $\{h\}$ $\{i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

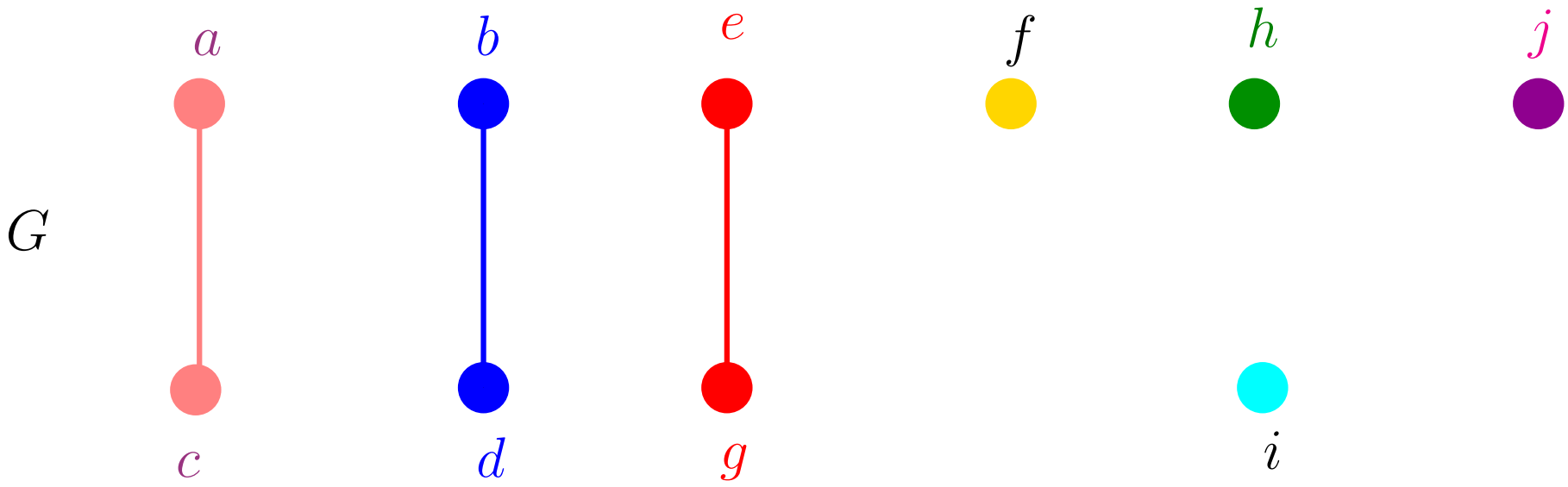


aresta	componentes
(e, g)	$\{a\}\ \{b, d\}\ \{c\}\ \{e, g\}\ \{f\}\ \{h\}\ \{i\}\ \{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

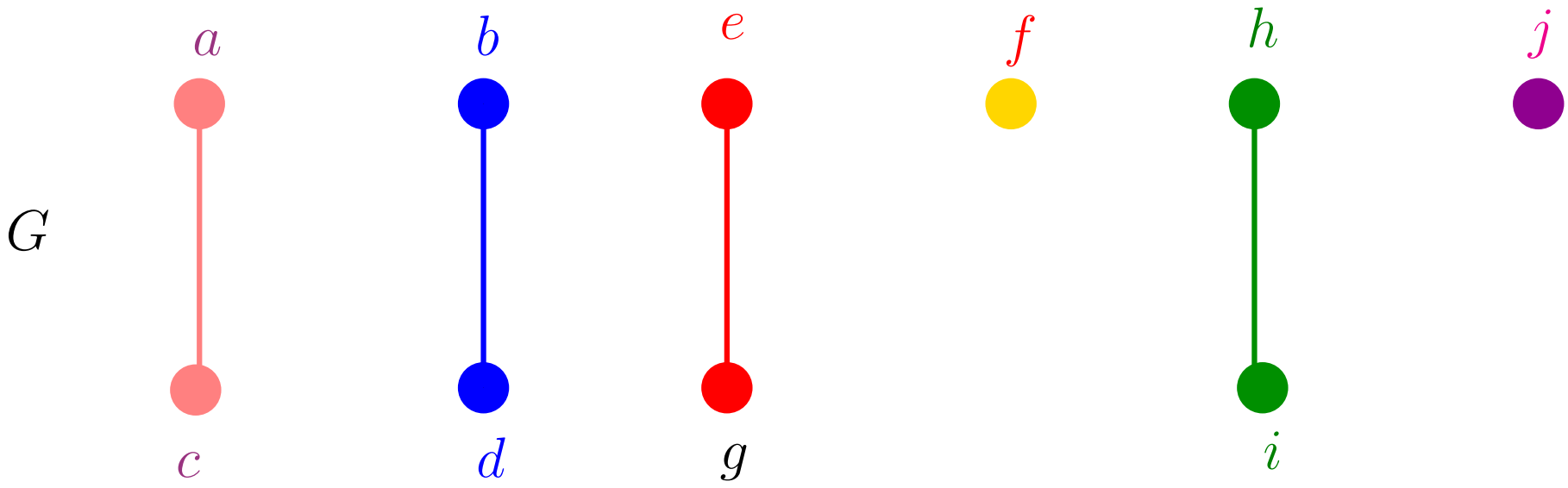


aresta	componentes
(a, c)	$\{a, c\}$ $\{b, d\}$ $\{e, g\}$ $\{f\}$ $\{h\}$ $\{i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico



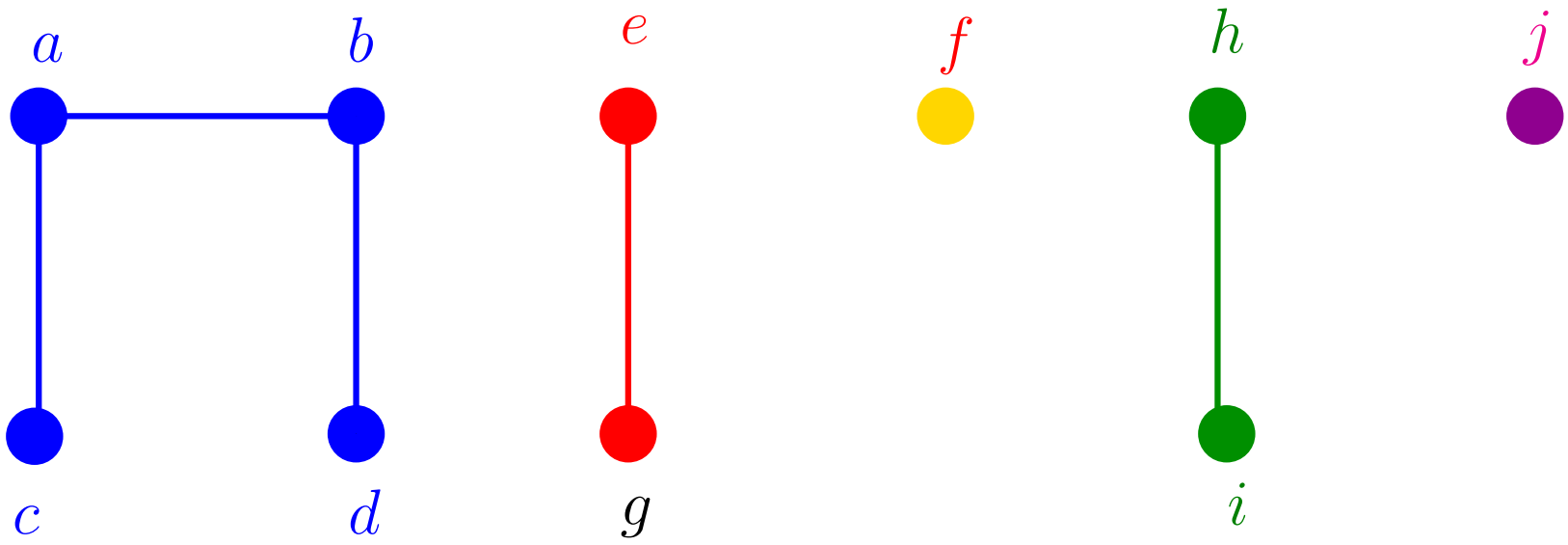
aresta	componentes
(h,i)	$\{a,c\}$ $\{b,d\}$ $\{e,g\}$ $\{f\}$ $\{h,i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

G



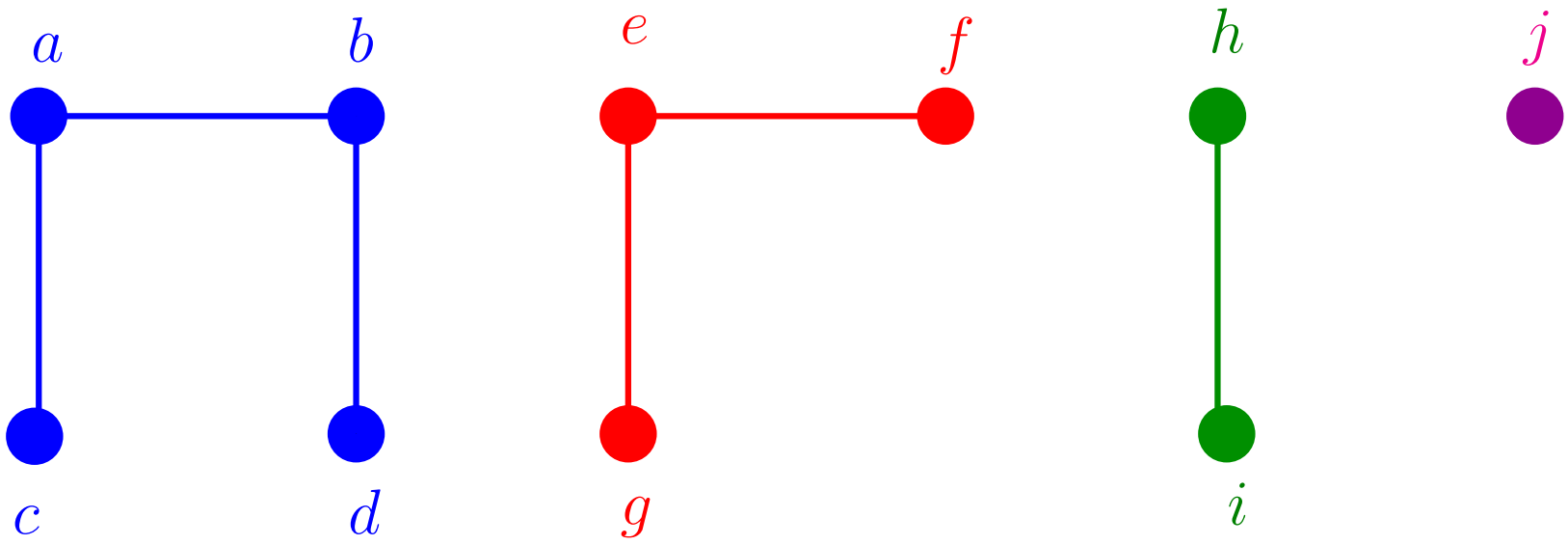
aresta	componentes
(a,b)	$\{a,b,c,d\}$ $\{e,g\}$ $\{f\}$ $\{h,i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

G



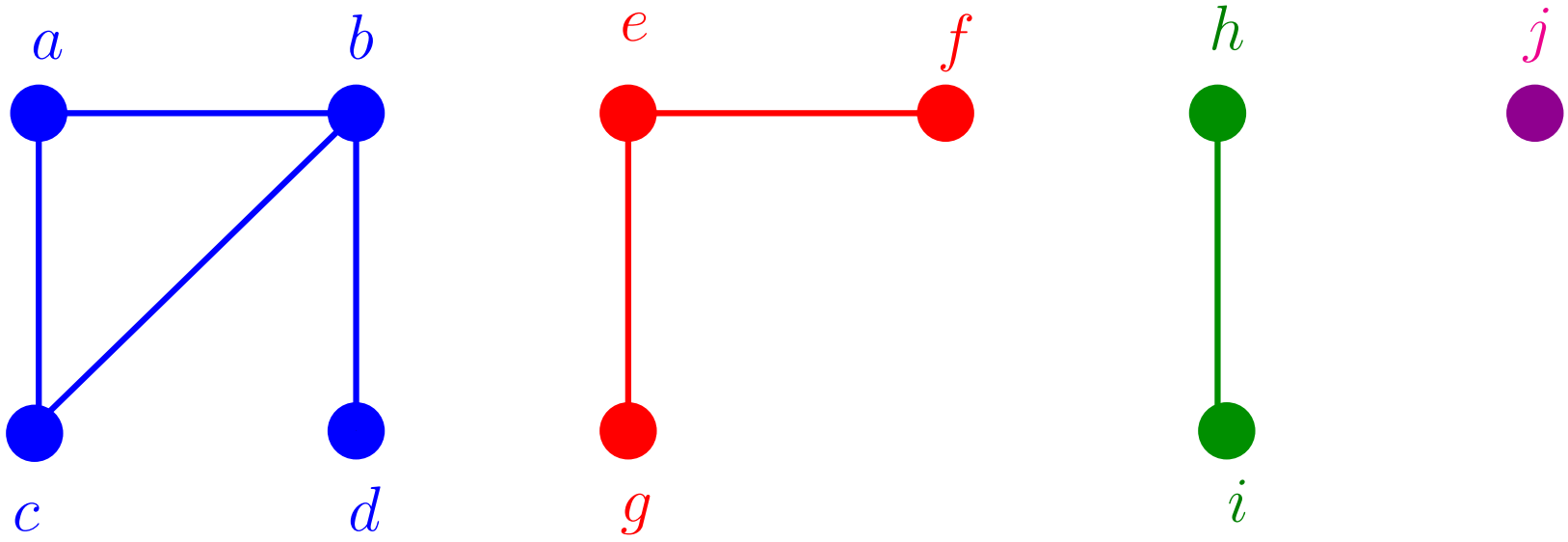
aresta	componentes
(e, f)	$\{a, b, c, d\}$ $\{e, f, g\}$ $\{h, i\}$ $\{j\}$

Coleção disjunta dinâmica

Conjuntos são modificados ao longo do tempo

Exemplo: grafo dinâmico

G



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) ≠ 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).

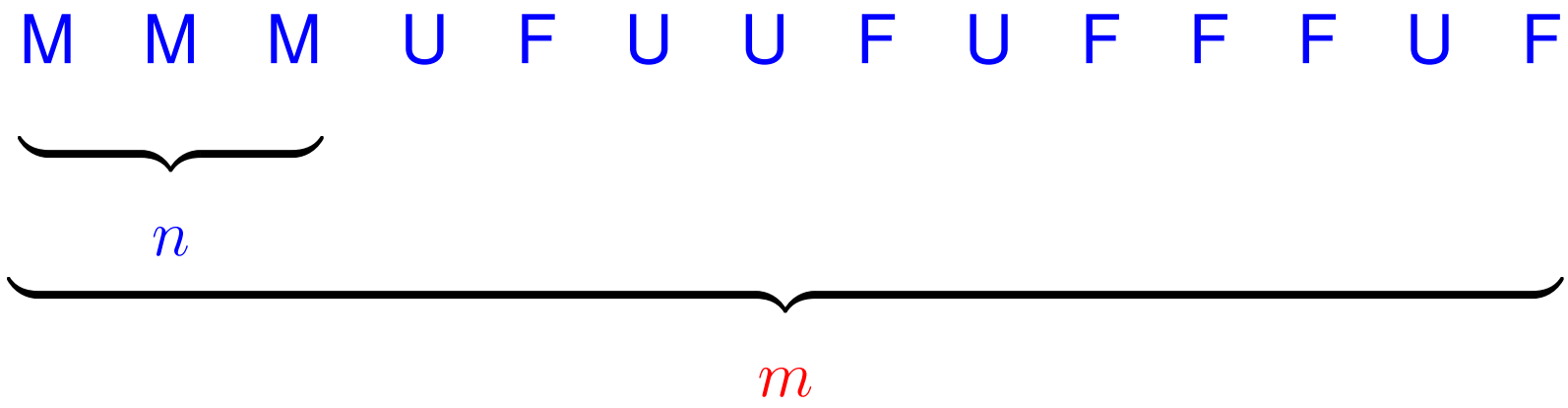
MST-KRUSKAL (G, w) $\quad \triangleright$ G conexo

```
−1   coloque arestas em ordem crescente de w
 0   A ← ∅
 1   para cada vértice v faça
 2       MAKESET (v)
 3   para cada aresta uv em ordem crescente de w faça
 4       se FINDSET (u) ≠ FINDSET (v)
 5           então UNION (u, v)
 8                 A ← A ∪ {uv}
 9   devolva A
```

"Avô" de todos os algoritmos gulosos.

Conjuntos disjuntos dinâmicos

Seqüência de operações MAKESET, UNION, FINDSET



Que estrutura de dados usar?

Compromissos (*trade-off*s)

Conclusões

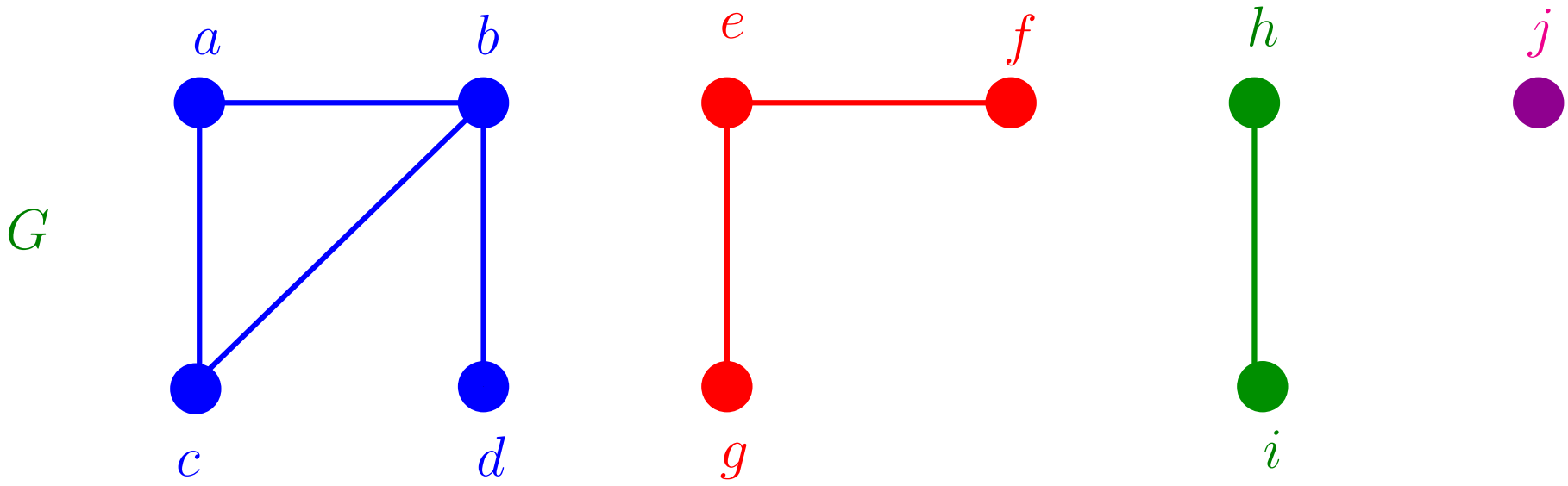
Se conjuntos disjuntos são representados através de listas ligadas e *weighted-union* é utilizada, então uma seqüência de m operações MAKESET, UNION e FINDSET, sendo que n são MAKESET, consome tempo $\mathrm{O}(m + n \lg n)$.

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 $m = \mathrm{O}(n^2)$.

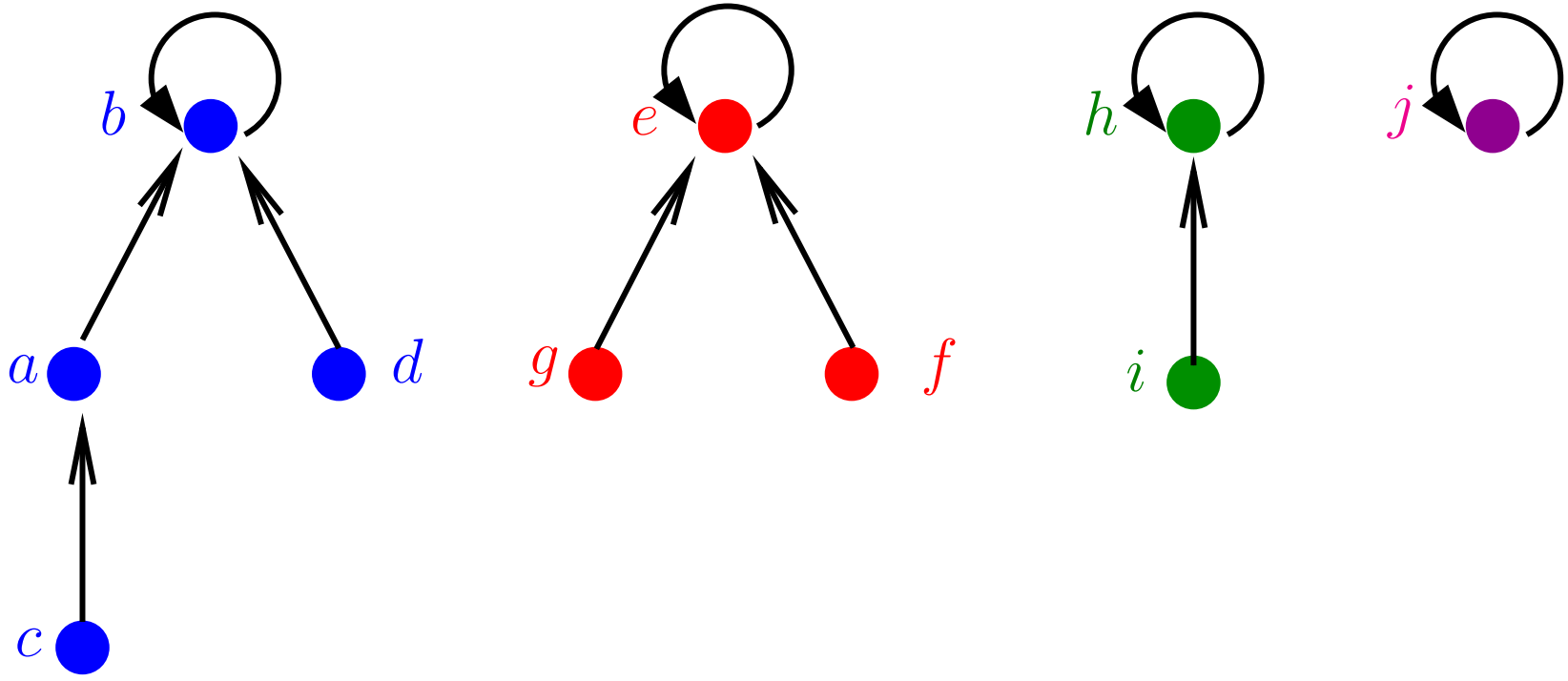
AULA 22

Estrutura *disjoint-set forest*



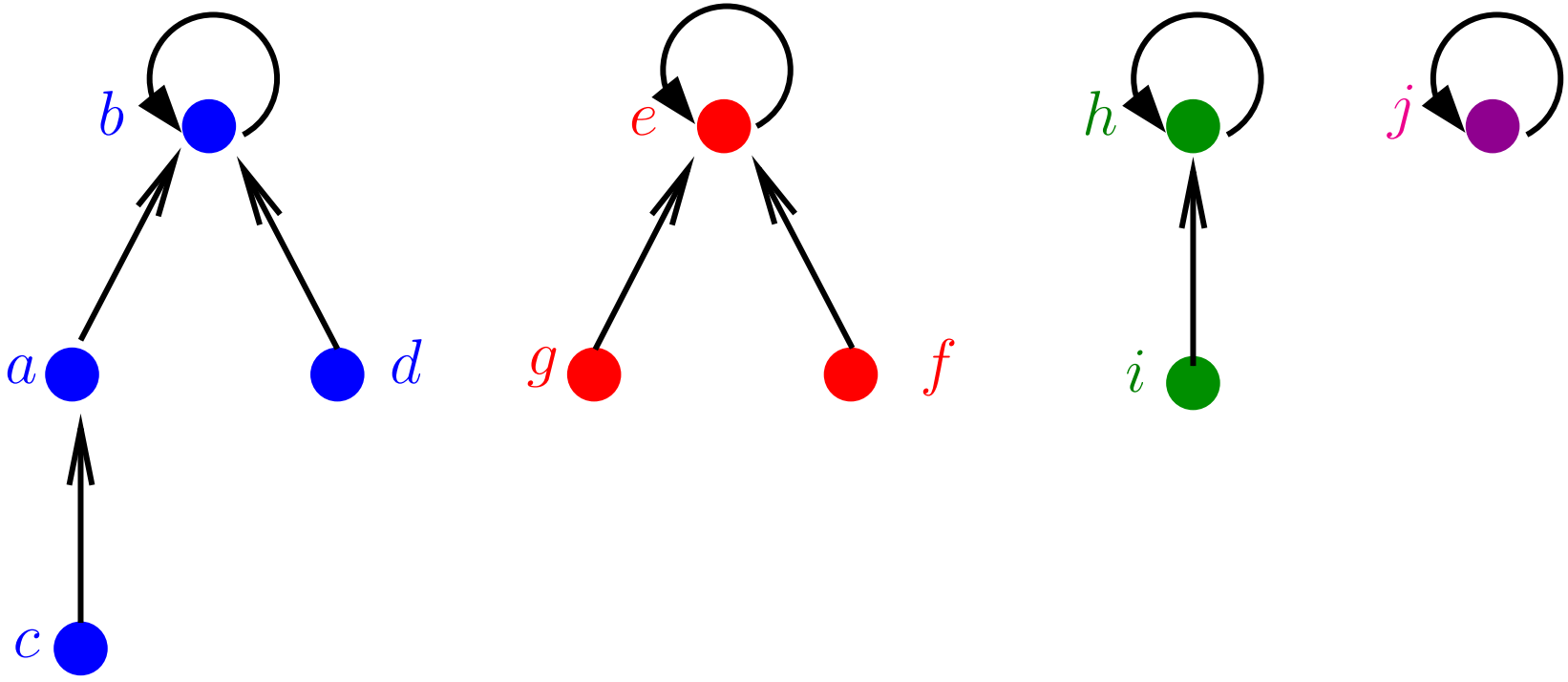
- cada conjunto tem uma $raiz$, que é o seu representate
- cada nó x tem um pai
- $pai[x] = x$ se e só se x é uma raiz

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$_0$



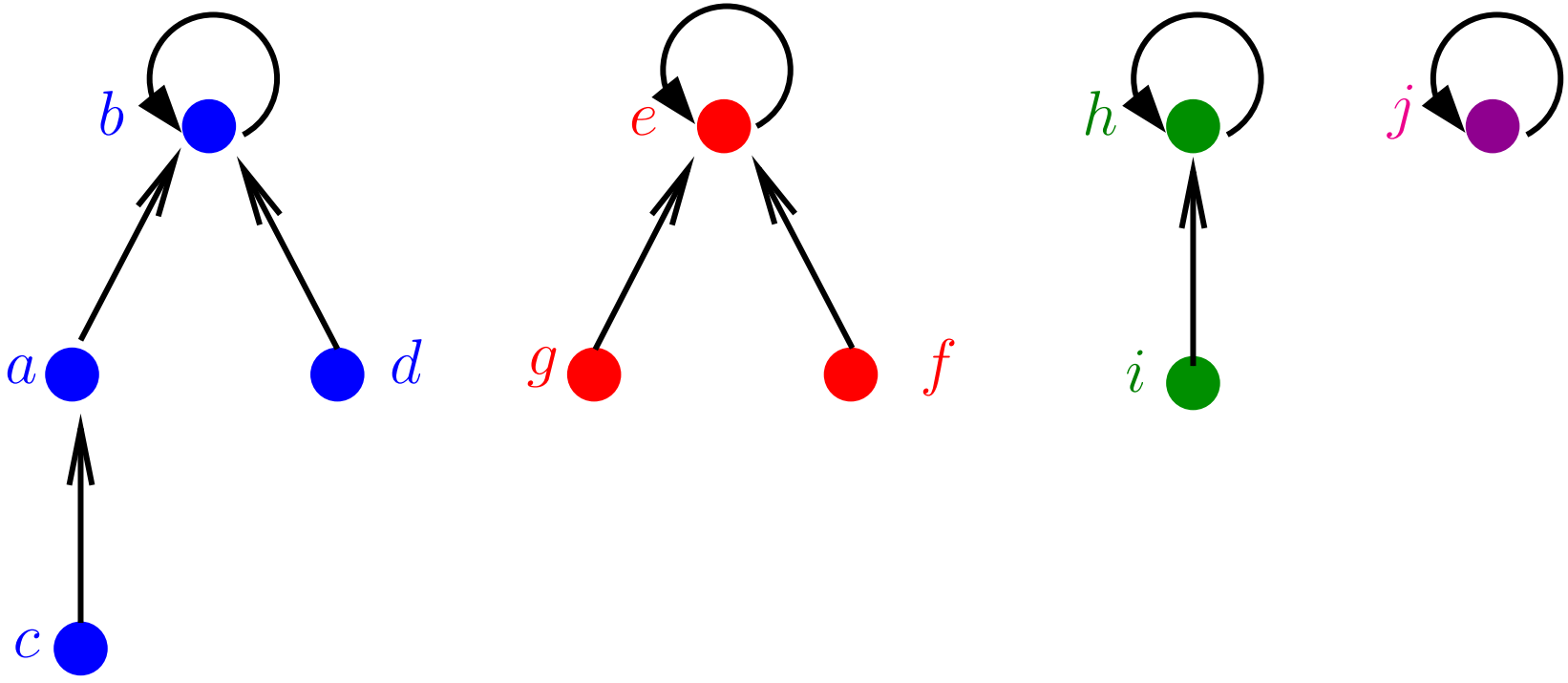
MAKESET$_0$ (x)

```
1    pai[x] ← x
```

FINDSET$_0$ (x)

```
1    enquanto pai[x] ≠ x faça
2          x ← pai[x]
3    devolva x
```

$\mathbf{FindSet}_1$



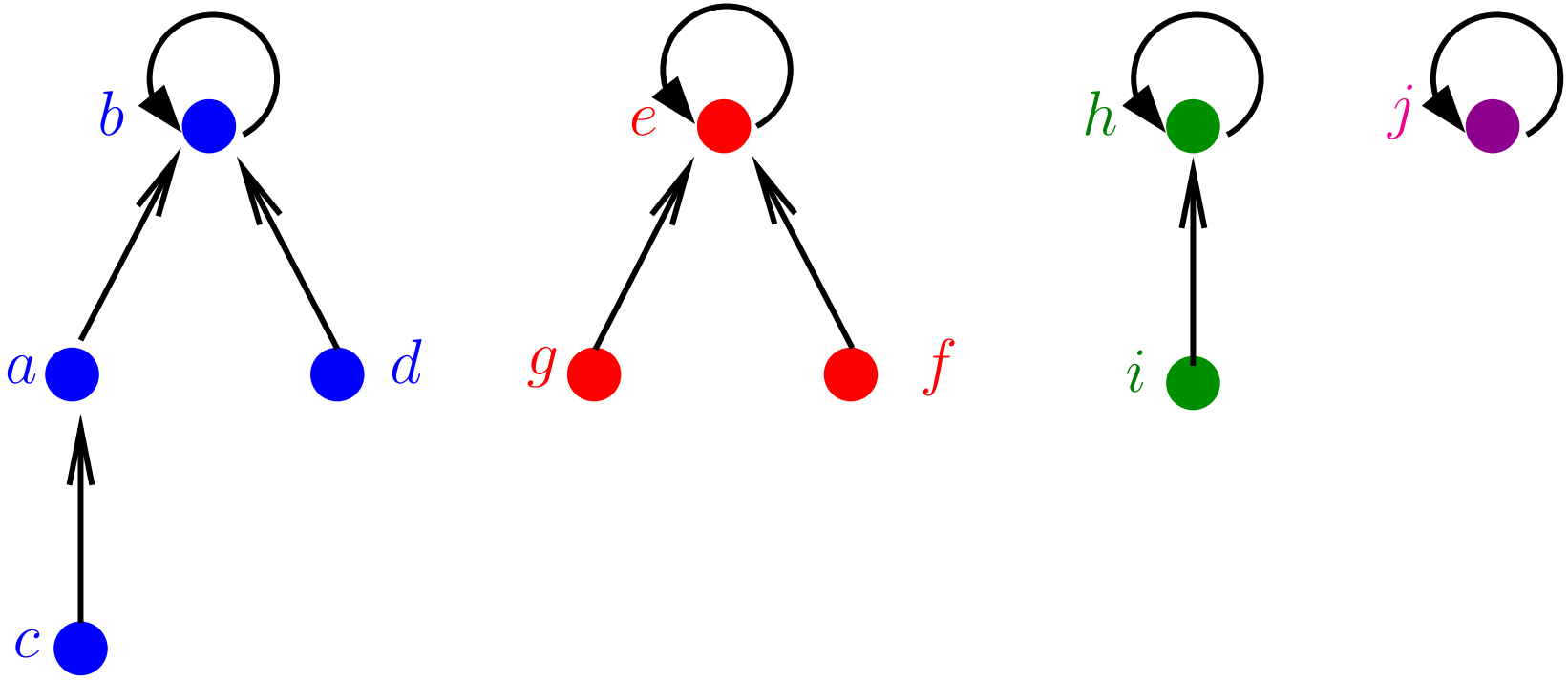
$\text{FINDSET}_1\ (x)$

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

2 **então devolva** x

3 **senão devolva** $\text{FINDSET}_1\ (pai[x])$

$\mathbf{Union}_0$



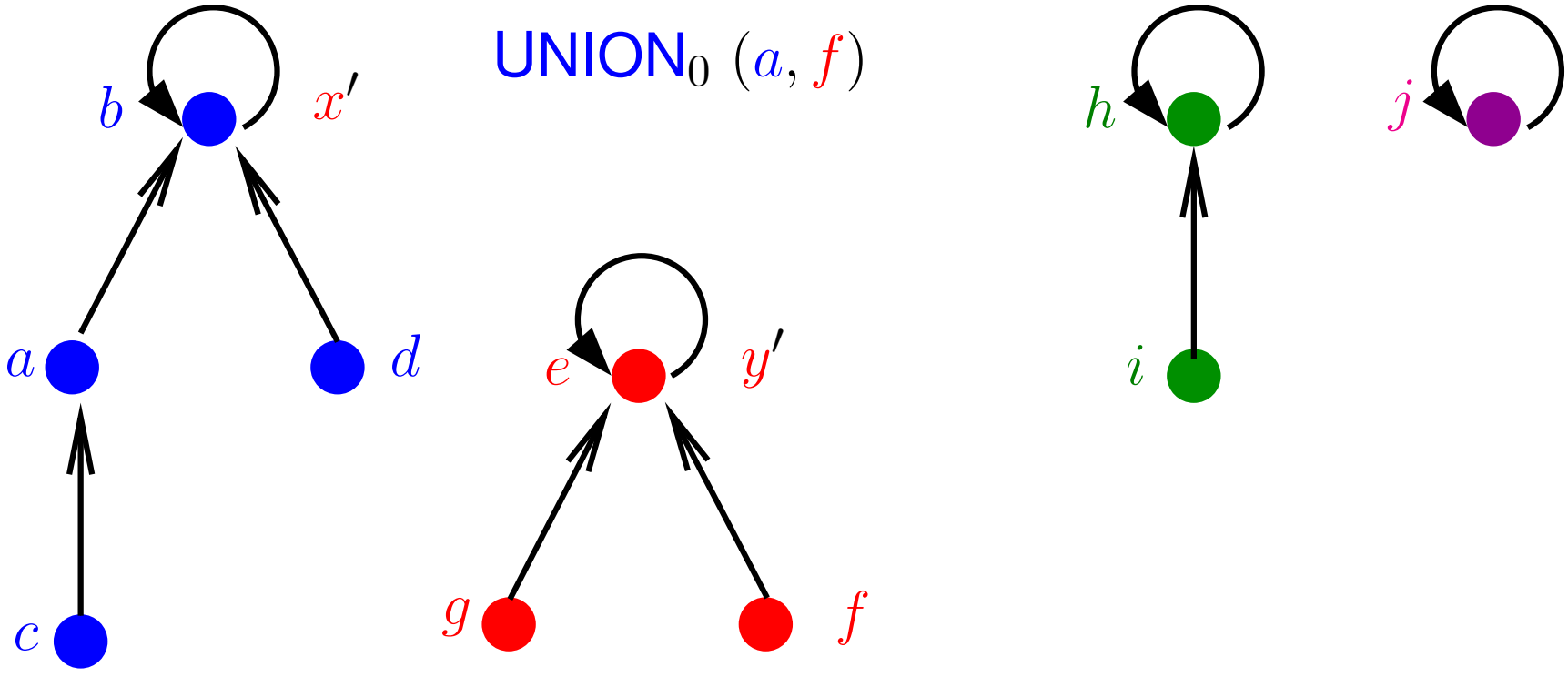
$\mathsf{UNION}_0\ (x, y)$

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

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

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

$\textsc{Union}_0$



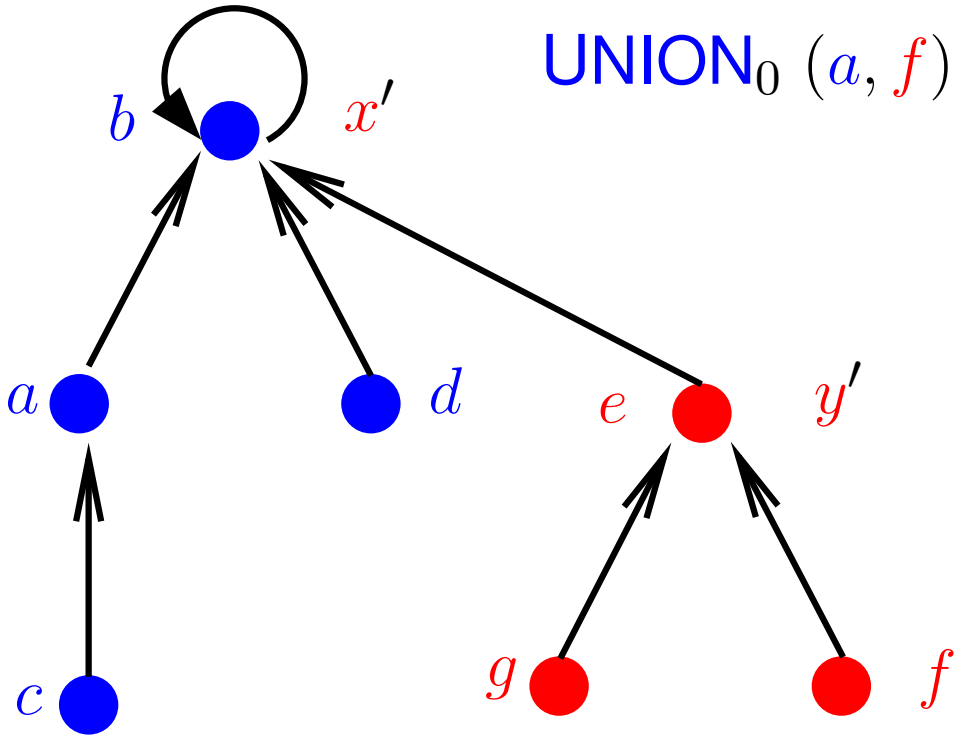
UNION$_0$ (x, y)

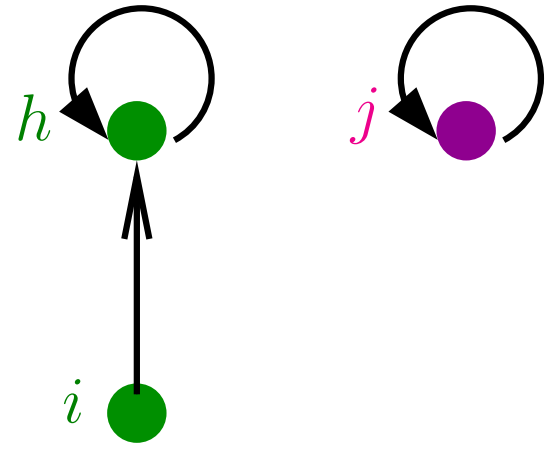
1 $x' \leftarrow$ FINDSET$_0$ (x)

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 $pai[y'] \leftarrow x'$

$\textbf{MakeSet}_0$, $\textbf{Union}_0$ e $\textbf{FindSet}_1$

$\text{MAKESET}_0\ (x)$

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

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

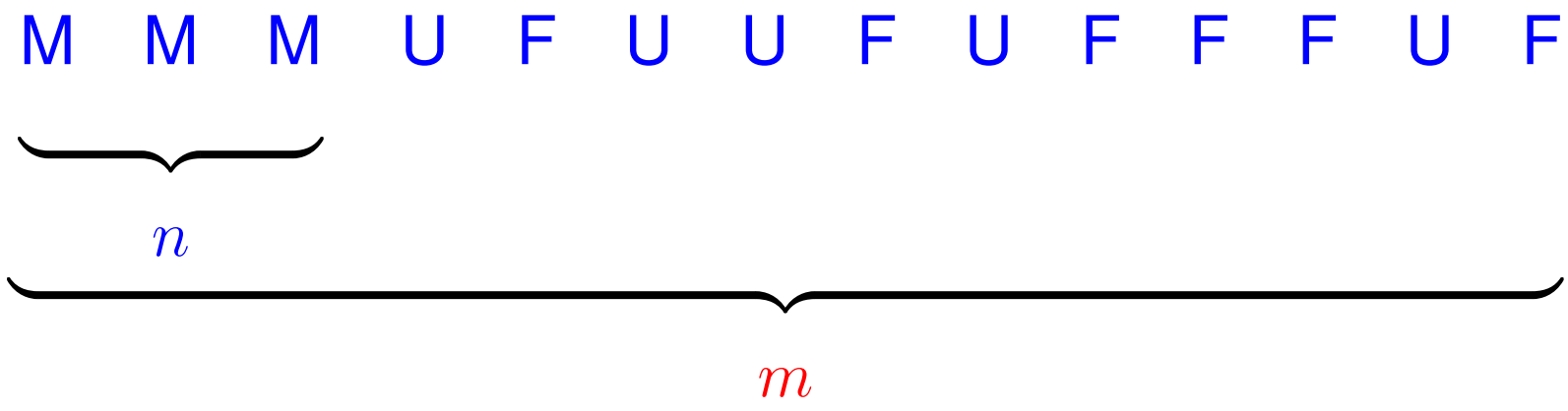
1 $\quad x' \leftarrow \text{FINDSET}_0\ (x)$
2 $\quad y' \leftarrow \text{FINDSET}_0\ (y)$
3 $\quad pai[y'] \leftarrow x'$

$\text{FINDSET}_1\ (x)$

1 $\quad$ **se** $pai[x] = x$
2 $\qquad$ **então devolva** x
3 $\qquad$ **senão devolva** $\text{FINDSET}_1\ (pai[x])$

Consumo de tempo

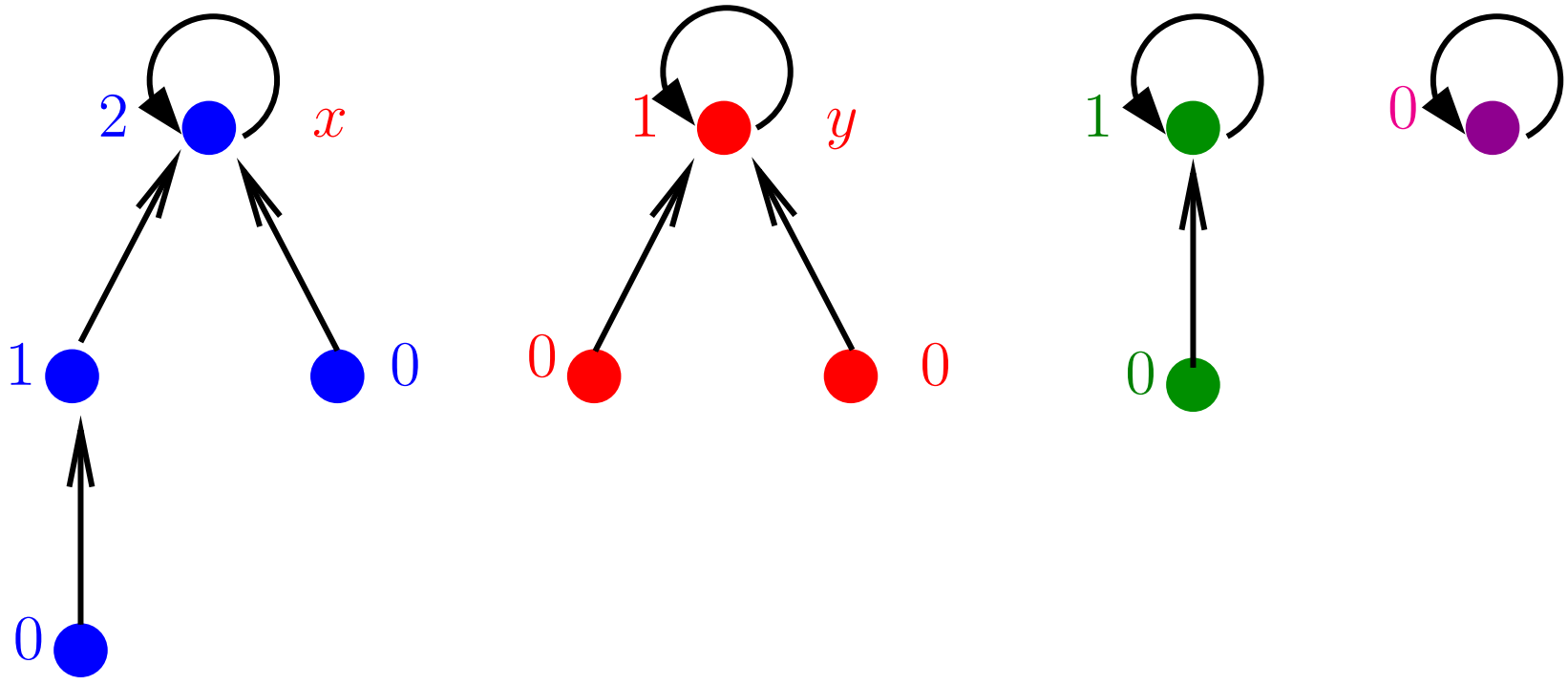
MAKESET$_0$	$\Theta(1)$
UNION$_0$	$\mathrm{O}(n)$
FINDSET$_0$	$\mathrm{O}(n)$



Custo total da seqüência:

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

Melhoramento 1: *union by rank*



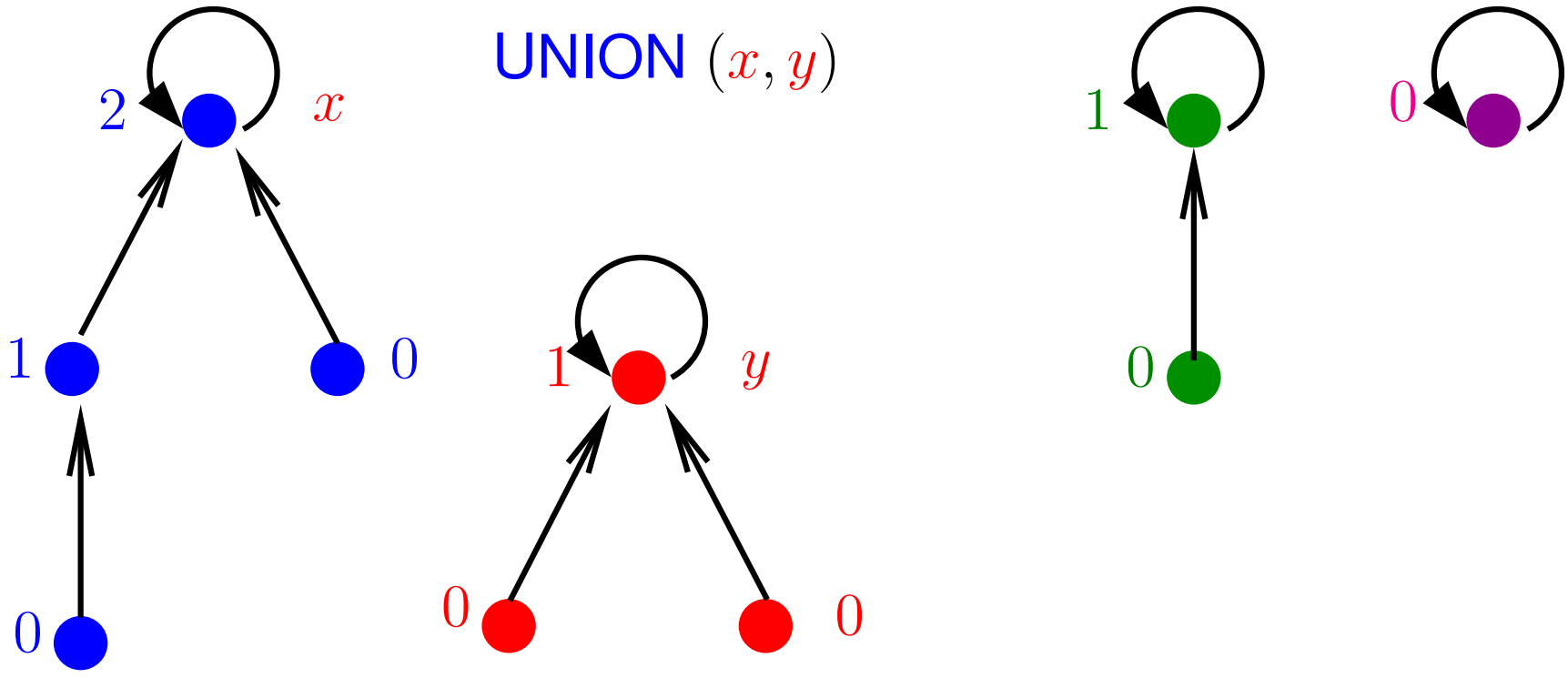
$rank[x]$ = posto do nó x

MAKESET (x)

1 $pai[x] \leftarrow x$

2 $rank[x] \leftarrow 0$

Melhoramento 1: *union by rank*



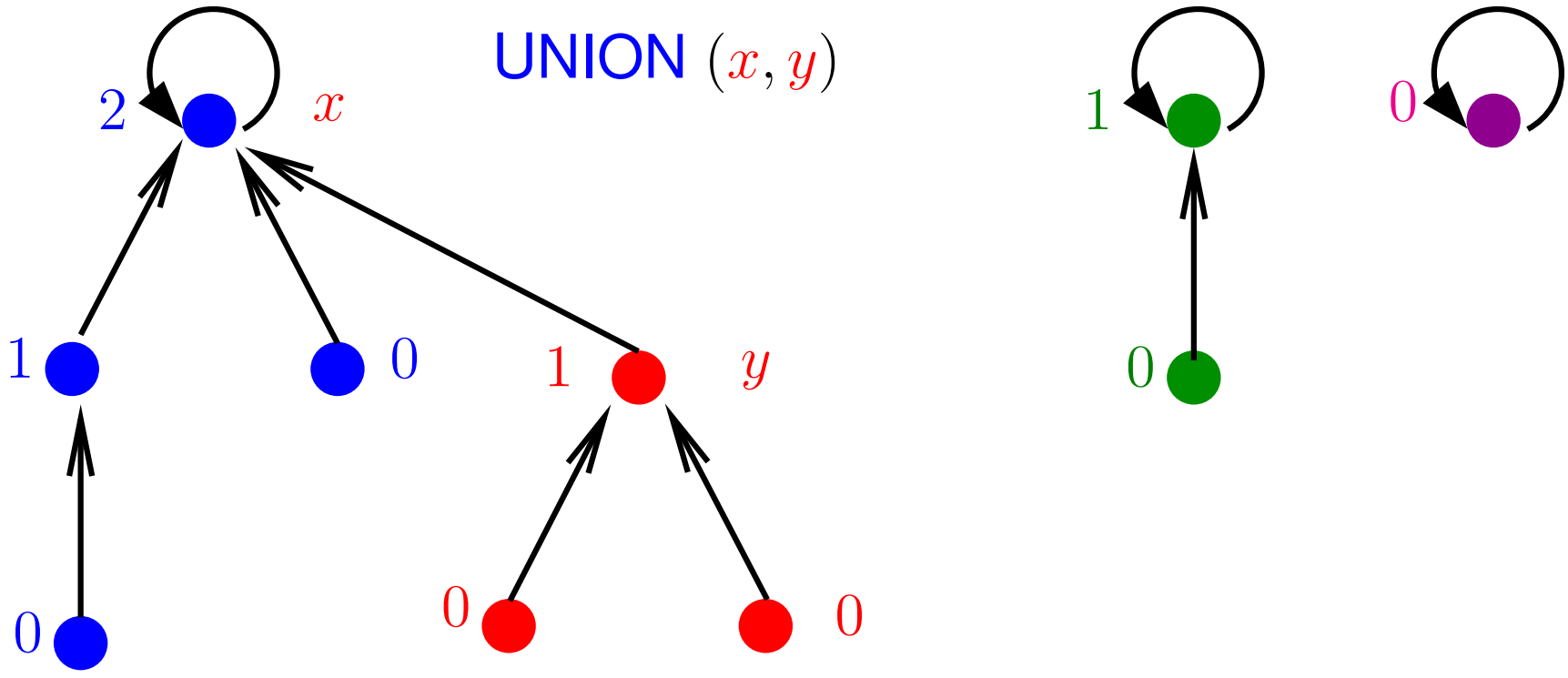
$rank[x]$ = posto do nó x

MAKESET (x)

1 $pai[x] \leftarrow x$

2 $rank[x] \leftarrow 0$

Melhoramento 1: *union by rank*



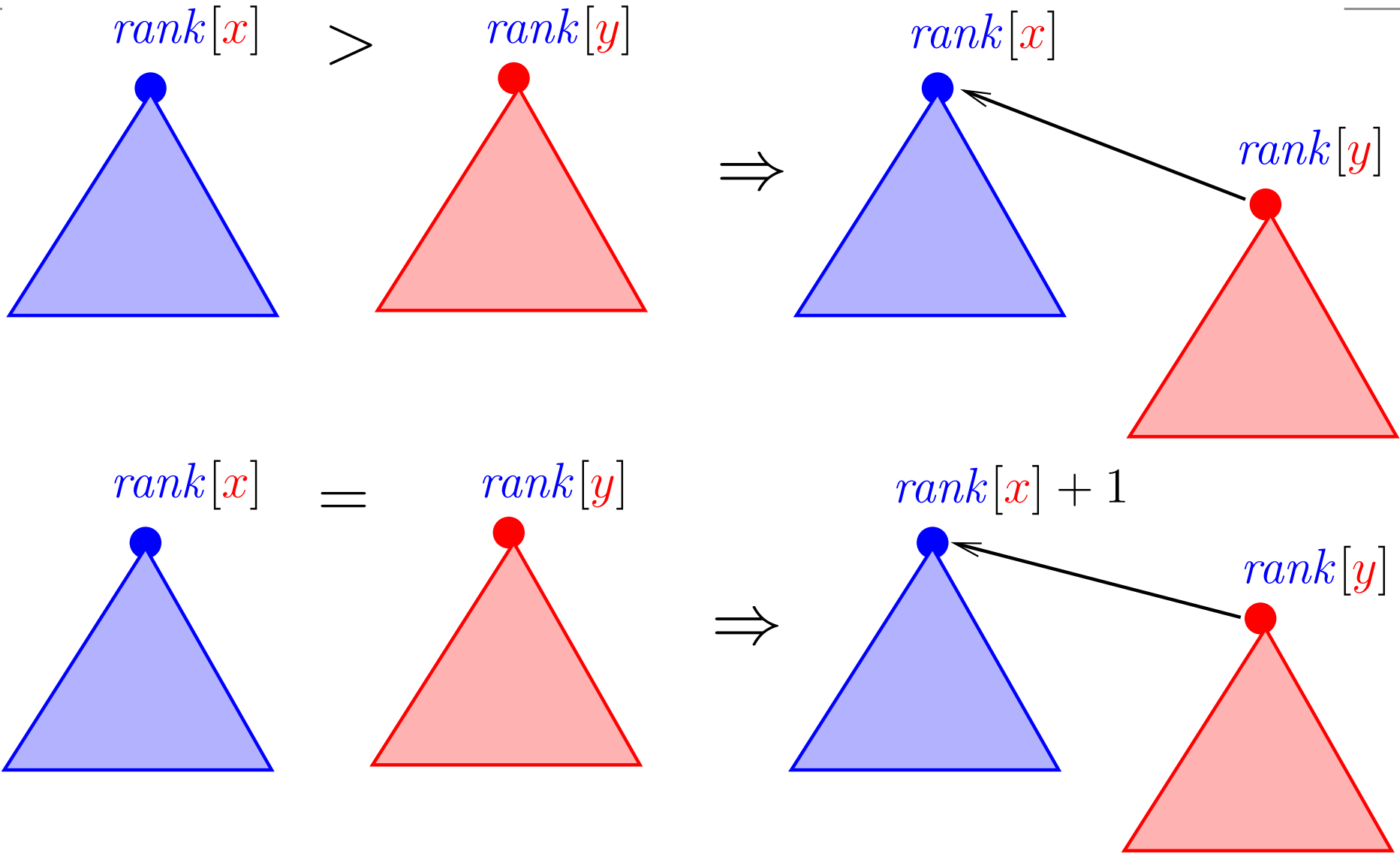
$rank[x]$ = posto do nó x

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)  ▷ com "union by rank"
1    x' ← FINDSET (x)
2    y' ← FINDSET (y)     ▷ supõe que x' ≠ y'
3    se rank[x'] > rank[y']
4          então pai[y'] ← x'
5          senão pai[x'] ← y'
6                se rank[x'] = rank[y']
7                      então rank[y'] ← rank[y'] + 1
```

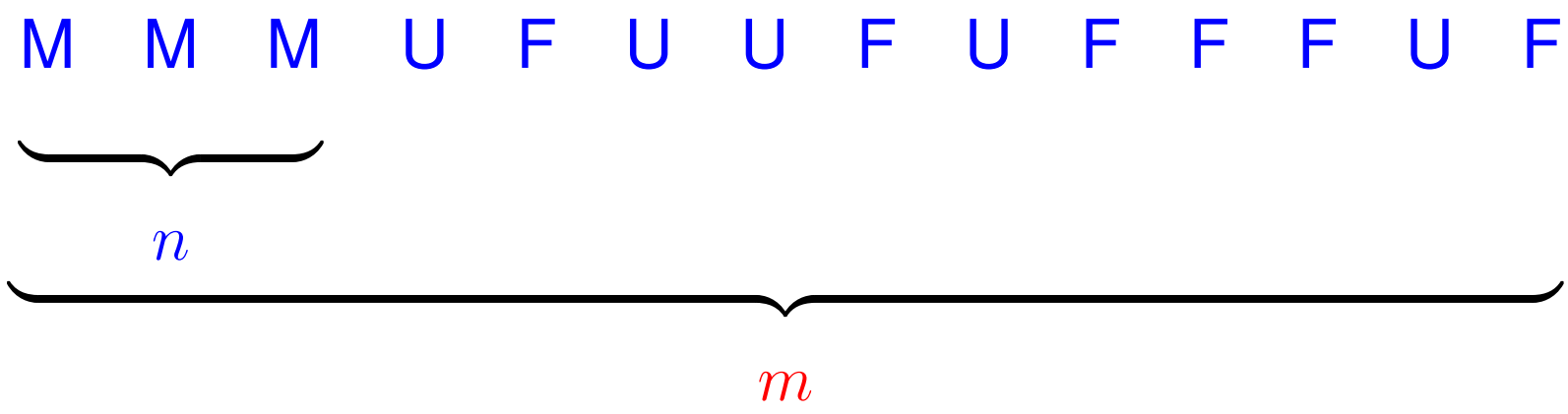
Melhoramento 1: invariantes

- $rank[x] \leq rank[pai[x]]$ 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



$altura(x) \leq \lg n$ 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: $\mathrm{O}(m \lg n)$

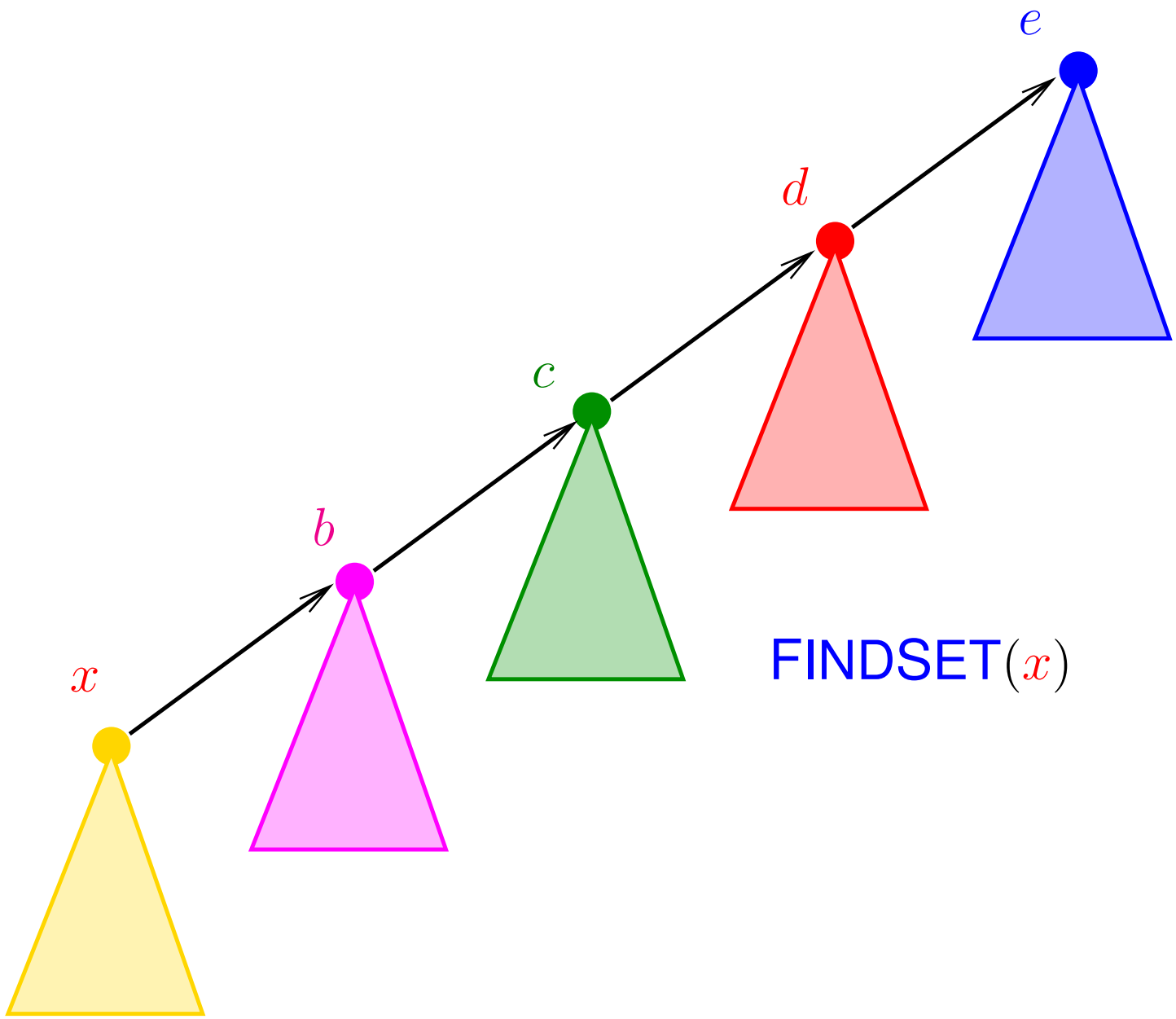
Conclusões

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

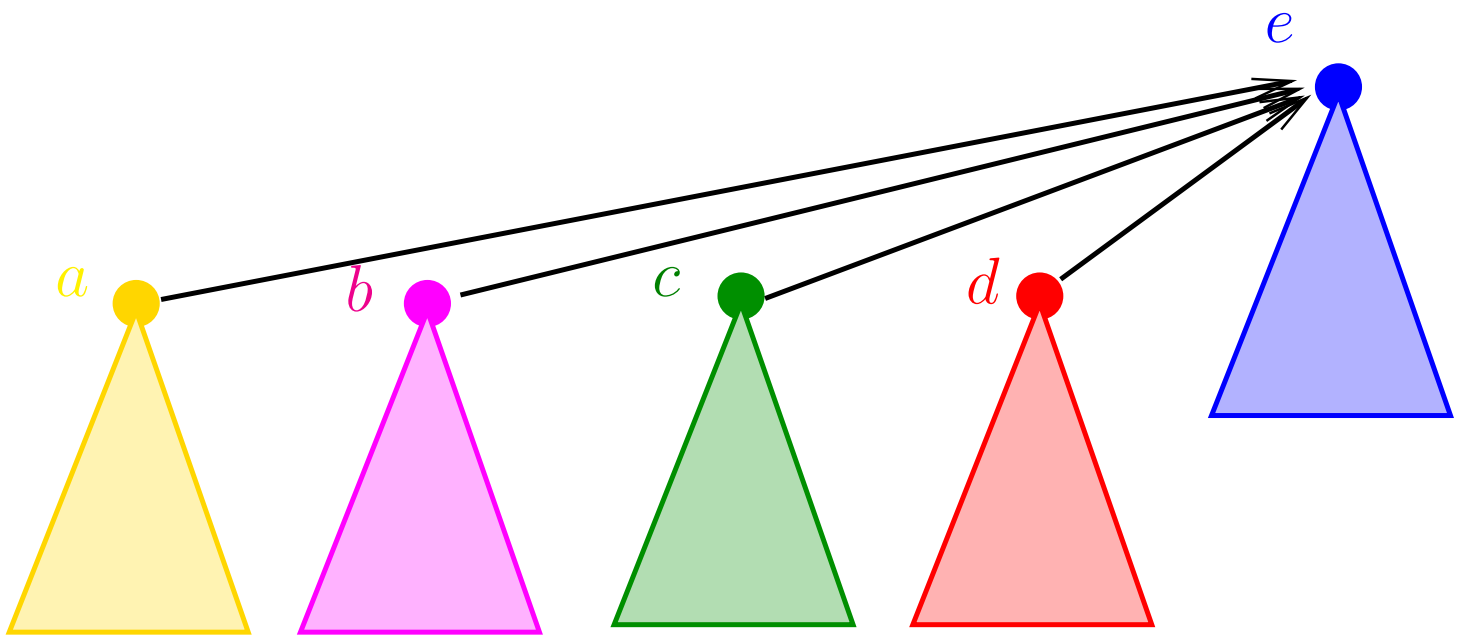
Se conjuntos disjuntos são representados através de disjoint-set forest com *union by rank*, então os algoritmos CONNECTED-COMPONENTS e MST-KRUSKAL consomem tempo $\mathrm{O}((n+m) \lg n)$.

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)  ▷ com "path compression"
1    se x ≠ pai[x]
2          então pai[x] ← 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

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

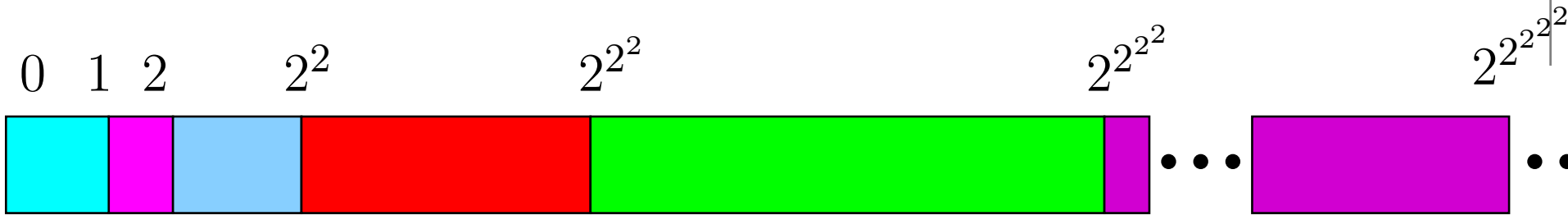
n	$\lg^* n$
1	0
2	1
3	2
4	2
5	3
⋮	⋮
15	3
16	3
⋮	⋮
65535	4
65536	4
⋮	⋮
$\underbrace{1000000000000000\cdots000000000000}_{80}$	5
⋮	⋮

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	$2^{2^2} = 4$
3	$2^{2^{2^2}} = 16$
4	$2^{2^{2^{2^2}}} = 2^{16} = 65536$
5	$2^{2^{2^{2^{2^2}}}} > \underbrace{10000000000000000 \cdots 000000000000}_{80}$
$\vdots$	$\vdots$

Blocos



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

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

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

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

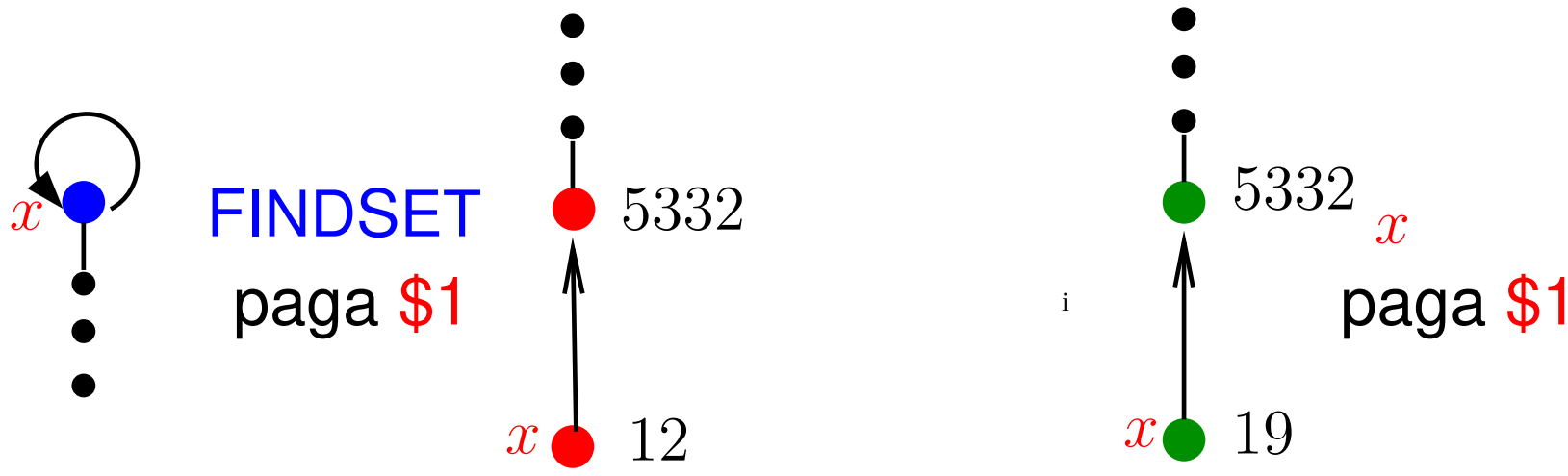
$bloco[i] = [t(i-1)+1\,..\,t(i)]$

Contabilidade

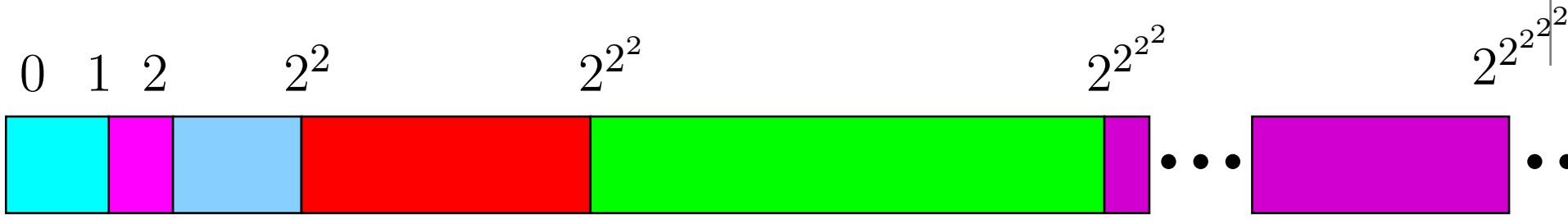
Custo de cada operação FINDSET(y) será contabilizado da seguinte maneira.

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

- $rank[x] \leq \lg n$ 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:

```
01  para i ← 1 até 16
02      faça MAKESET (x_i)
03  para i ← 1 até 15 em passos de 2
04      faça UNION (x_i, x_{i+1})
05  para i ← 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_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.)