Aula 20: Estruturas para união e busca

David Déharbe
Programa de Pós-graduação em Sistemas e Computação
Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada

Download me from `http://DavidDeharbe.github.io`.

PPgSC

Plano da aula

Introdução

Propriedades

Implementação
- Implementação por listas
- Implementação por árvores

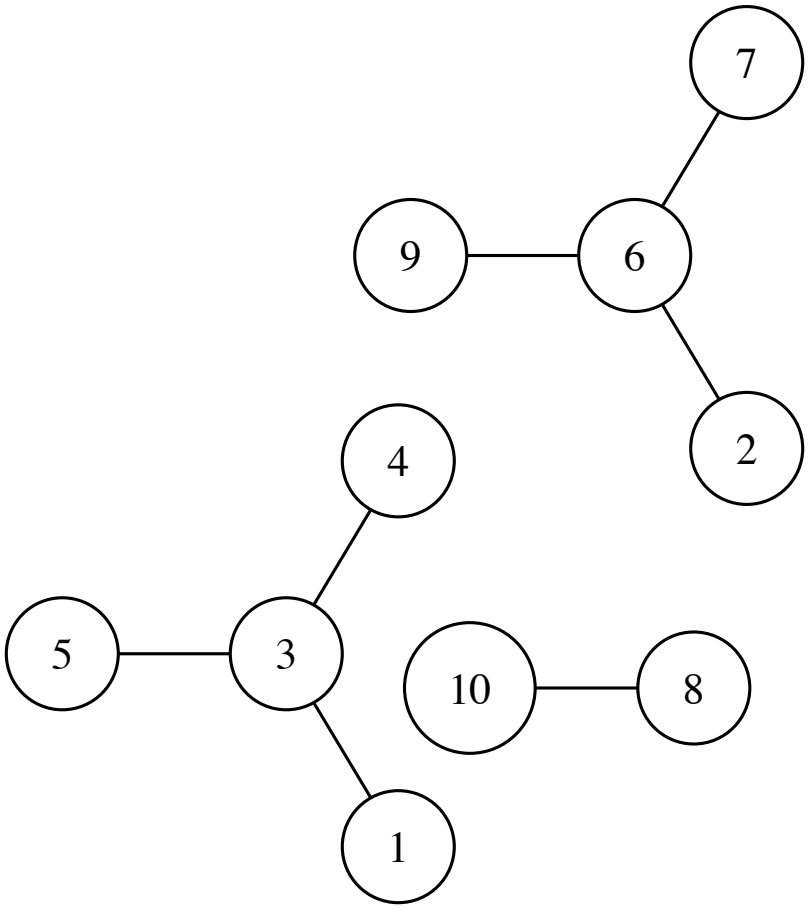
União e busca

Introdução

- Cenário de aplicação:
  - registros devem ser mantidos em grupos
  - teste se dois elementos pertencem ao mesmo grupo (busca)
  - juntar dois grupos (fusão)
  - e também:
    - quantos grupos diferentes?
    - qual o tamanho de cada grupo?
    - qual o maior grupo?
  - exemplo: componentes conexos em um grafo

PPgSC

Ilustração



- $G = (V, E)$,
- $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, e
- $E = \{(1, 3), (2, 6), (3, 4), (3, 5), (6, 7), (6, 9), (8, 10)\}$.

PPgSC

Propriedades

- Os grupos formam uma partição do conjunto dos registros
- Esta partição induz uma relação R sobre os registros:

$$a \sim b \text{ sse } grupo(A) = grupo(B)$$

- $\sim$ é uma relação de equivalência
  - reflexiva $\forall e \cdot e \sim e$
  - simétrica $\forall e, e' \cdot e \sim e' \Rightarrow e' \sim e$
  - transitiva $\forall e_0, e_1, e_2 \cdot e_0 \sim e_1 \wedge e_1 \sim e_2 \Rightarrow e_0 \sim e_2$.

PPgSC

Especificação das operações

- Criação de um novo grupo $\textsc{Make-Set}(x)$
  - registro x
  - resultado: o grupo criado
- Busca do grupo $\textsc{Find-Set}(x)$
  - registro x
  - resultado: o grupo de x
- Junção de dois grupos $\textsc{Union-Set}(x, y)$
  - registros x, y
  - resultado: o grupo com x, y, e todos os elementos que estavam com x e com y

PPgSC

Aplicação

GRAPH-SCC(G)

```
1  for v ∈ G.V
2      MAKE-SET(v)
3  for e ∈ G.E
4      UNION-SET(e.src, e.dst)
```

Aplicação

GRAPH-SCC(G)

```
1  for v ∈ G.V
2      MAKE-SET(v)
3  for e ∈ G.E
4      UNION-SET(e.src, e.dst)
```

inicial	$\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\},\{9\},\{10\}$
$(1,3)$	$\{1,3\},\{2\},\{4\},\{5\},\{6\},\{7\},\{8\},\{9\},\{10\}$
$(2,6)$	$\{1,3\},\{2,6\},\{4\},\{5\},\{7\},\{8\},\{9\},\{10\}$
$(3,4)$	$\{1,3,4\},\{2,6\},\{5\},\{7\},\{8\},\{9\},\{10\}$
$(3,5)$	$\{1,3,4,5\},\{2,6\},\{7\},\{8\},\{9\},\{10\}$
$(6,7)$	$\{1,3,4,5\},\{2,6,7\},\{8\},\{9\},\{10\}$

PPgSC

Estruturas de dados

listas encadeadas busca mais eficiente

árvores união mais eficiente

Princípio comum:

- cada grupo elege um registro representante
- o representante identifica o grupo

Estado

Implementação por lista

Para cada registro x

- $x.rep$ é o grupo de x
- $x.next$ é o próximo elemento do grupo

Criação

Implementação por lista

MAKE-SET(x)

1 $x.rep = x$

2 $x.next = x$

FIND-SET(x)

1 **return** $x.rep$

União

Implementação por lista

```
UNION-SET(x1, x2)
 1   r1 = FIND-SET(x1)
 2   r2 = FIND-SET(x2)
 3   tmp = r1.next
 4   r1.next = r2
 5   x = r2
 6   repeat
 7         x.rep = r1
 8         x = x.next
 9   until x.next == r2
10   x.rep = r1
11   x.next = tmp
12   return r1
```

União

Implementação por lista

UNION-SET(x_1, x_2)

```
 1  r1 = FIND-SET(x1)
 2  r2 = FIND-SET(x2)
 3  tmp = r1.next
 4  r1.next = r2
 5  x = r2
 6  repeat
 7        x.rep = r1
 8        x = x.next
 9  until x.next == r2
10  x.rep = r1
11  x.next = tmp
12  return r1
```

Complexidade: $\Theta(|grupo(x_2)|)$

PPgSC

Ilustração da união

Implementação por lista

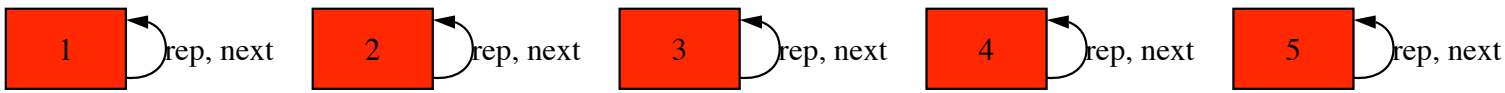
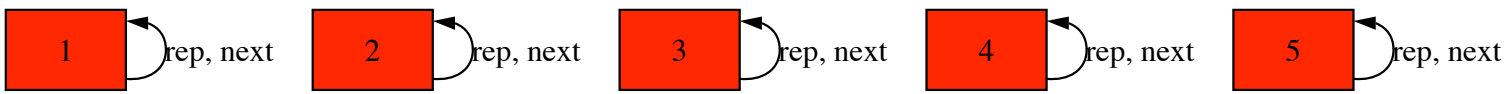


Ilustração da união

Implementação por lista



$$1 \equiv 2$$

Ilustração da união

Implementação por lista

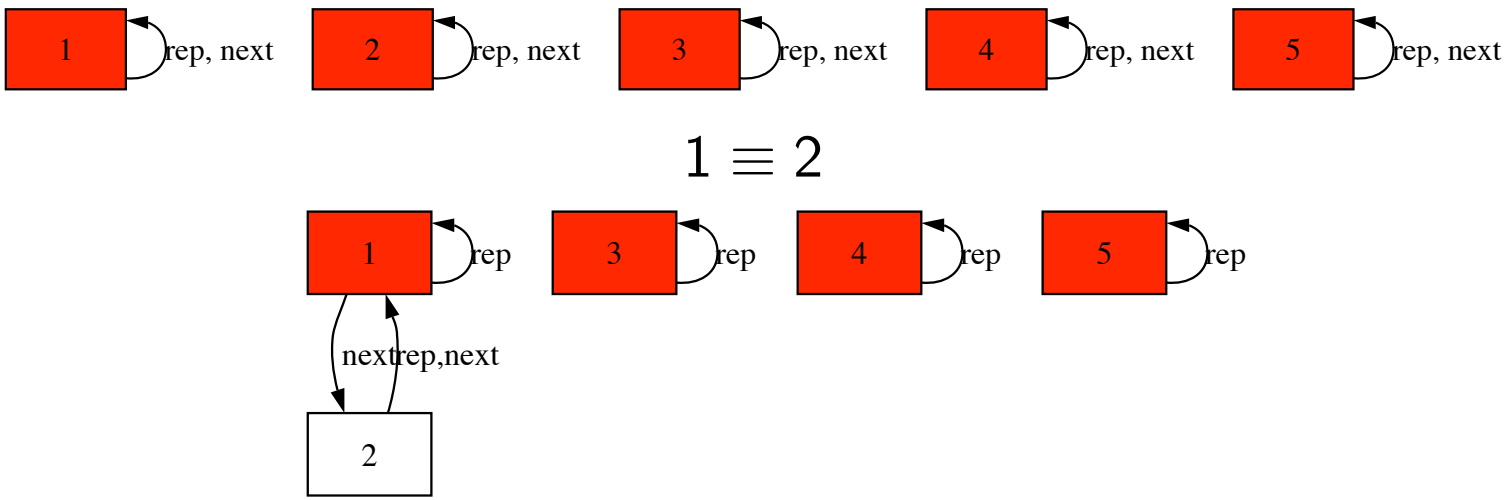


Ilustração da união

Implementação por lista

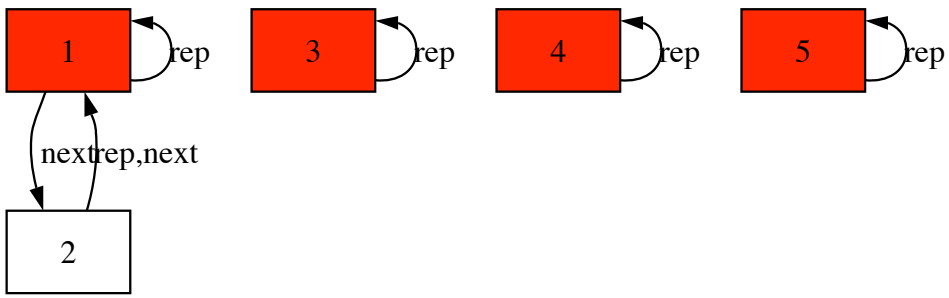
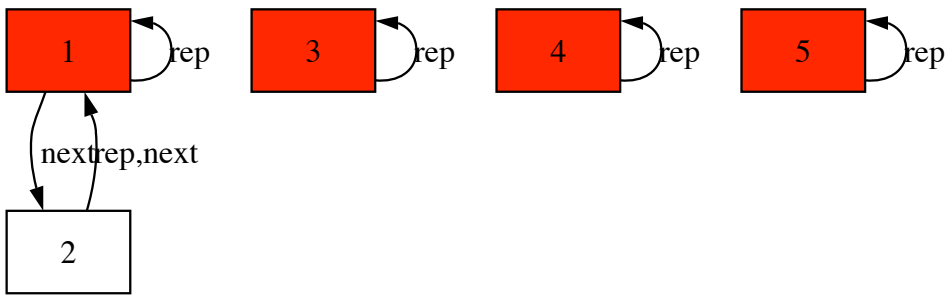


Ilustração da união

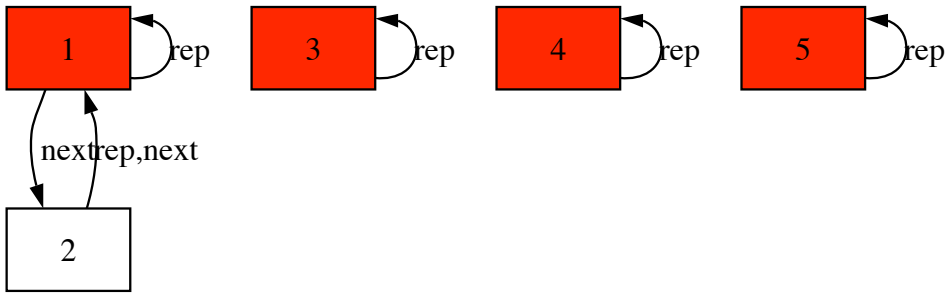
Implementação por lista



$$3 \equiv 4$$

Ilustração da união

Implementação por lista



$3 \equiv 4$

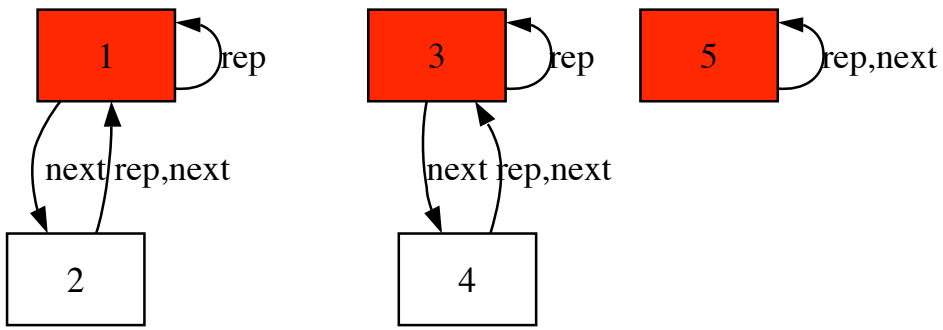


Ilustração da união

Implementação por lista

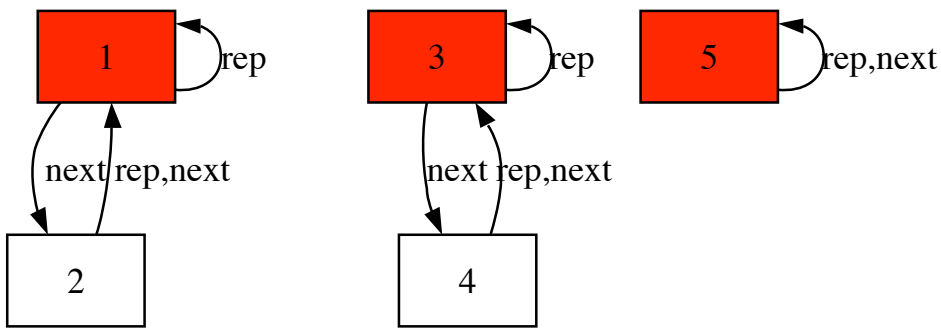
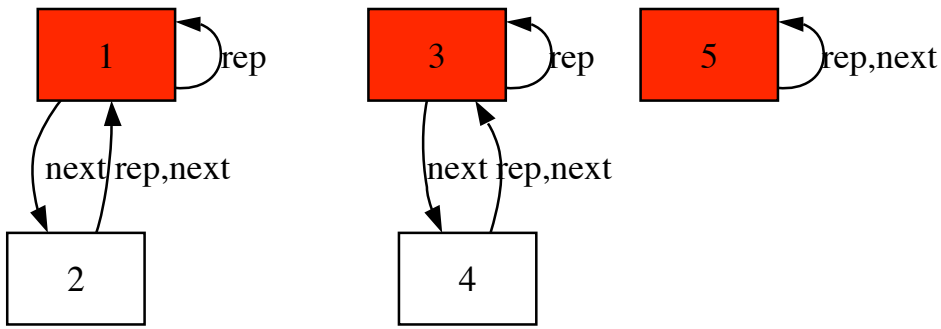


Ilustração da união

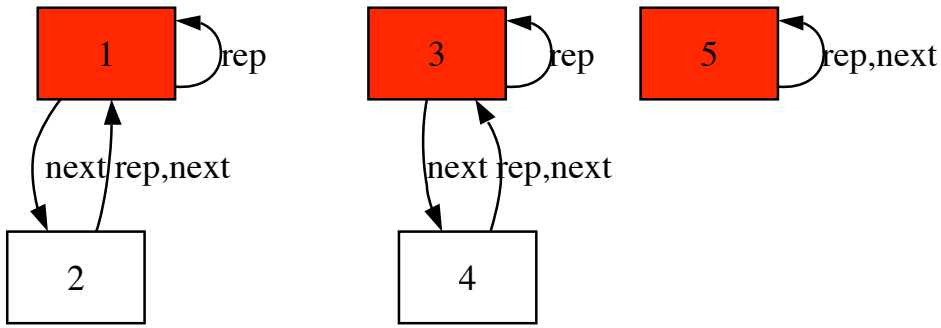
Implementação por lista



$5 \equiv 2$

Ilustração da união

Implementação por lista



$5 \equiv 2$

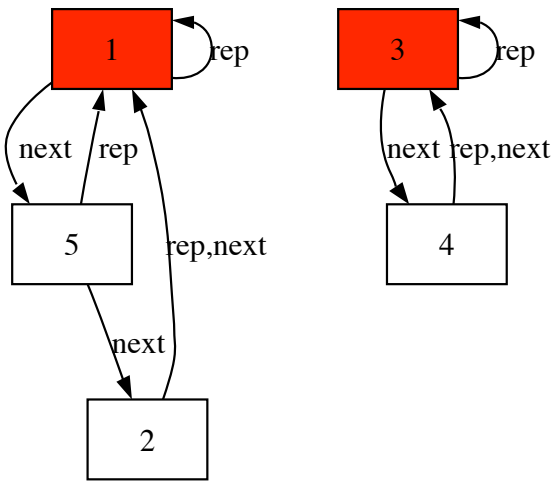


Ilustração da união

Implementação por lista

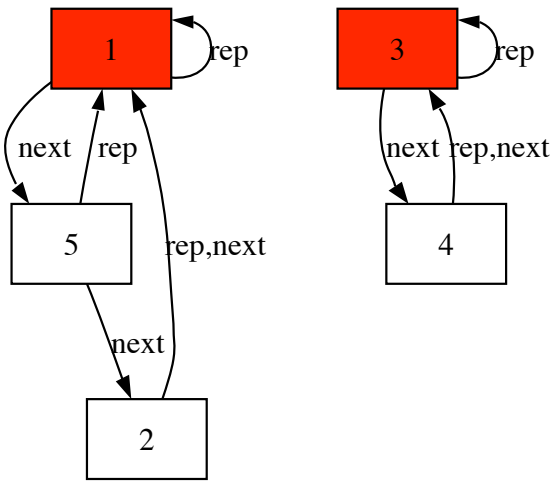
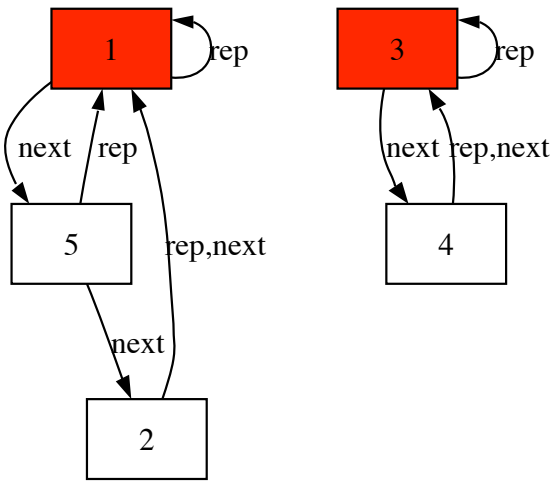


Ilustração da união

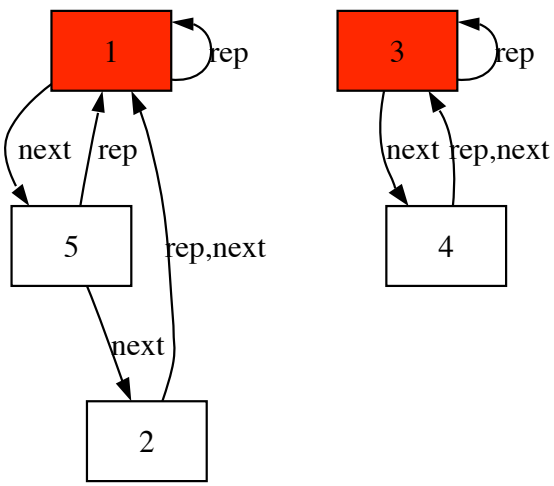
Implementação por lista



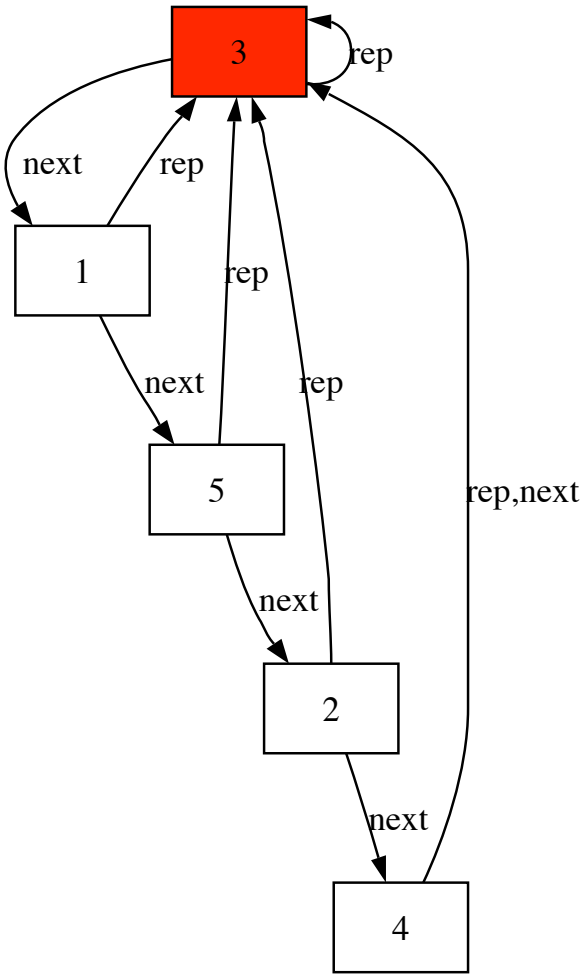
$$1 \equiv 3$$

Ilustração da união

Implementação por lista



$1 \equiv 3$



União por tamanho

- a solução apresentada pode ser otimizada facilmente
- incluir o menor grupo no maior
- $\Theta(\min(|grupo(x_1)|, |grupo(x_2)|))$
- realização:
  - estado: $x.rank$ tamanho do grupo (só precisa ser atualizado quando $x.rep = x$)
  - união:
    1. troque r_1 e r_2 quando $r_1.rank < r_2.rank$
    2. $r_1.rank = r_1.rank + r_2.rank$

Estado

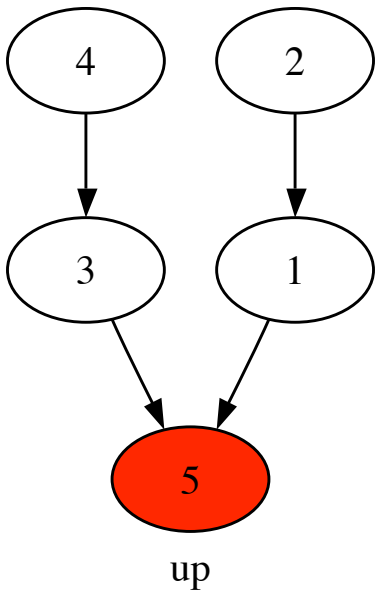
Implementação por árvores

- $x.up$ é um registro do mesmo grupo que x
- r é representante de um grupo quando $r.up = r$
- o fecho transitivo da relação entre nós induzida por up relaciona cada registro com seu representante

Estado

Implementação por árvores

- $x.up$ é um registro do mesmo grupo que x
- r é representante de um grupo quando $r.up = r$
- o fecho transitivo da relação entre nós induzida por up relaciona cada registro com seu representante



Criação

Implementação por árvores

MAKE-SET(x)

1 $x.up = x$

FIND-SET(x)

```
1  while x ≠ x.up
2        x = x.up
3  return x
```

União

Implementação por árvores

UNION-SET(x_1, x_2)

1 $r_1 = \text{FIND-SET}(x_1)$
2 $r_2 = \text{FIND-SET}(x_2)$
3 $r_2.up = r_1$
4 **return** r_1

União

Implementação por árvores

UNION-SET(x_1, x_2)

1 $r_1 = \text{FIND-SET}(x_1)$
2 $r_2 = \text{FIND-SET}(x_2)$
3 $r_2.up = r_1$
4 **return** r_1

Complexidade?

Ilustração da união

Implementação por árvores

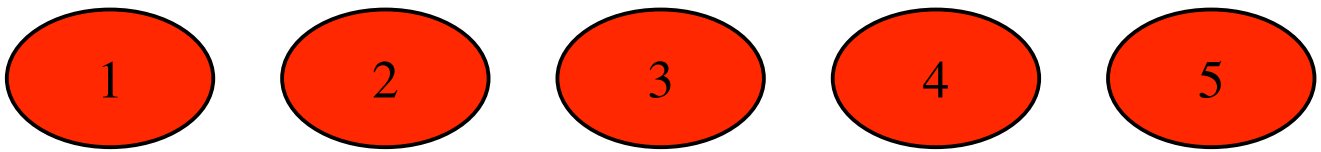
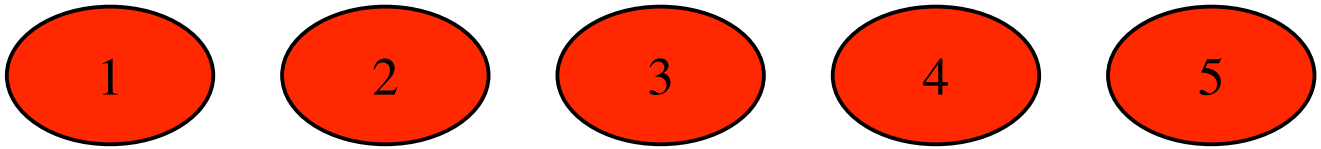


Ilustração da união

Implementação por árvores



$1 \equiv 2$

Ilustração da união

Implementação por árvores

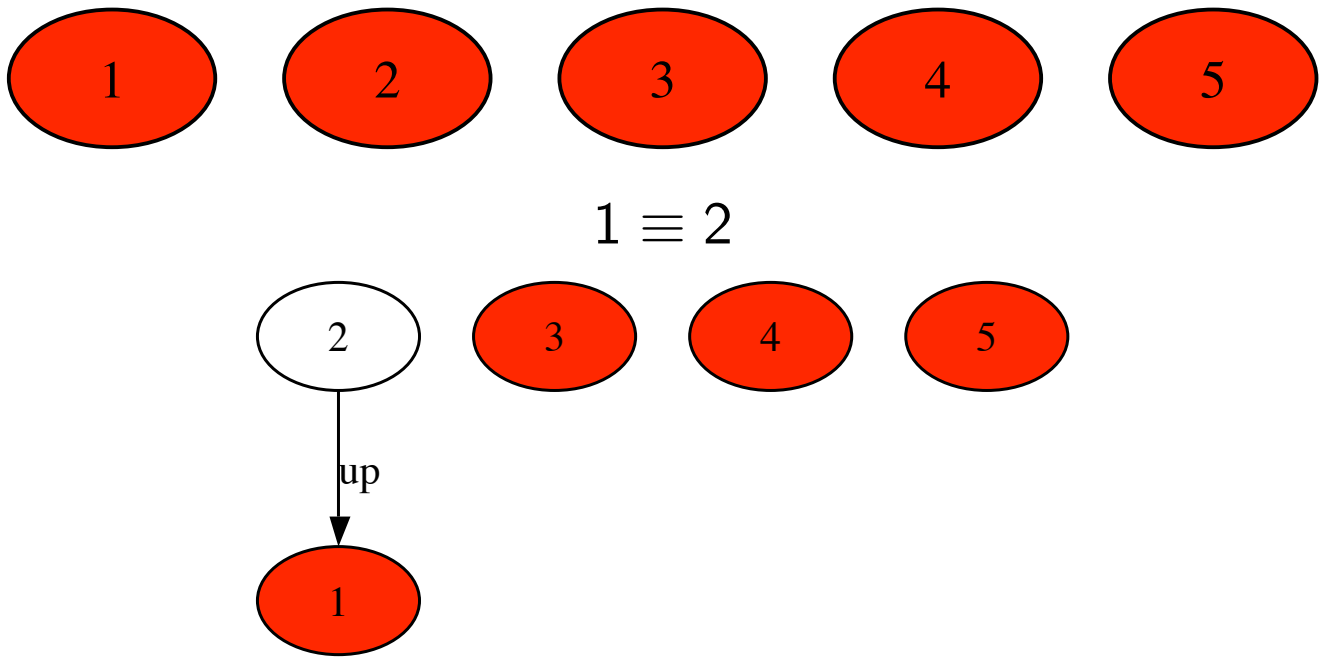


Ilustração da união

Implementação por árvores

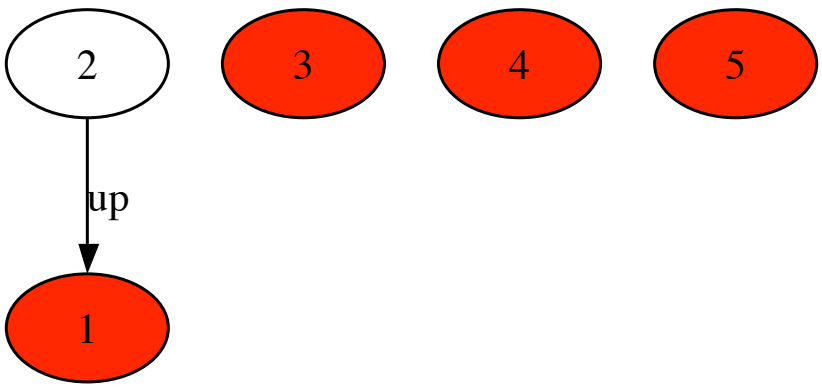
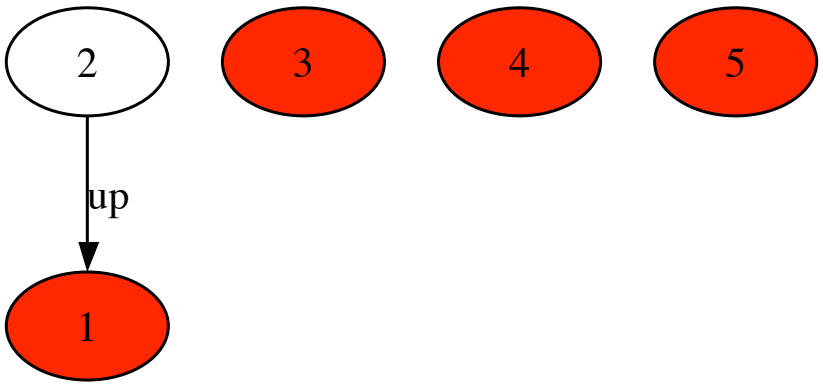


Ilustração da união

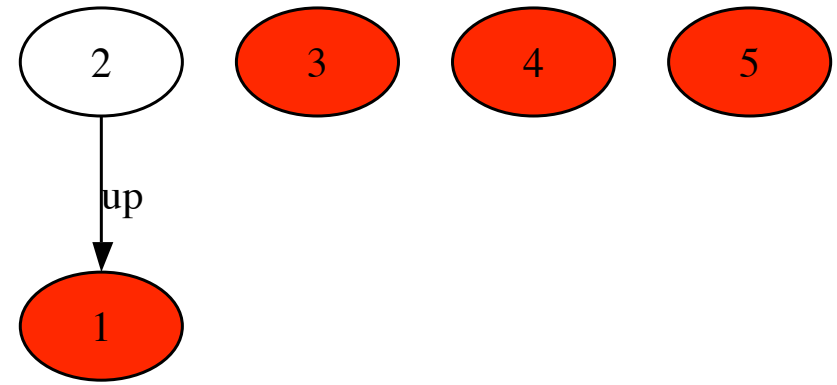
Implementação por árvores



$3 \equiv 4$

Ilustração da união

Implementação por árvores



$3 \equiv 4$

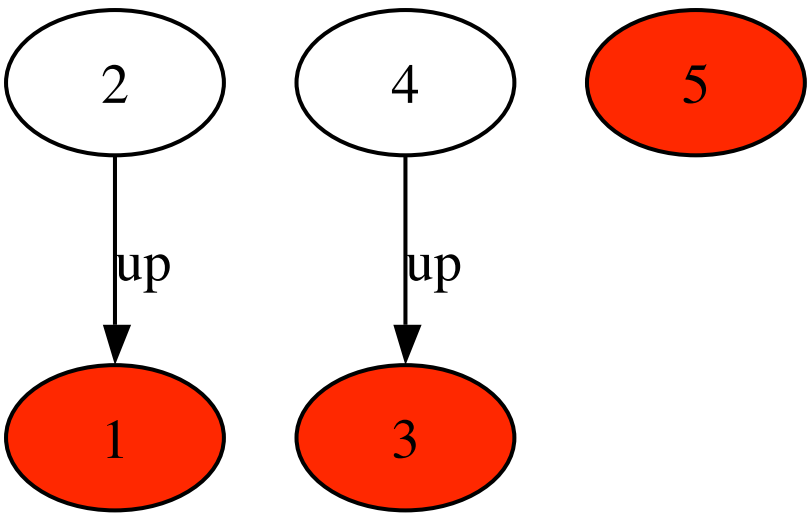


Ilustração da união

Implementação por árvores

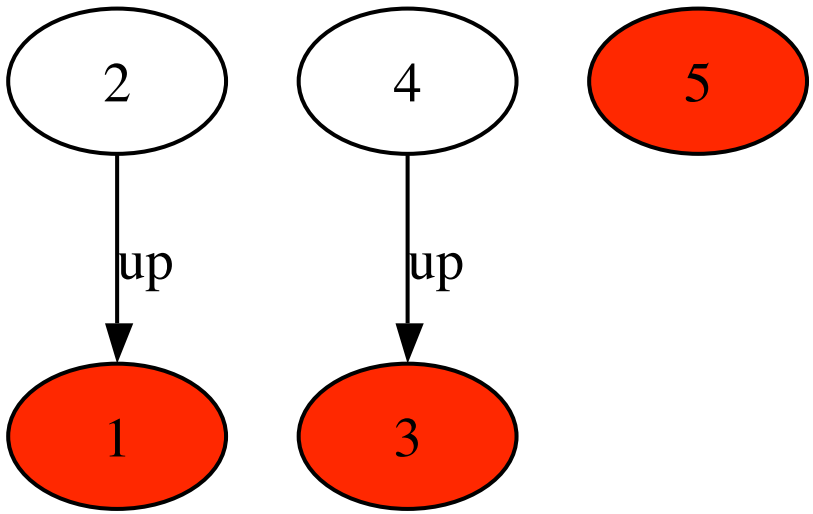
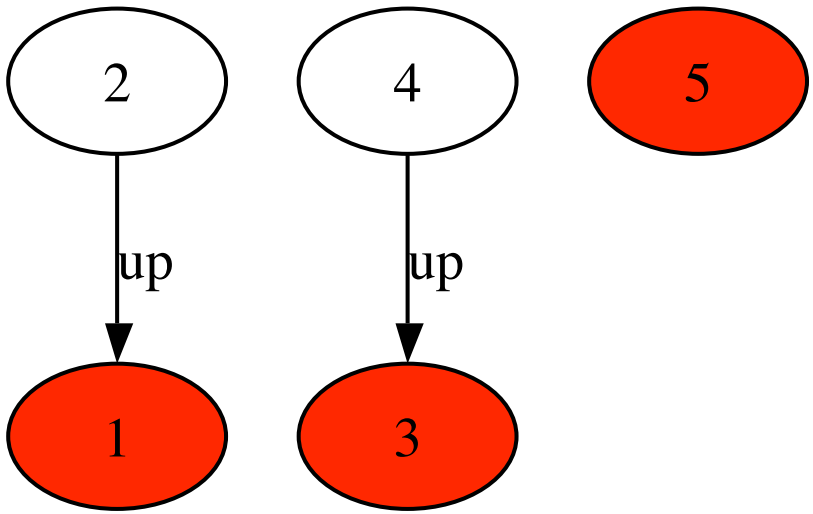


Ilustração da união

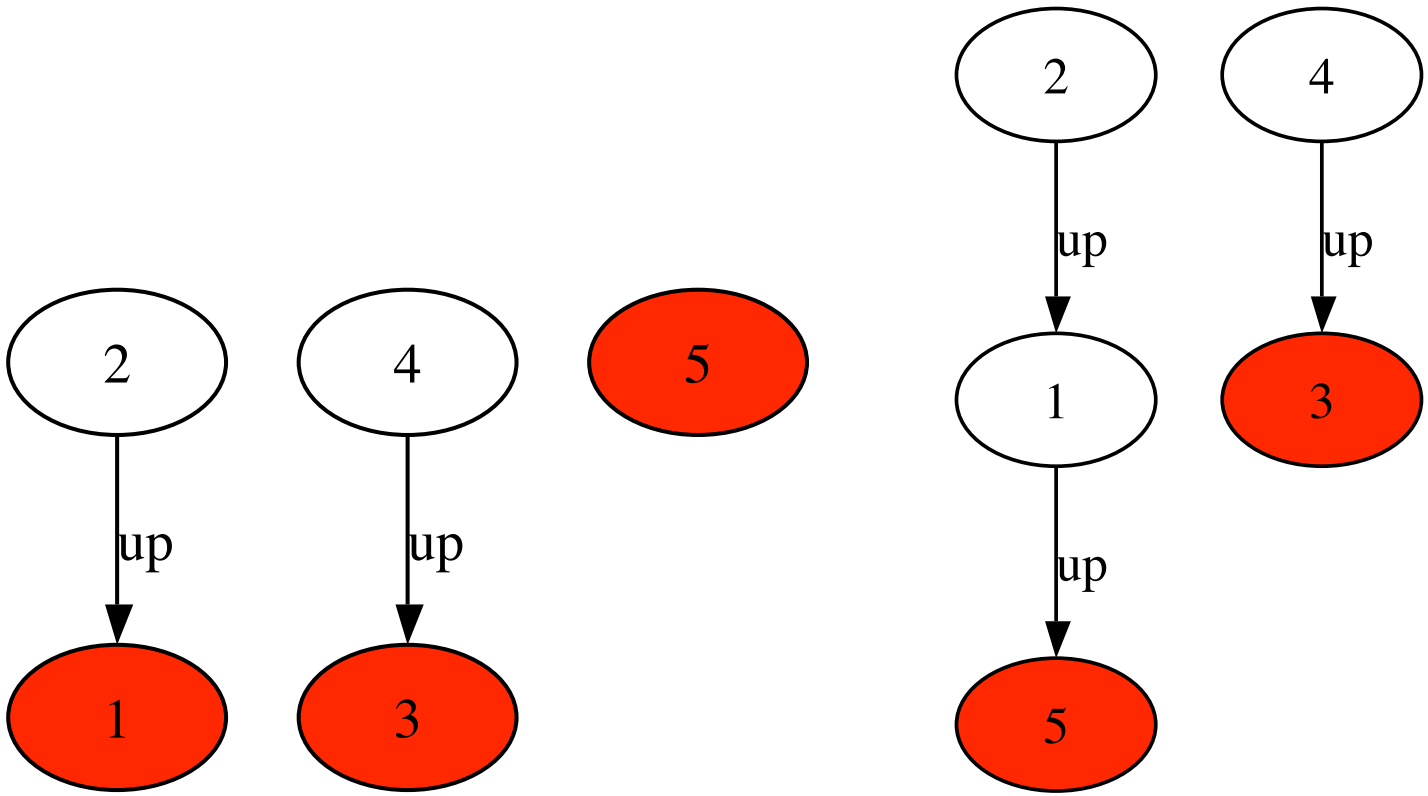
Implementação por árvores



$5 \equiv 2$

Ilustração da união

Implementação por árvores



$5 \equiv 2$

Ilustração da união

Implementação por árvores

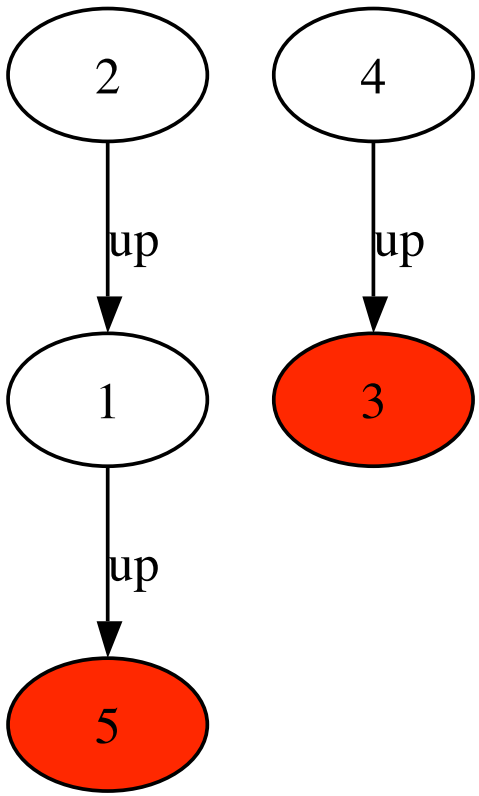
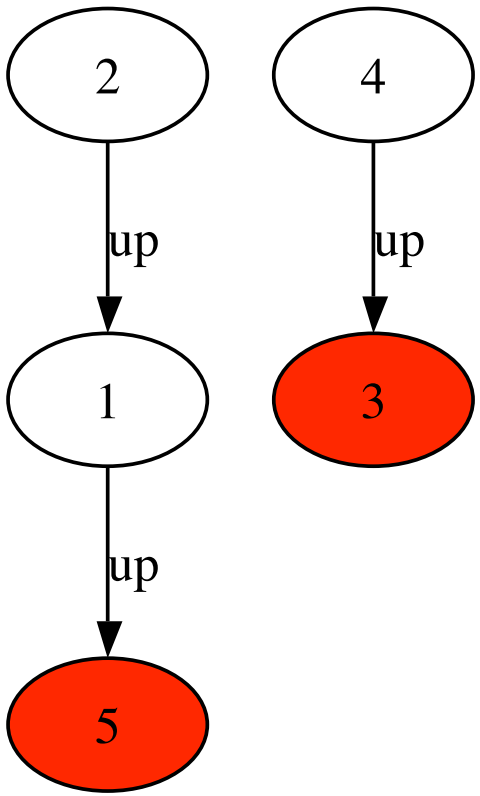


Ilustração da união

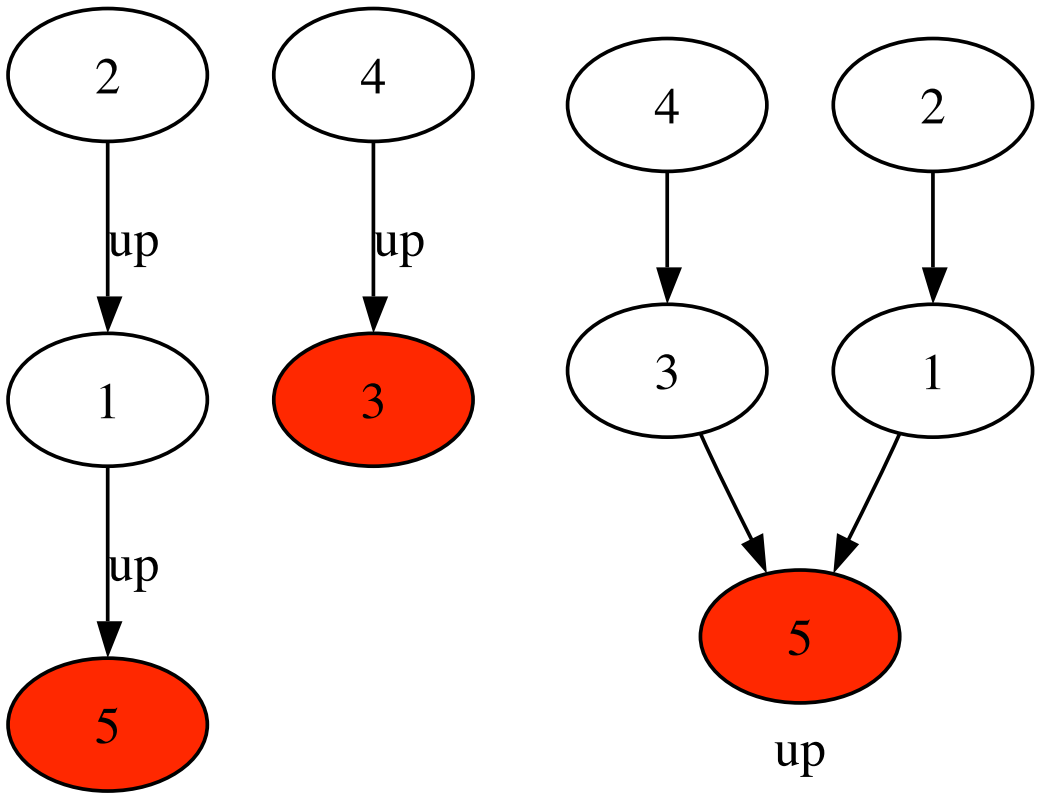
Implementação por árvores



$1 \equiv 3$

Ilustração da união

Implementação por árvores



$1 \equiv 3$

União por tamanho

- UNION-SET(x_1, x_2) aumenta a altura de todos os nós do grupo de x_2
- o custo da operação de busca depende da altura
- o custo da operação de fusão depende da altura
- para minimizar o custo ao longo da execução, sempre escolher como grupo destino o que era inicialmente maior
- mesma técnica que para as listas

PPgSC

União por tamanho

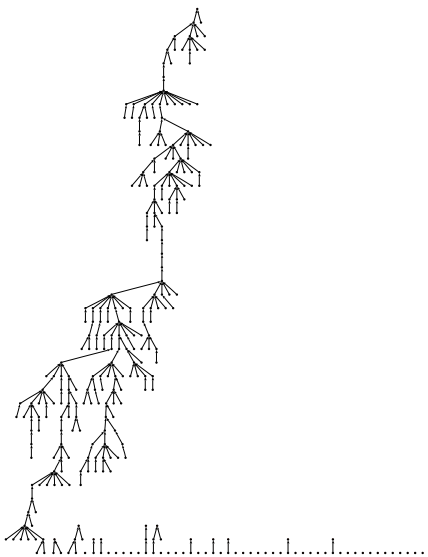
Estudo experimental

- é gerado aleatoriamente um grafo de 300 vertices e 300 arestas
- são impressas as árvores formadas pelos grupos depois de aplicar GRAPH-SCC
  - sem união por tamanho
  - com união por tamanho

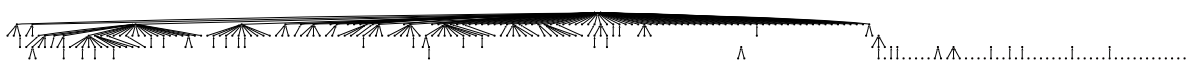
União por tamanho

Estudo experimental

sem



com



Compressão de caminhos

- a busca é uma operação essencial
- o custo depende da altura do valor procurado
- como diminuir a altura de forma eficiente?

Compressão de caminhos

- a busca é uma operação essencial
- o custo depende da altura do valor procurado
- como diminuir a altura de forma eficiente?
- aproveitar a busca para encurtar o caminho até o representante
  - no final da busca, atribuir *up* com o representante do grupo
  - durante a busca, encurtar o caminho: $x.up = x.up.up$
- pode ser combinado com a união por tamanho

PPgSC

- compressão completa:

```
FIND-SET(x)
1  tmp = x
2  while x ≠ x.up
3        x = x.up
4  while tmp ≠ x
5        y = tmp.up
6        tmp.up = x
7        tmp = y
8  return x
```

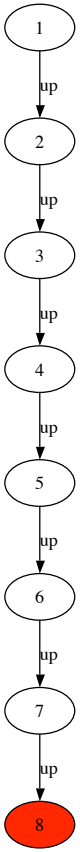
- compressão por divisão:

FIND-SET(x)

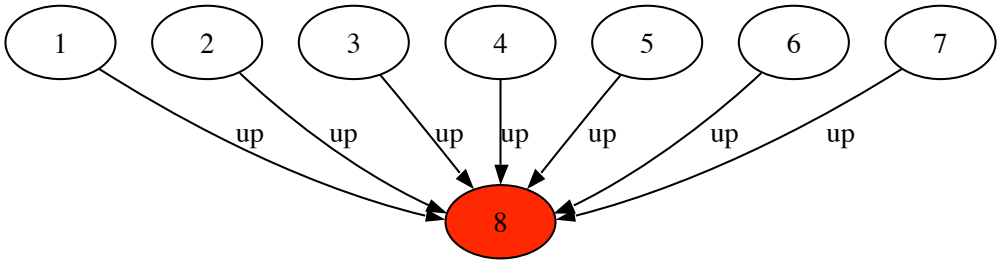
```
1  while x ≠ x.up
2        x.up = x.up.up
3        x = x.up
4  return x
```

Ilustração

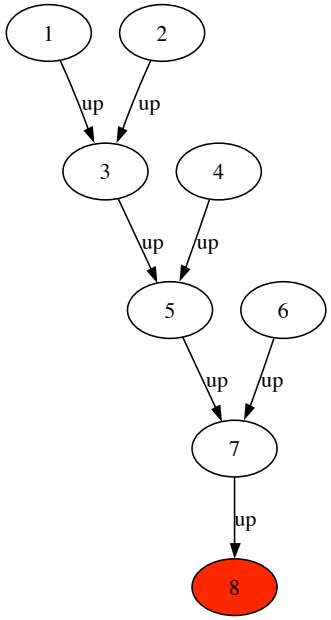
Compressão de caminhos



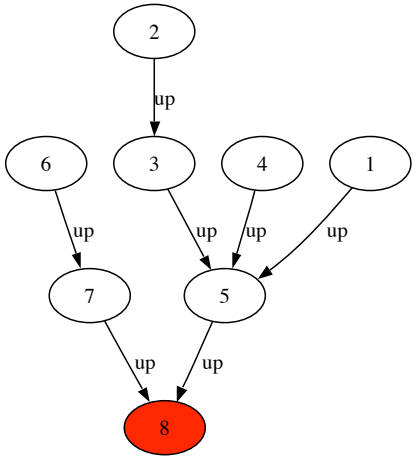
Estado inicial



Após busca de 1, com compressão completa



Após busca de 1, com compressão por divisão



Após duas buscas de 1, com compressão por divisão

Estudo experimental

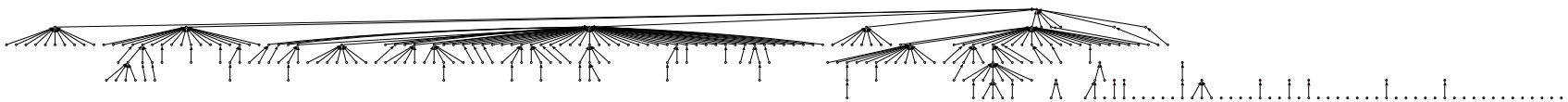
- é gerado aleatoriamente um grafo de 300 vertices e 300 arestas
- são impressas as árvores formadas pelos grupos depois de aplicar GRAPH-SCC
  - compressão completa
  - compressão por divisão
  - sem compressão
- são impressas as árvores formadas pelos grupos depois de aplicar GRAPH-SCC e 300 buscas aleatórias
  - compressão completa
  - compressão por divisão
  - união por tamanho

PPgSC

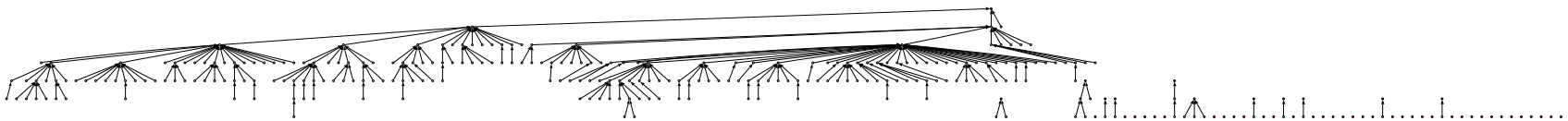
Estudo experimental

Após aplicação de GRAPH-SCC

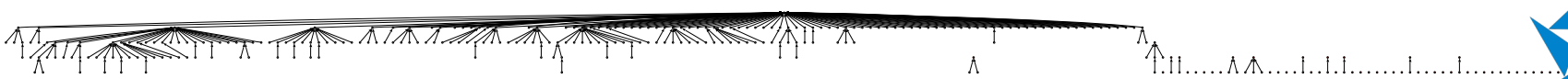
Compressão completa:



Compressão por divisão:



União por tamanho, sem compressão:

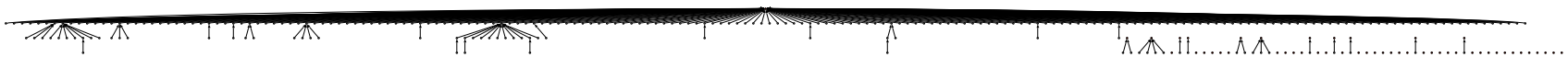


PPgSC

Estudo experimental

Após aplicação de GRAPH-SCC e 300 buscas

Compressão completa:



Compressão por divisão:

União por tamanho, sem compressão:

PPgSC