Conjuntos Union-Find

ESTRUTURAS DE DADOS

João Emanuel Mendonça Apóstolo
Lanna Luara Novaes Silva
Lavínia Louise Rosa Santos
Maria Eduarda Pires Possari dos Santos

Introdução

Conjuntos Disjuntos

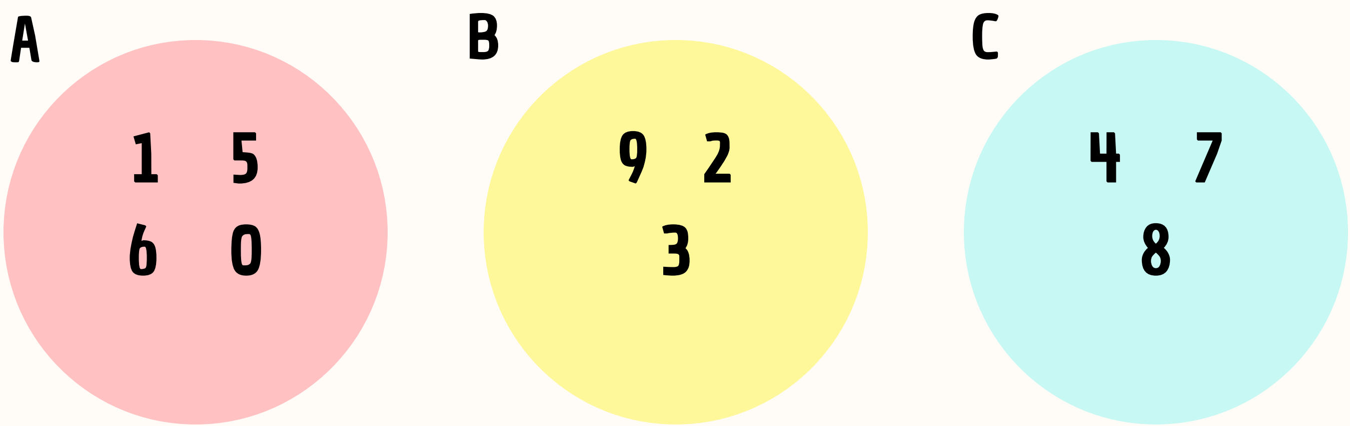
01 DEFINIÇÃO DE CONJUNTOS

Um conjunto é uma reunião de elementos

02 CONJUNTOS DISJUNTOS

É a relação entre conjuntos que não possuem elementos em comum

03 REPRESENTAÇÃO DE CONJUNTOS DISJUNTOS



$A \cap B = \emptyset$

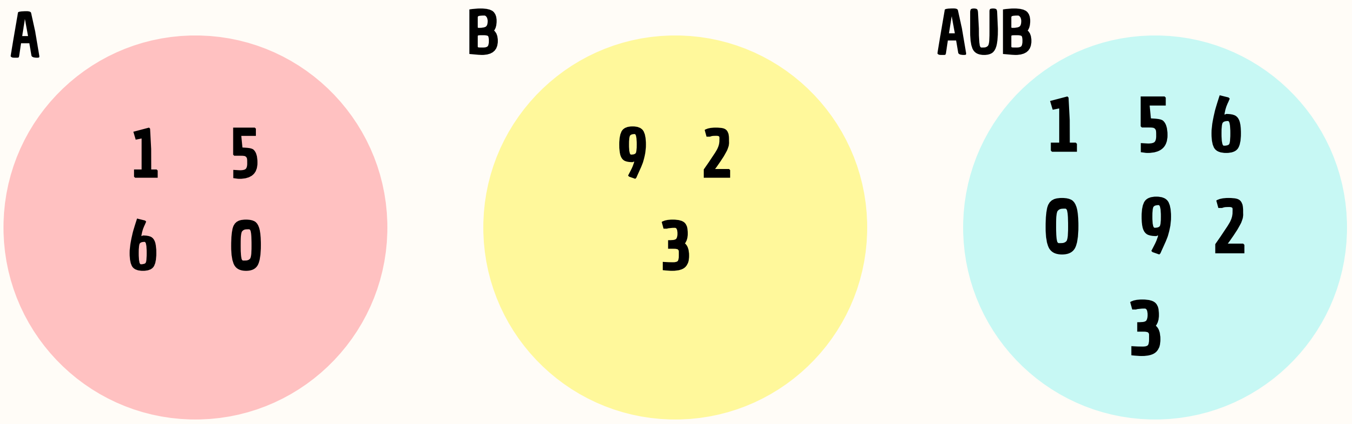
$A \cap C = \emptyset$

$B \cap C = \emptyset$

Questionamentos básicos

01 COMO UNIR DOIS CONJUNTOS DISJUNTOS?

02 COMO ENCONTRAR O CONJUNTO QUE DETERMINADO ELEMENTO FAZ PARTE?



Principais Características

01

OBJETIVO

Manter uma coleção de conjuntos disjuntos, que se modificam ao longo do tempo.

02

REPRESENTAÇÃO

Cada conjunto é identificado por seu representante.

03

OPERAÇÕES

A únicas operações permitida são a união de conjuntos e a busca de um elemento. Portanto, remover elementos de um conjunto ou partir um conjunto em dois não é permitido.

Vantagens

01

SIMPLICIDADE

No geral é uma estrutura simples de se implementar em virtude do seu estrito número de operações

02

UNIÃO E BUSCA EFICIENTES

Em média, a complexidade das operações de união e de busca é quase constante, o que permite que a sua eficiência não seja significativamente reduzida ao se utilizar um maior número de dados

03

AMPLA GAMA DE APLICAÇÕES

Essa estrutura pode ser aplicada nos mais diversos casos relacionados a conjuntos e principalmente em grafos, sendo utilizada em um grande número de algoritmos e problemas relacionas a eles

Desvantagens

01

USO DE MEMÓRIA

Como cada elemento em um conjunto disjunto é representado por um nó, para um grande número de dados a estrutura pode consumir uma significativa quantidade de memória

02

OPERAÇÕES LIMITADAS

Outras estruturas de dados podem ser mais adequadas para uso em problemas que exigem uma manipulação maior de dados, já que os conjuntos union find suportam apenas a união de conjuntos e busca de elementos

03

COMPLEXIDADE EM CASOS ADVERSOS

Com a implementação dessa estrutura utilizando árvore é possível que ela acabe ficando desbalanceada após serem realizadas uniões sucessivas, e o uso de otimizações seja recomendado para um melhor desempenho e eficiência da estrutura

Descrição

Três Operações Básicas

01

INICIALIZAÇÃO

No início, cada elemento é um conjunto único de si próprio.

02

BUSCA

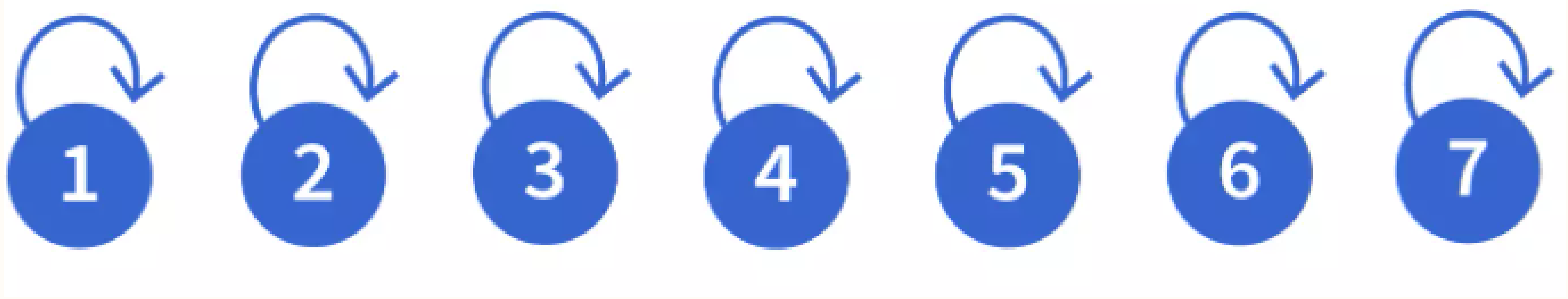
Ela define a busca de um elemento específico de um determinado conjunto. Retorna o representante do conjunto.

03

UNIÃO

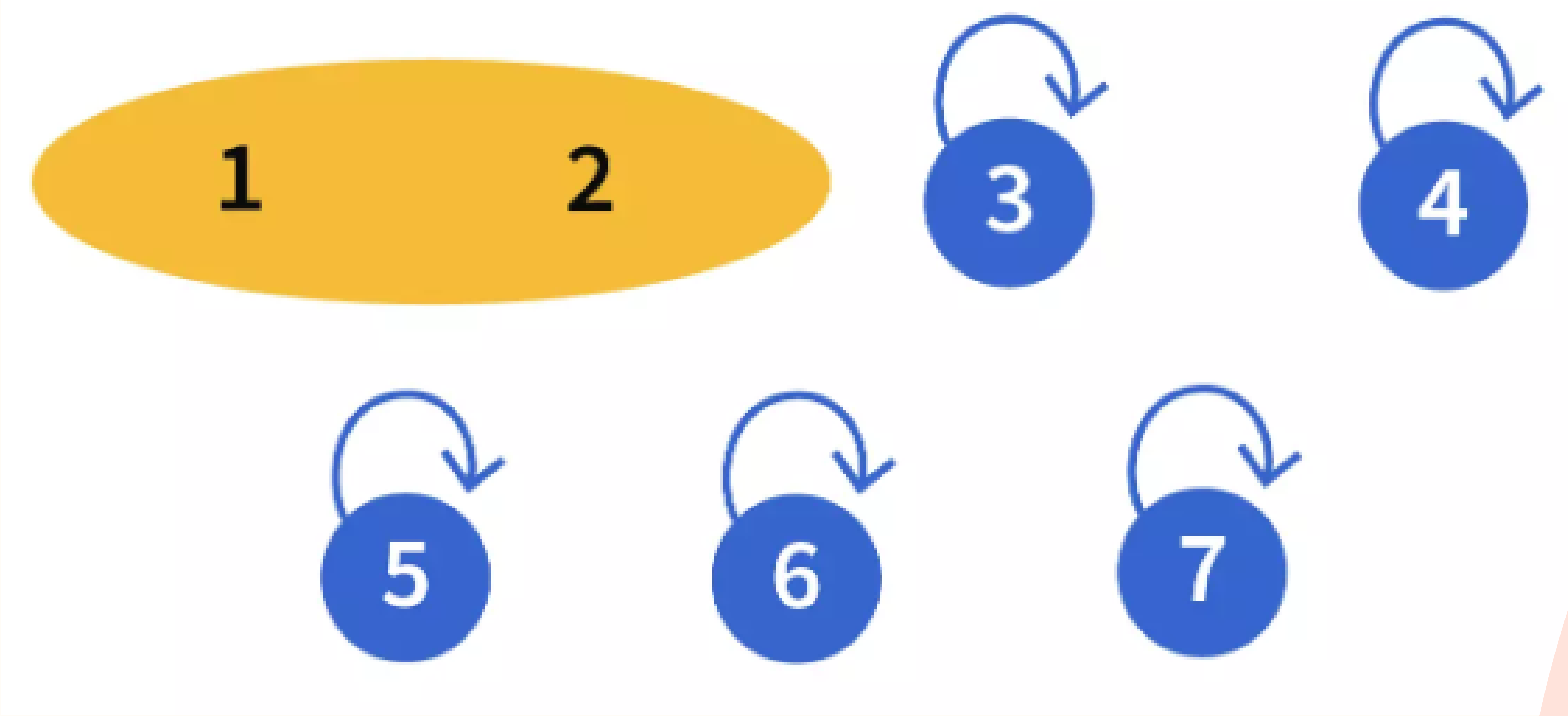
Essa operação une dois conjuntos, gerando assim um novo conjunto caracterizado por ser a junção de outros dois conjuntos.

Exemplo 1



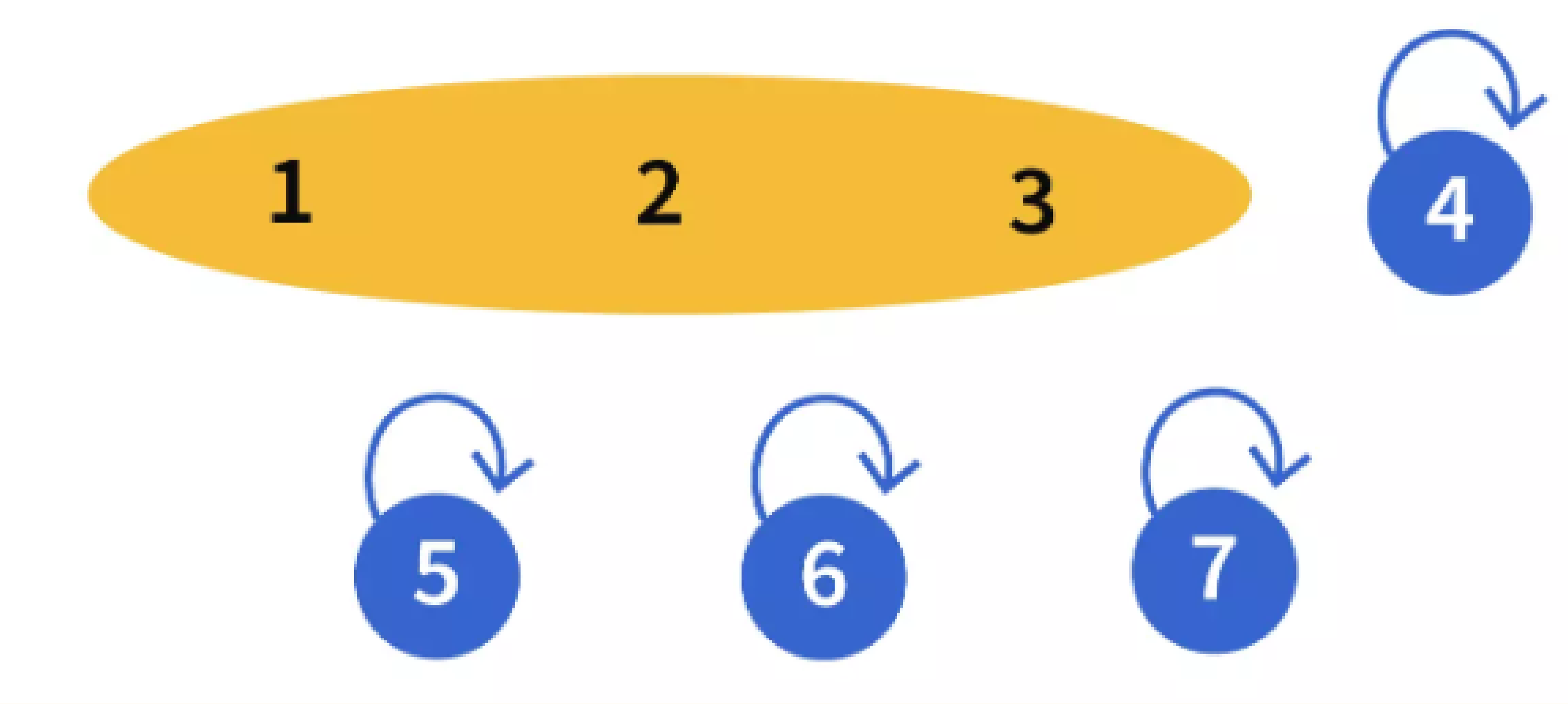
Exemplo 1

Union(1, 2)



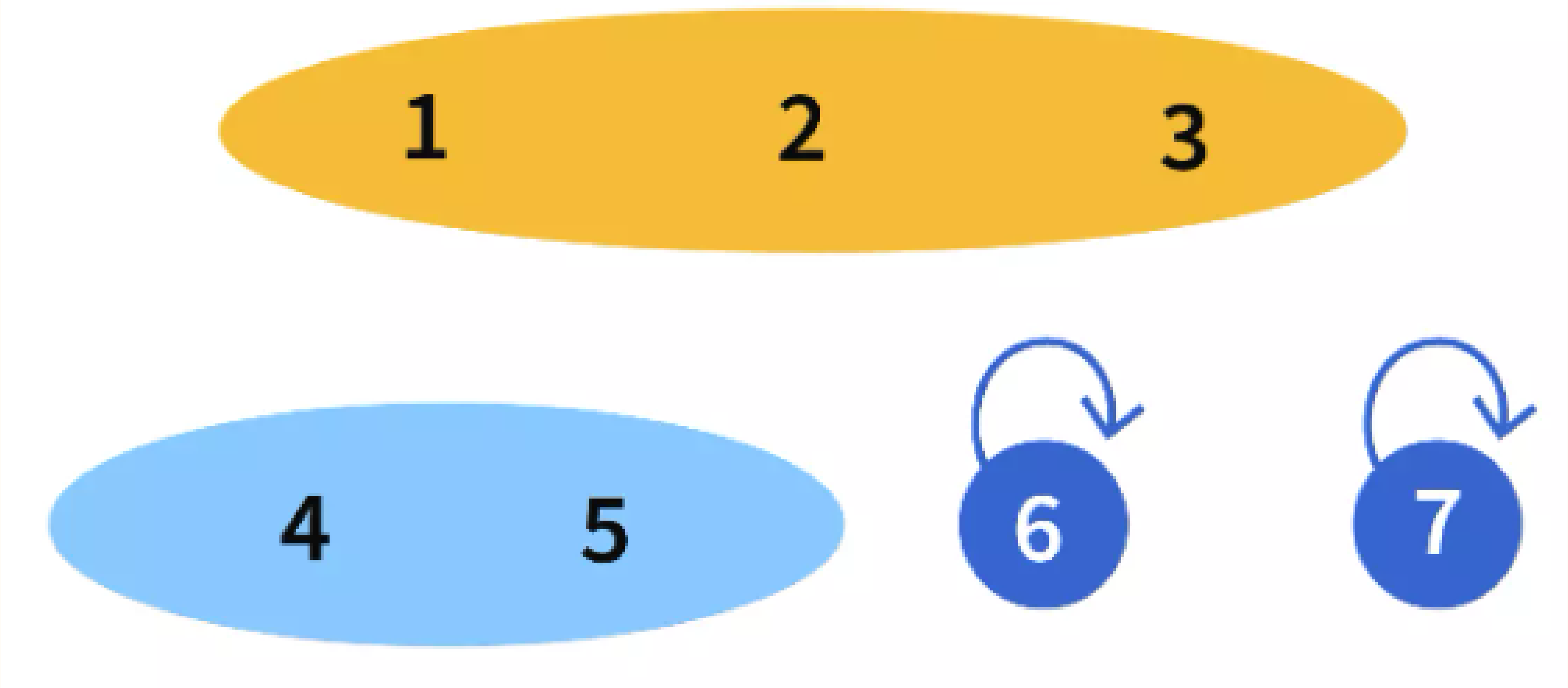
Exemplo 1

Union(2, 3)



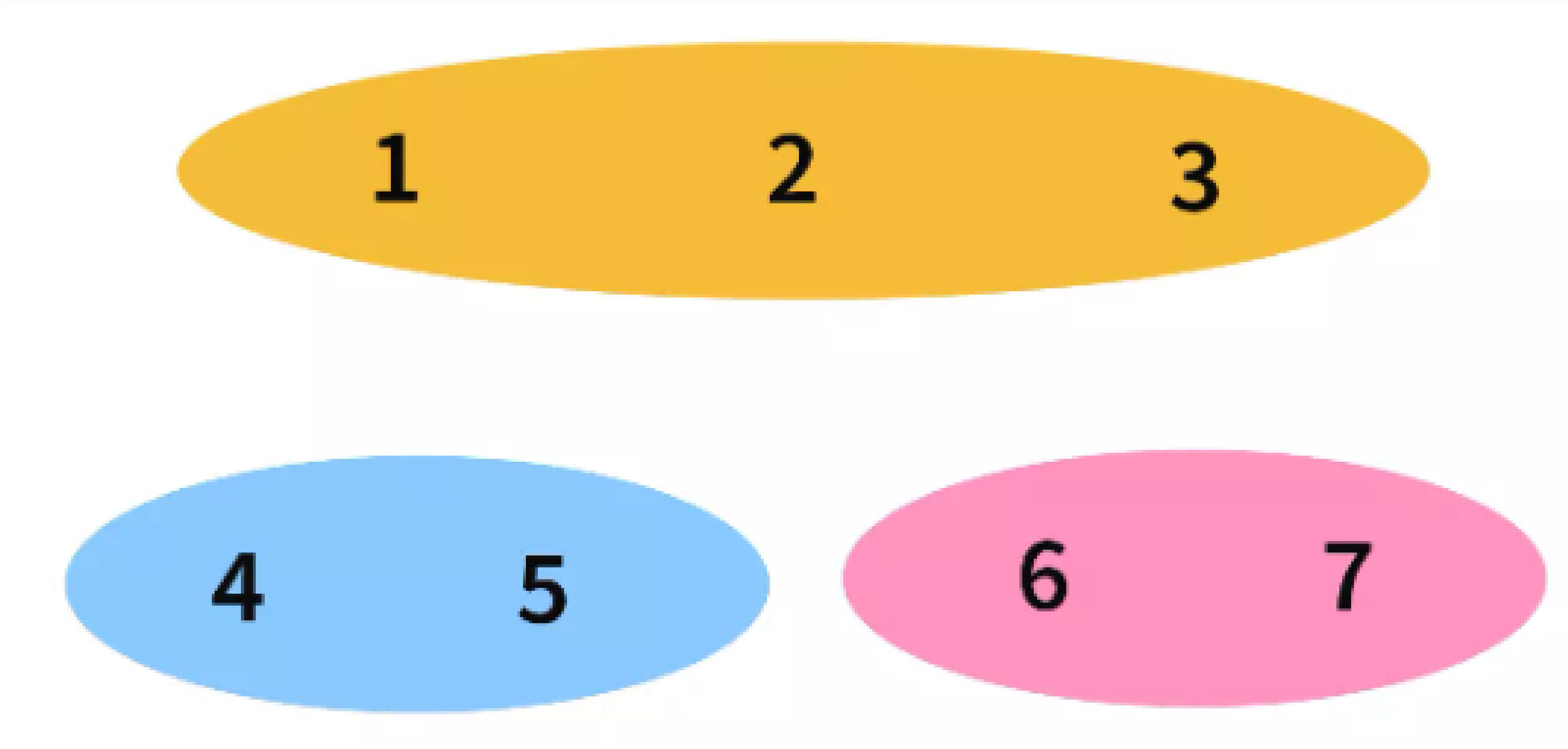
Exemplo 1

Union(4, 5)



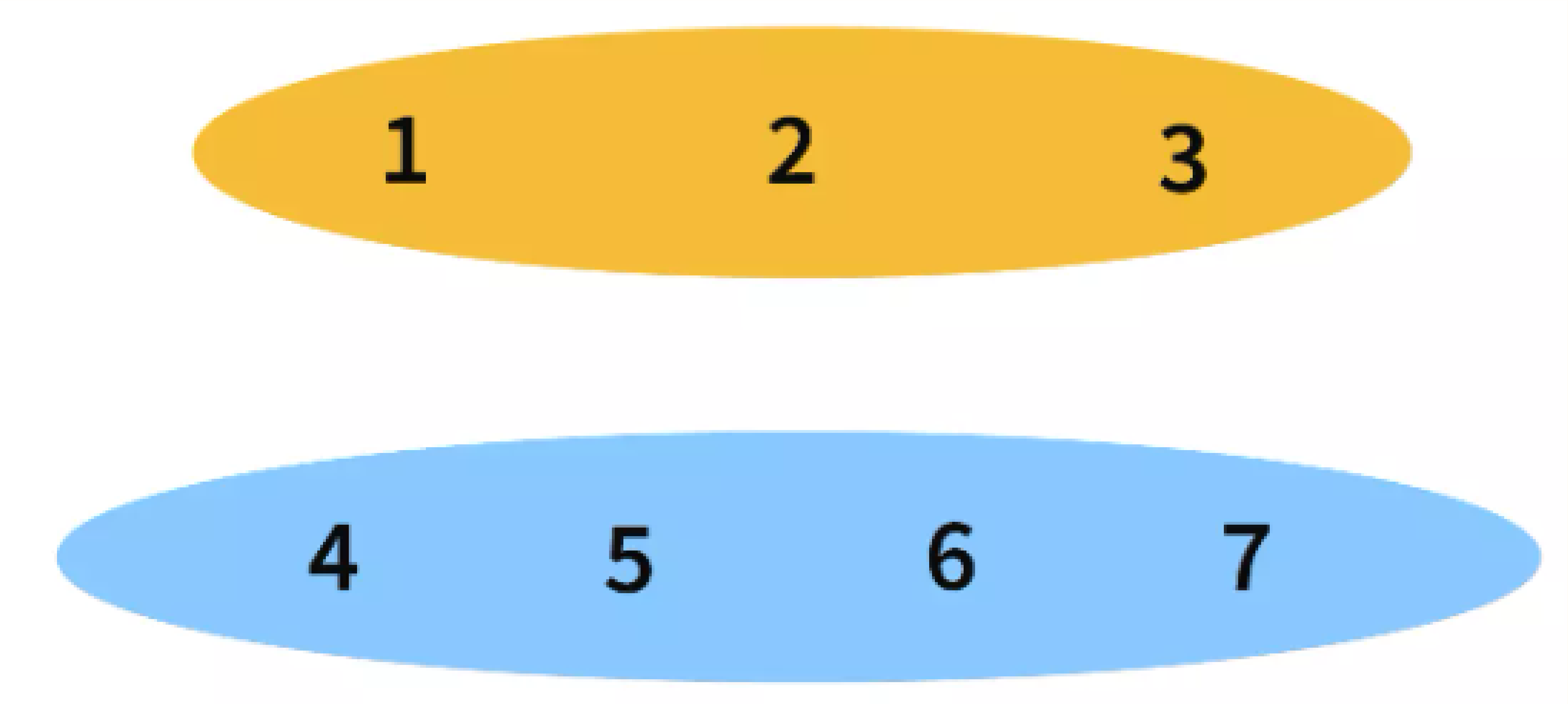
Exemplo 1

Union(6, 7)



Exemplo 1

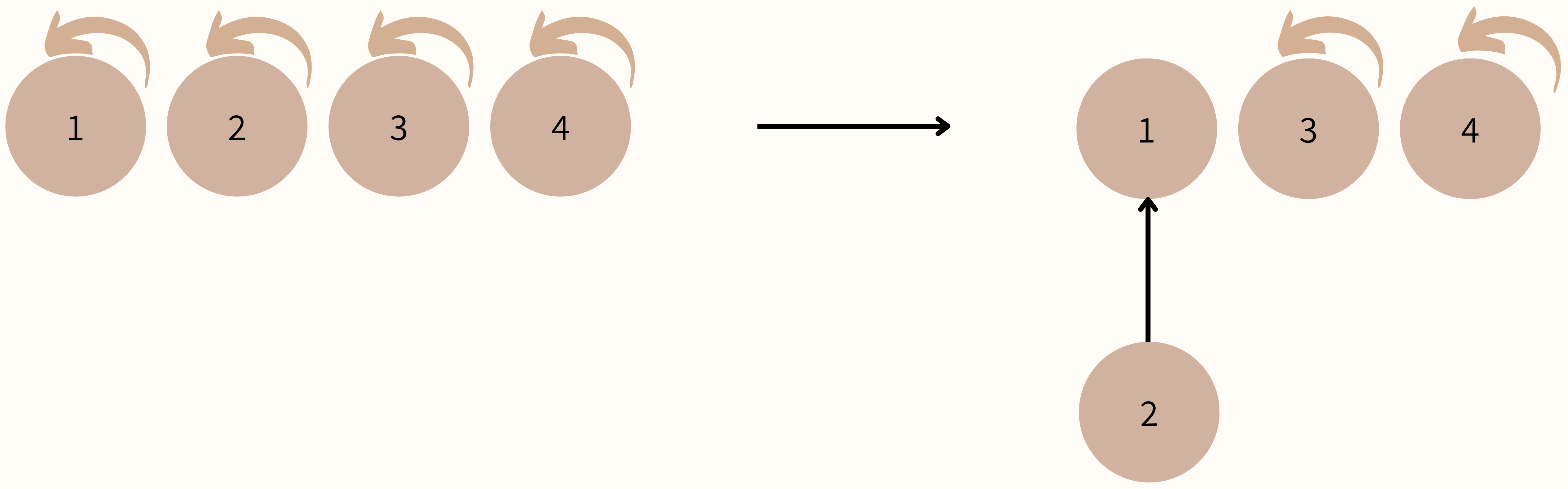
Union(5, 6)



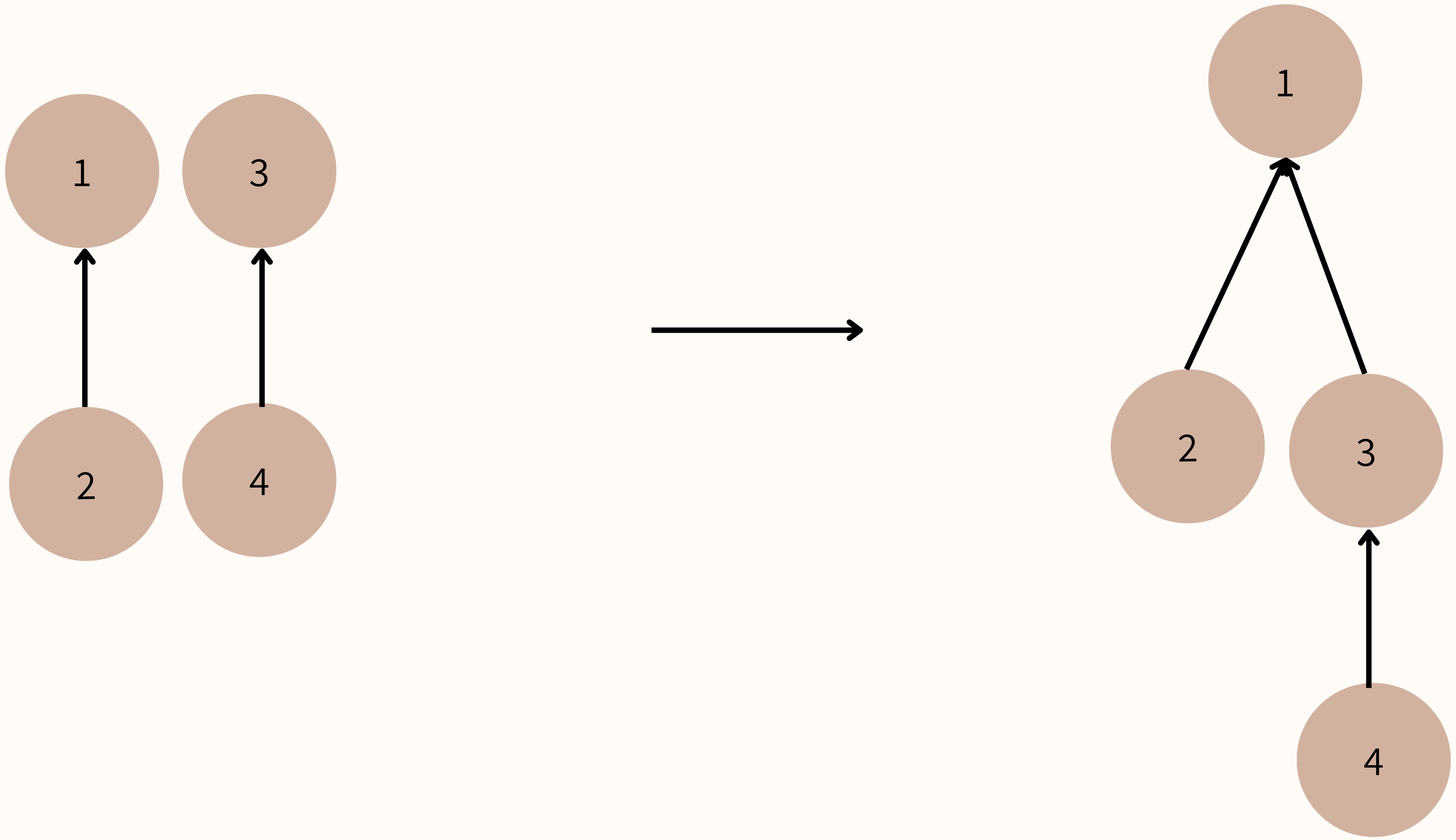
Exemplo 1

Union(2, 6)

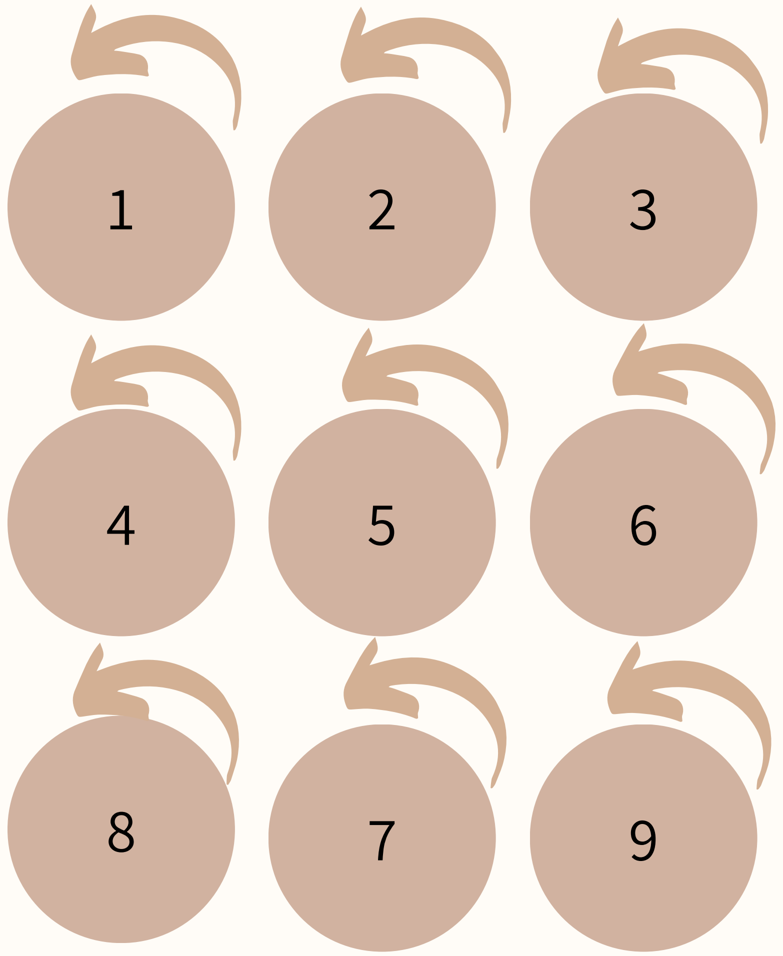
Exemplo 2



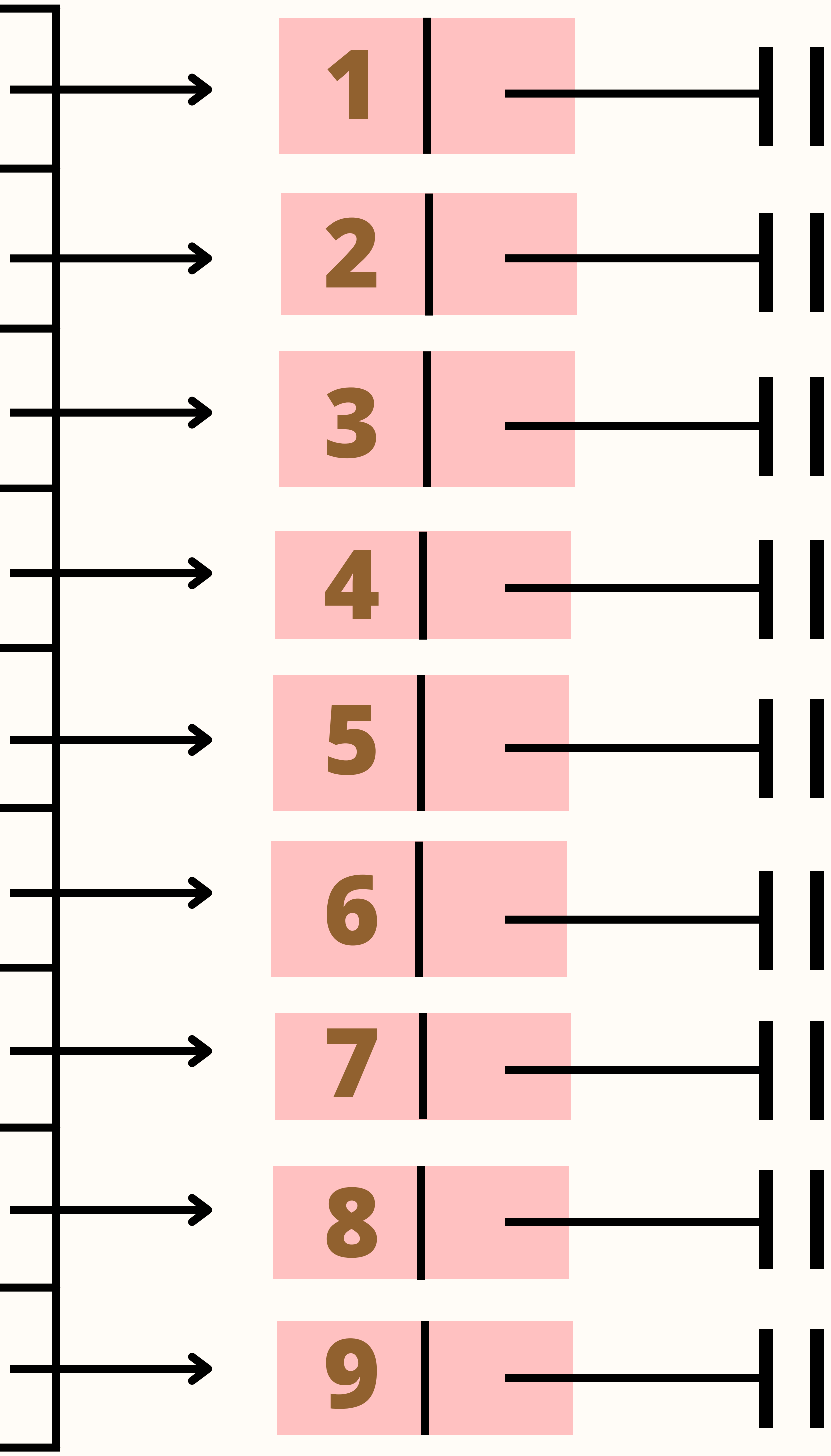
Exemplo 2



Exemplo 3



	R	T	L
1	1	1	
2	2	1	
3	3	1	
4	4	1	
5	5	1	
6	6	1	
7	7	1	
8	8	1	
9	9	1	



R = Representante
T = Tamanho
L = Lista do número

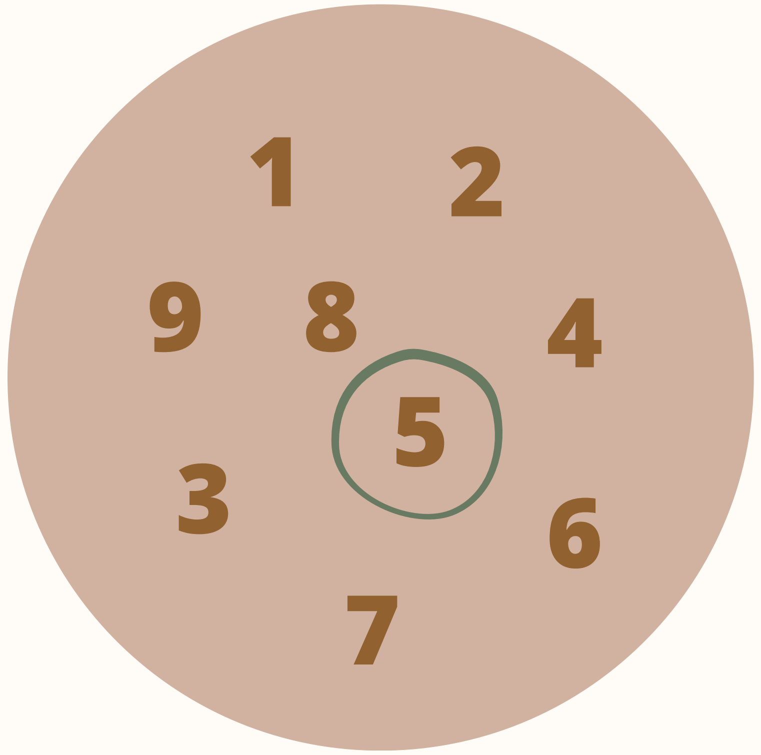
Exemplo 3

- **Operações realizadas:**

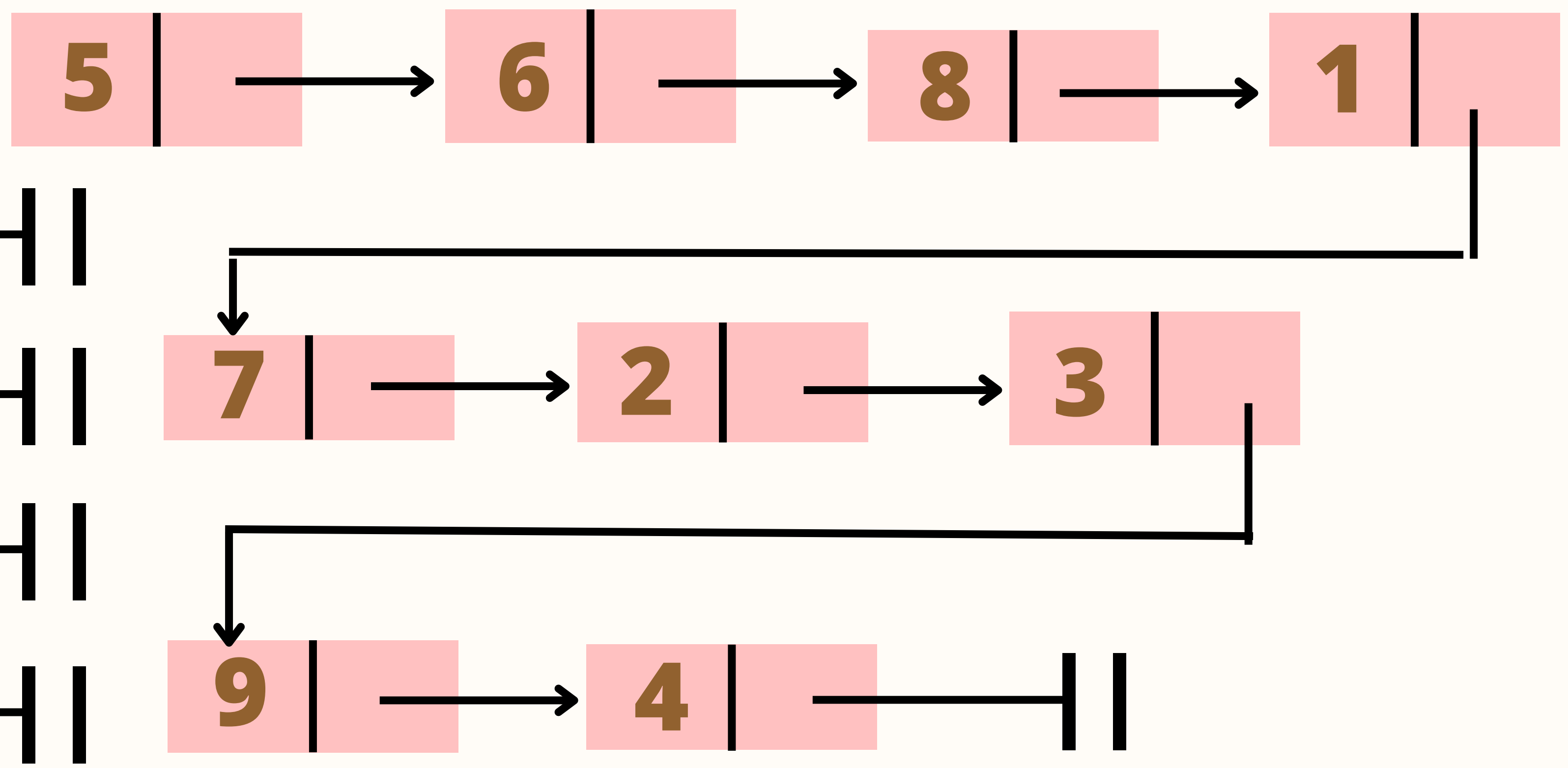
Union(2, 3)	**Union(2, 4)**
Union(5,6)	**Union(6, 8)**
Union(3, 9)	**Union(6, 7)**
Union(1, 7)	**Union(3, 4)**

Union(3, 8)

Exemplo 3



	R	T	L
1	5	0	
2	5	0	
3	5	0	
4	5	0	
5	5	9	
6	5	0	
7	5	0	
8	5	0	
9	5	0	



R = Representante
T = Tamanho
L = Lista do número

Implementação Encadeada

Roteiro de Código

01 ESTRUTURA

Struct Elemento:

- valor
- apontador para o próximo elemento
- apontador para o seu representante
- apontador para o seu conjunto

Struct Conjunto:

- apontador para o elemento da cabeça
- apontador para o elemento da cauda
- tamanho

02 MAKE SET

- Recebe um valor como parâmetro : ‘valor’
- Aloca memória para um conjunto e para um elemento: ‘conjunto’ e ‘elemento’
- elemento->valor = valor
- elemento->próximo = NULL
- elemento->representante = elemento
- elemento->conjunto = conjunto
- conjunto->cabeça = elemento
- conjunto->cauda = elemento
- conjunto->tamanho = 1
- Retorna o conjunto unitário

03 FIND SET

- Recebe um elemento como parâmetro e retorna seu representante

Roteiro de Código

04

UNION SET

- Recebe dois elementos como parâmetro: A e B
- X = findset(A)->conjunto
- Y = findset(B)->conjunto
- Se X->tamanho <= Y->tamanho
- Para todo ‘e’ em X, e->representante = B
- Y->tamanho = X->tamanho + Y->tamanho
- X->tamanho = 0
- Y->cauda->próximo = X->cabeça
- Y->cauda = X->cauda
- X->cabeça = NULL
- X->cauda = NULL
- E se o tamanho do primeiro for maior que o do segundo faz esse mesmo passo a passo mas invertendo o primeiro com o segundo

Aplicações

Possíveis Aplicações

01

SEGMENTAÇÃO DE IMAGENS

Permite o agrupamento de pixels com propriedades similares

02

APRENDIZADO DE MÁQUINA

Permite o agrupamento de dados com características similares

03

ALGORITMO DE KRUSKAL

Permite manter controle dos componentes conexos de um grafo e garantir que as propriedades da árvore geradora mínima em construção não sejam violadas.

Algoritmo de Kruskal

01 O QUE É

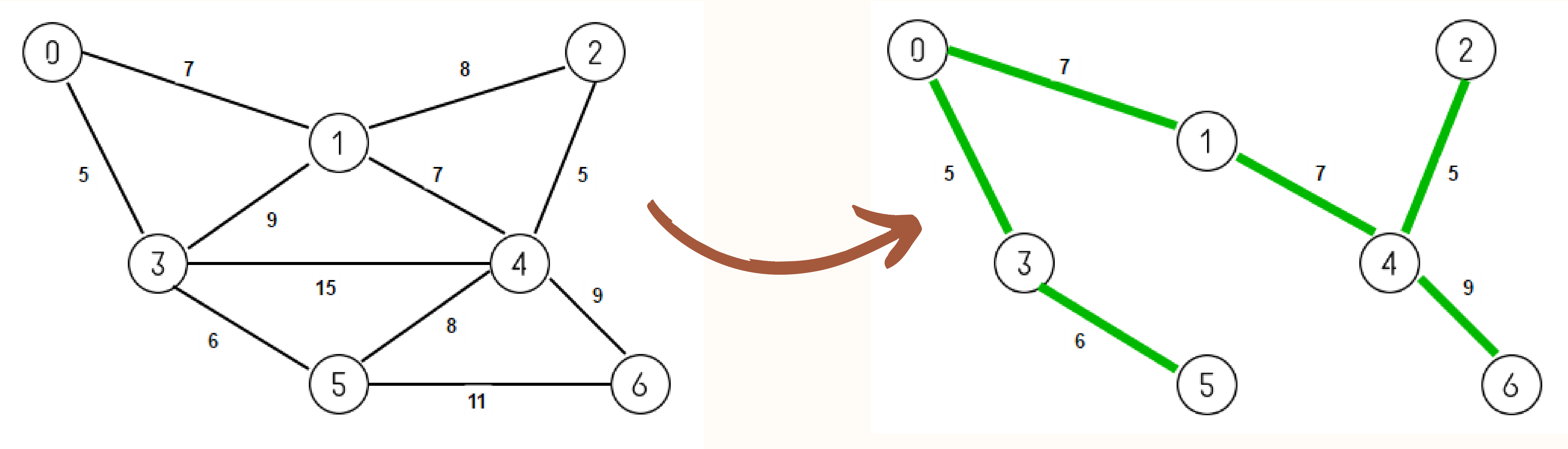
O Algoritmo de Kruskal é um algoritmo de otimização realizado em grafos conexos e não dirigidos

02 OBJETIVO

Seu objetivo é construir uma árvore geradora mínima de um grafo valorado que atenda aos critérios anteriores

03 ÁRVORE GERADORA MÍNIMA

É o subconjunto das arestas do grafo que cobre todos os vértices presentes nele de modo a soma dos pesos das arestas seja a menor possível.



Algoritmo de Kruskal

04

USO DA ESTRUTURA UNION FIND

É utilizada para verificar se a inserção de uma aresta irá ou não violar as propriedades da árvore geradora mínima

05

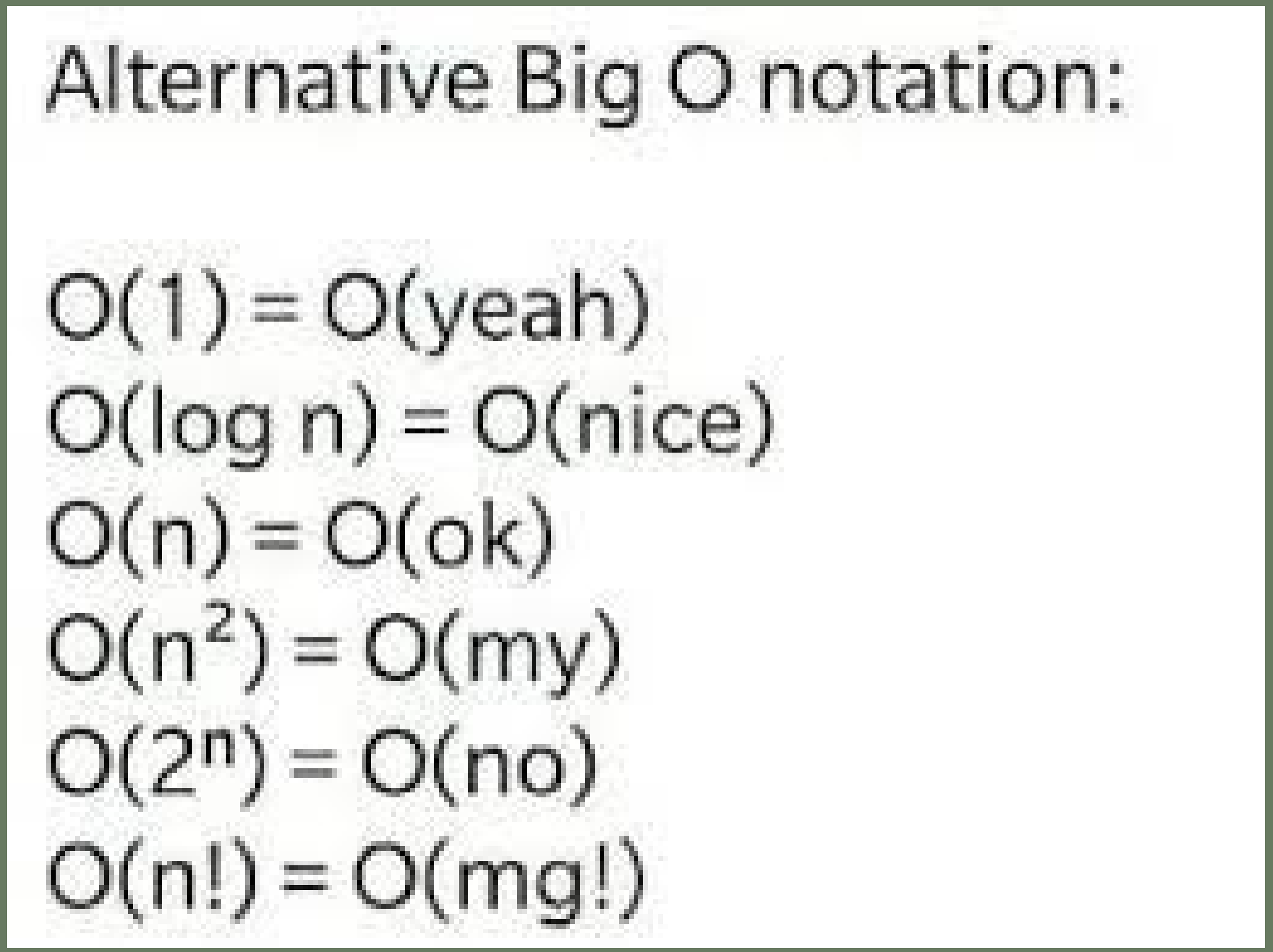
A VERIFICAÇÃO

A aresta apenas pode ser incluída quando os vértices conectados por elas não pertencem ao mesmo conjunto disjunto, já que se pertencessem iriam gerar um ciclo na árvore, o que não é permitido

06

INLCUINDO UMA ARESTA

Se a aresta passar pela verificação então ela é incluída e os conjuntos disjuntos aos quais seus vértices pertenciam são unidos

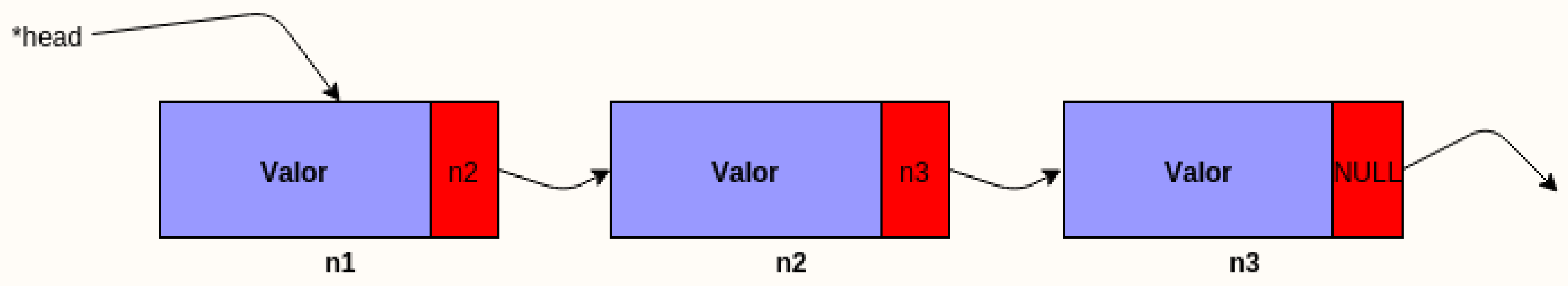


Complexidade do Algoritmo

O tempo de execução para as operações, na grande maioria das aplicações da estrutura, é na ordem de O(A(n)), onde A é a função inversa de Ackermann, a qual cresce de modo extremamente devagar.

Por isso, o tempo de execução é considerado constante.

Complexidade do Algoritmo Escolhido



Lista Encadeada

Na lista encadeada, a inicialização e busca sempre gastam o valor de O(1) para ser executada, ou seja, é constante.

Otimização

Podem ser feitos algumas melhorias nos Conjuntos Union-Find

01

COMPRESSÃO DE CAMINHO

Path Compression

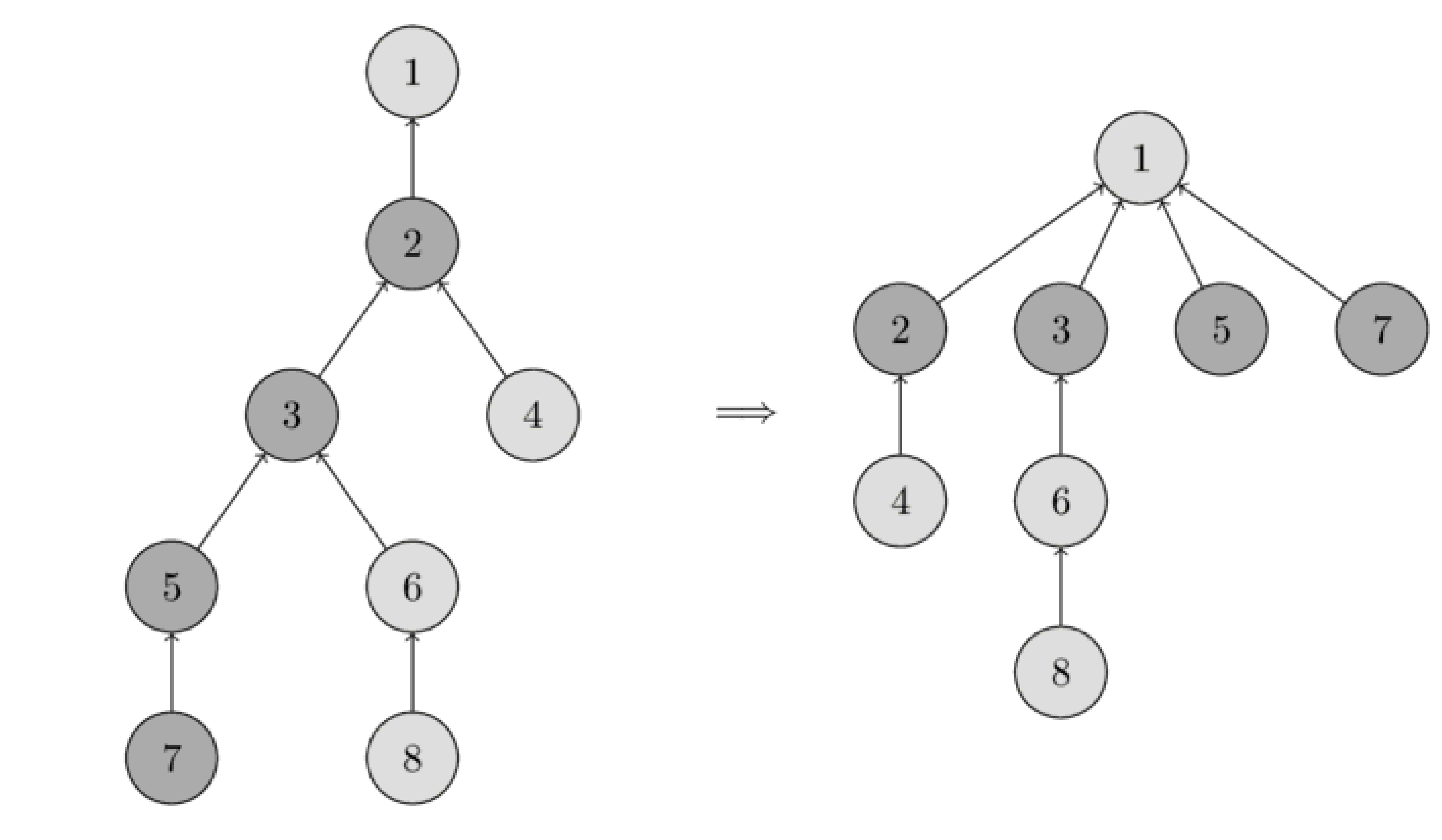
Cada nó aponta diretamente para a raiz

02

UNIÃO POR TAMANHO

Union by rank

Raiz da menor árvore aponta para a raiz da maior árvore



Compressão de Caminho

Otimização da operação Find
Atualiza o apontador de cada nó visitado para apontar diretamente para o representante do conjunto
Diminui a altura da árvore e, consequentemente, melhora a eficiência de operações Find futuras.

União por Tamanho

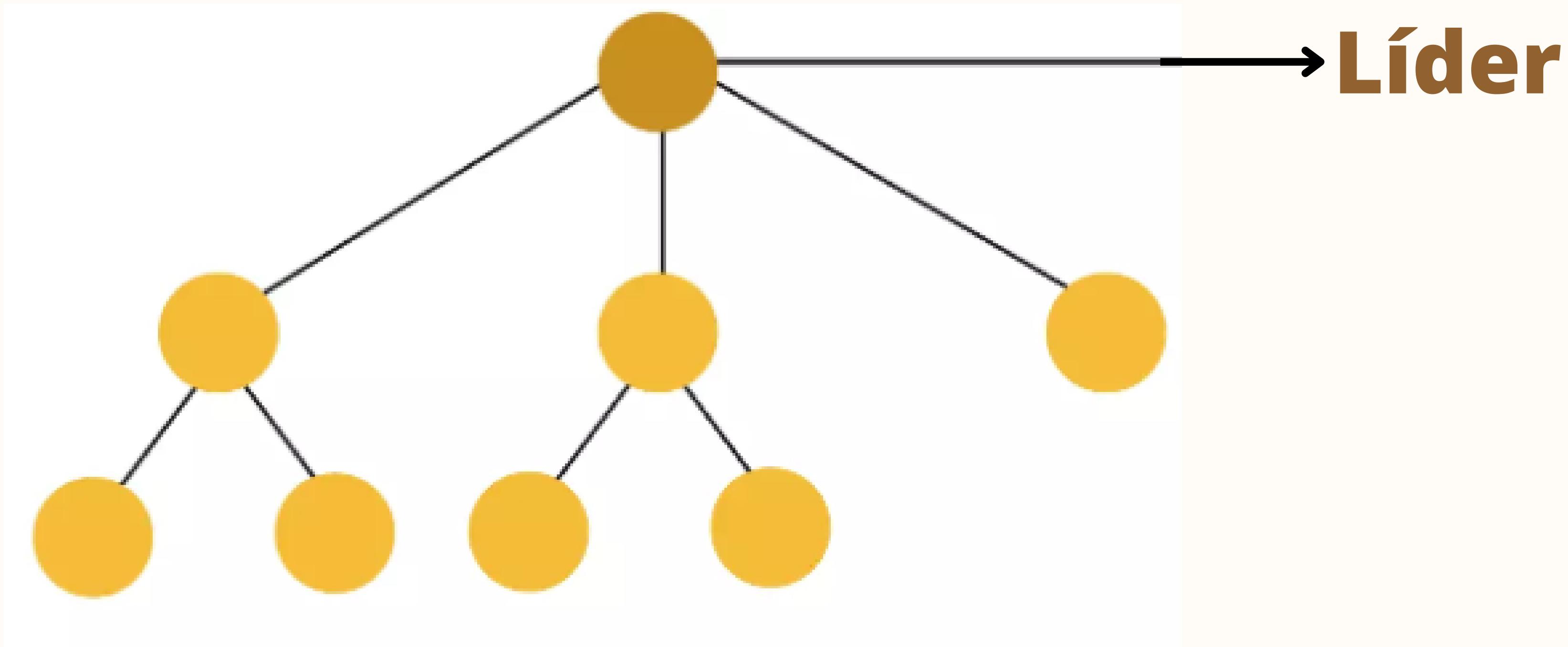
É necessário manter informações sobre o tamanho de cada conjunto

Assim, durante a operação de União, o conjunto de menor tamanho sempre se unirá ao de maior tamanho.

Isso ajuda a evitar um desbalanceamento de entre conjuntos e a operação de busca seja mais rápida.

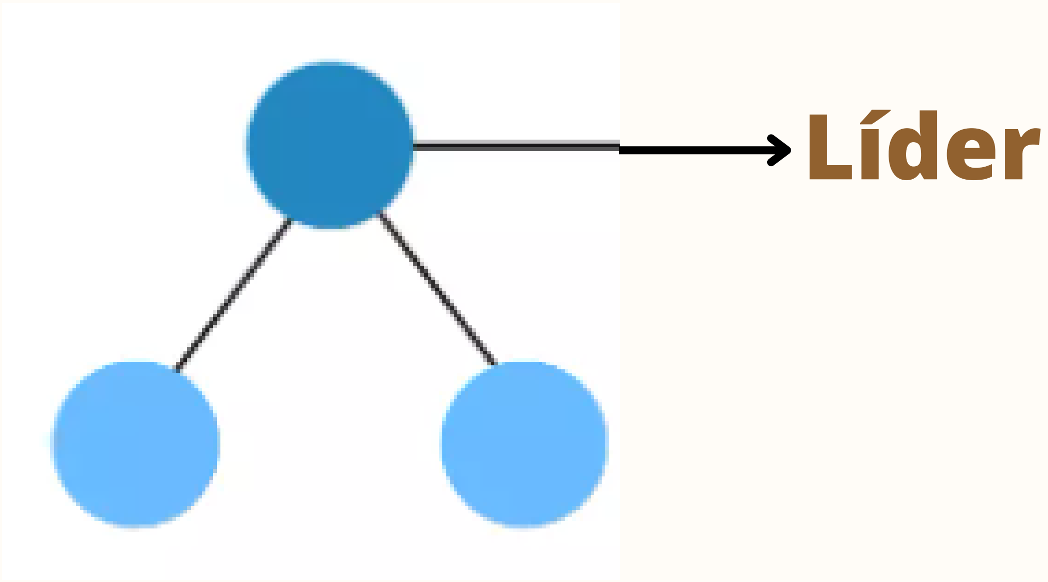
Exemplo

Árvore 1



Tamanho/Altura = 2

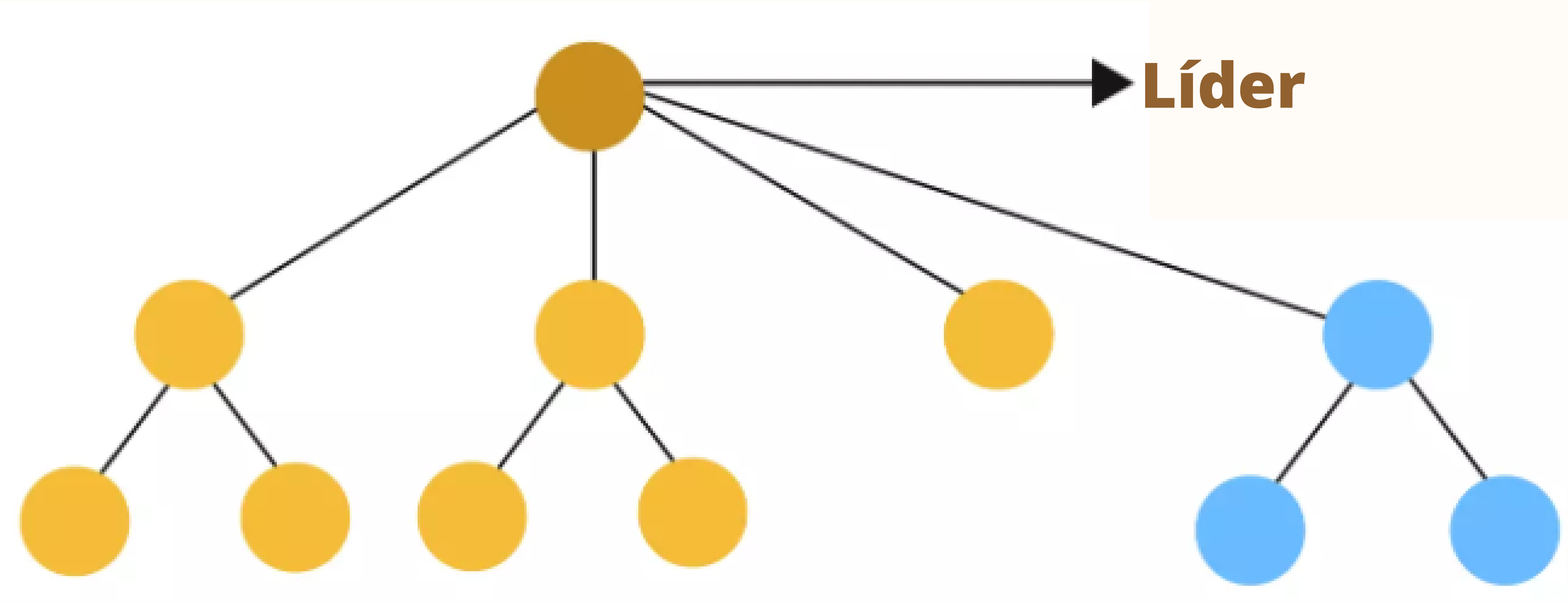
Árvore 2



Tamanho/Altura = 1

Exemplo

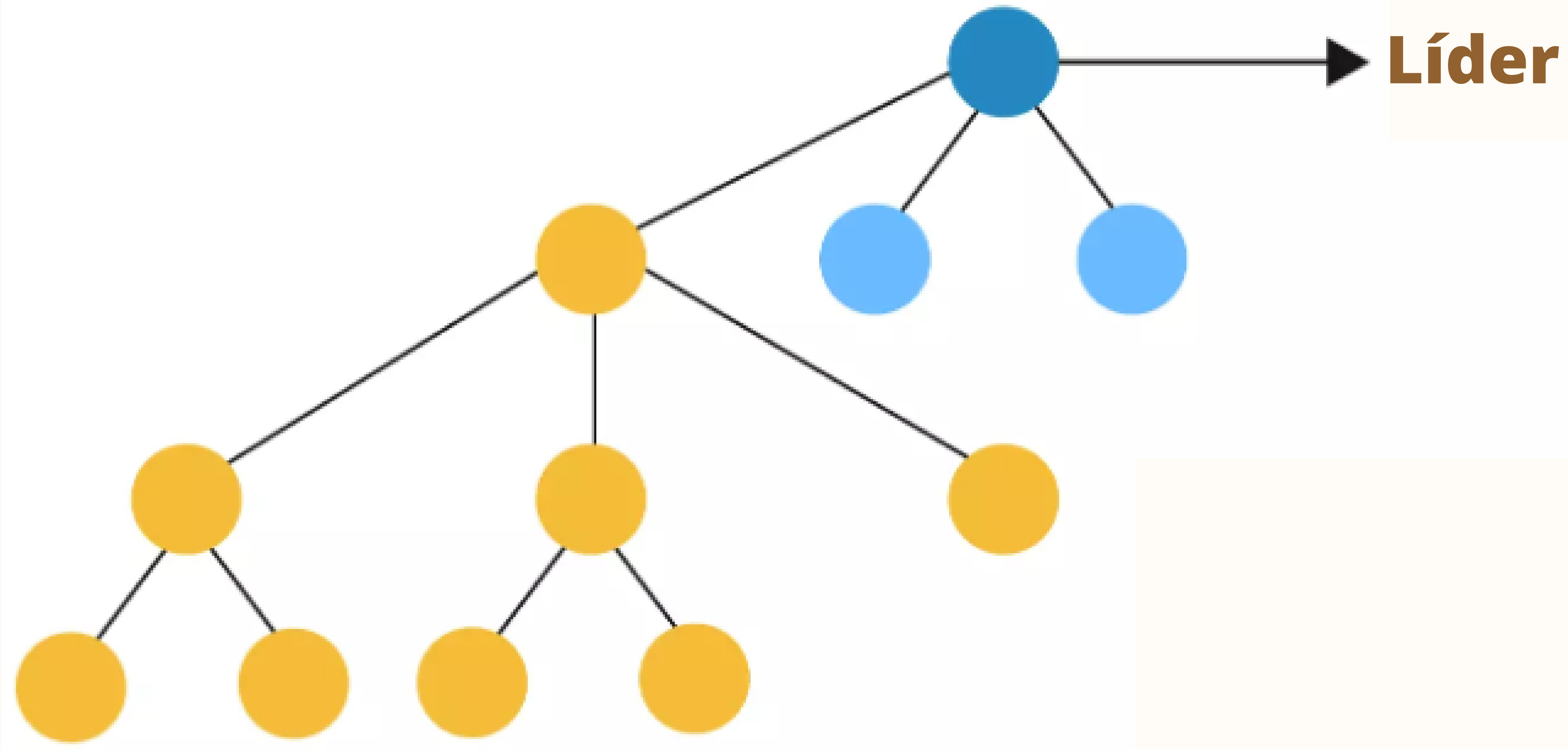
- **União por Tamanho**



Tamanho/Altura = 2

Exemplo

- **União sem verificar o tamanho**



Tamanho/Altura = 3

Complexidade do Algoritmo com Otimização

Árvore:

Utilizando árvore e com a otimização de compressão de caminho, nós conseguimos passar de O(n) para O(log n) na busca pelo representante do elemento.

Podemos também tentar diminuir o tempo de execução da união usando a união por tamanho em que, na maioria dos cenários, será O(A(n)) e no pior O(log n). Originalmente, qualquer união levaria O(log n).

Implementação por Árvore com otimização

Referências Bibliográficas:

https://www.scaler.com/topics/data-structures/disjoint-set/

https://en.wikipedia.org/wiki/Disjoint-set_data_structure

https://www.techiedelight.com/pt/disjoint-set-data-structure-union-find-algorithm/

https://cp-algorithms.com/data_structures/disjoint_set_union.html

http://desenvolvendosoftware.com.br/estruturas-de-dados/conjuntos-disjuntos.html#onde-conjuntos-disjuntos-são-usados

https://www.techiedelight.com/pt/kruskals-algorithm-for-finding-minimum-spanning-tree/

https://www.techiedelight.com/pt/union-find-algorithm-cycle-detection-graph/