



Union-Find

Equipe

Christian Gabriel da Silva Sabino

Felipe Gabriel Marques dos Santos

José Cristovão Vieira dos Santos Junior

Vítor Gabriel dos Santos Oliveira

Motivação

- Podemos supor o seguinte problema:
- Exemplo:

Hão n ilhas onde algumas estão conectadas por pontes. É necessário identificar as ilhas nas quais estão conectadas e as que não estão. Com a identificação das que não estão, iremos conectá-las.

Union-Find - O que é?

- O algoritmo Union-Find é uma estrutura de dados que gerencia e rastreia um conjunto de elementos divididos em conjuntos, podendo ser unitários, disjuntos, permitindo combinar dois conjuntos em um só e determinar qual conjunto um elemento pertence e, também, verificar se dois elementos pertencem ao mesmo conjunto.

Union-Find - Para que serve?

- Ele é usado para resolver problemas de conectividade, onde é necessário agrupar elementos e checar rapidamente se pertencem ao mesmo grupo. Como no exemplo das ilhas conectadas por pontes.

Union-Find - Definições

- **Set:** Um grupo de elementos que não se sobrepõem a outros conjuntos. Cada elemento pertence a um único conjunto.
- **Operação Find:** Uma operação que retorna o representante (ou "raiz") do conjunto ao qual um elemento pertence. Essa raiz serve como identificador único do conjunto.
- **Operação Union:** Uma operação que une dois conjuntos em um só, atualizando o representante dos elementos para que passem a compartilhar o mesmo conjunto.

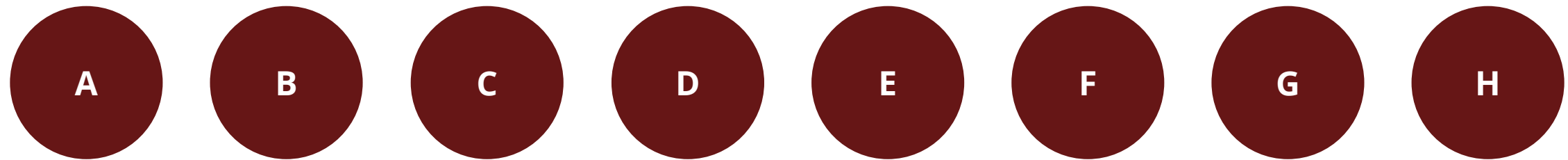
Union-Find - Definições

- **Representante:** Um elemento especial que atua como o identificador principal do conjunto. Todos os elementos de um conjunto compartilham o mesmo representante.
- **Union by Rank:** Técnica de otimização que une o conjunto menor ao maior para manter a estrutura de dados balanceada e reduzir a profundidade da árvore.
- **Path Compression (Compactação de Caminho):** Técnica de otimização que encurta o caminho dos elementos ao representante, fazendo com que cada elemento aponte diretamente para a raiz, tornando as próximas buscas mais rápidas.

Union

```
void unir(UnionFind *uf, int elemento1, int elemento2) {  
  
    int raiz1 = find(uf, elemento1);  
    int raiz2 = find(uf, elemento2);  
  
    if (raiz1 == raiz2) return;  
  
    if (uf->altura[raiz1] > uf->altura[raiz2]) {  
        uf->raiz[raiz2] = raiz1;  
    } else if (uf->altura[raiz1] < uf->altura[raiz2]) {  
        uf->raiz[raiz1] = raiz2;  
    } else {  
        uf->raiz[raiz2] = raiz1;  
        uf->altura[raiz1] += 1;  
    }  
}
```

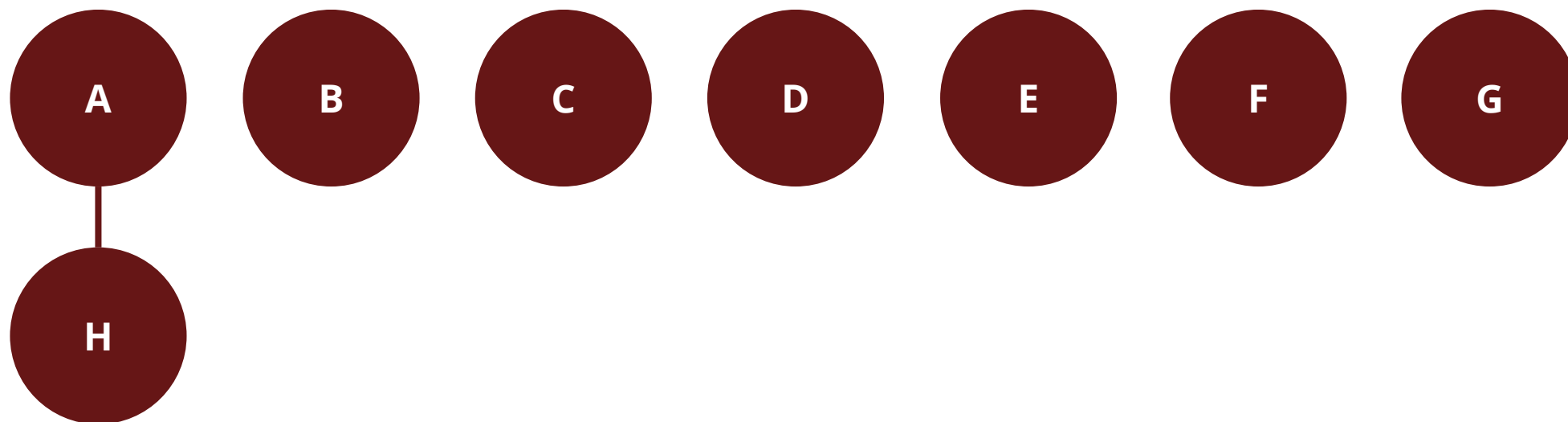
Animação - União (sem Path Compression/Union by Rank)



`union(A, H)`

Animação - União (sem Path Compression/Union by Rank)

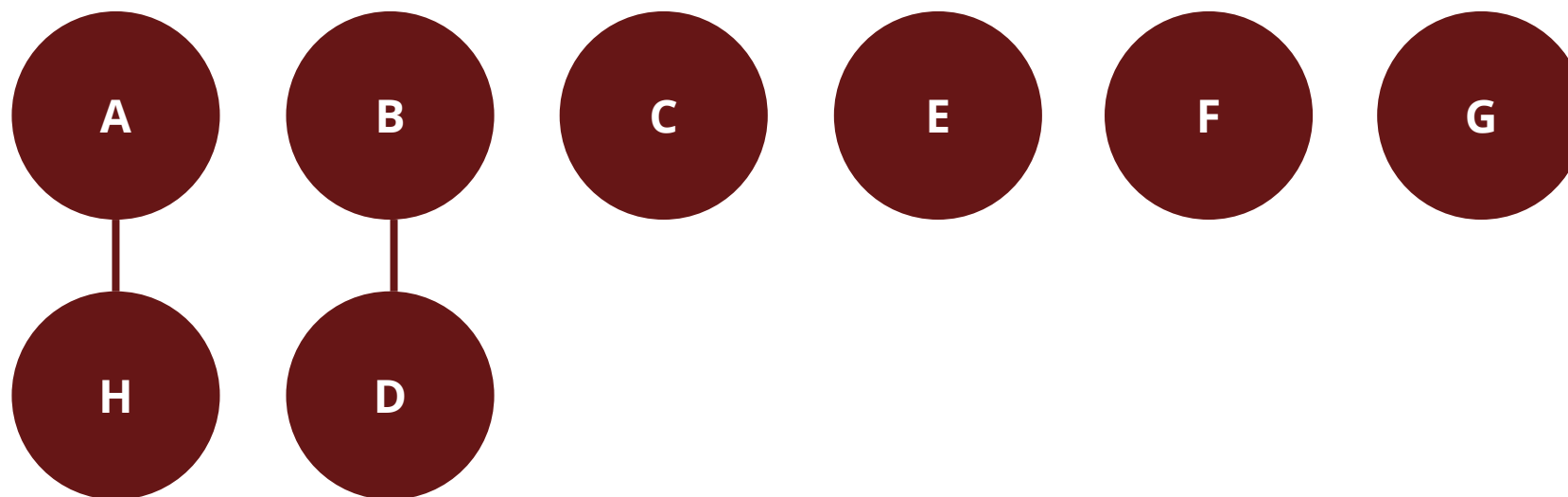
union(A, H)



union(B, D)

Animação - União (sem Path Compression/Union by Rank)

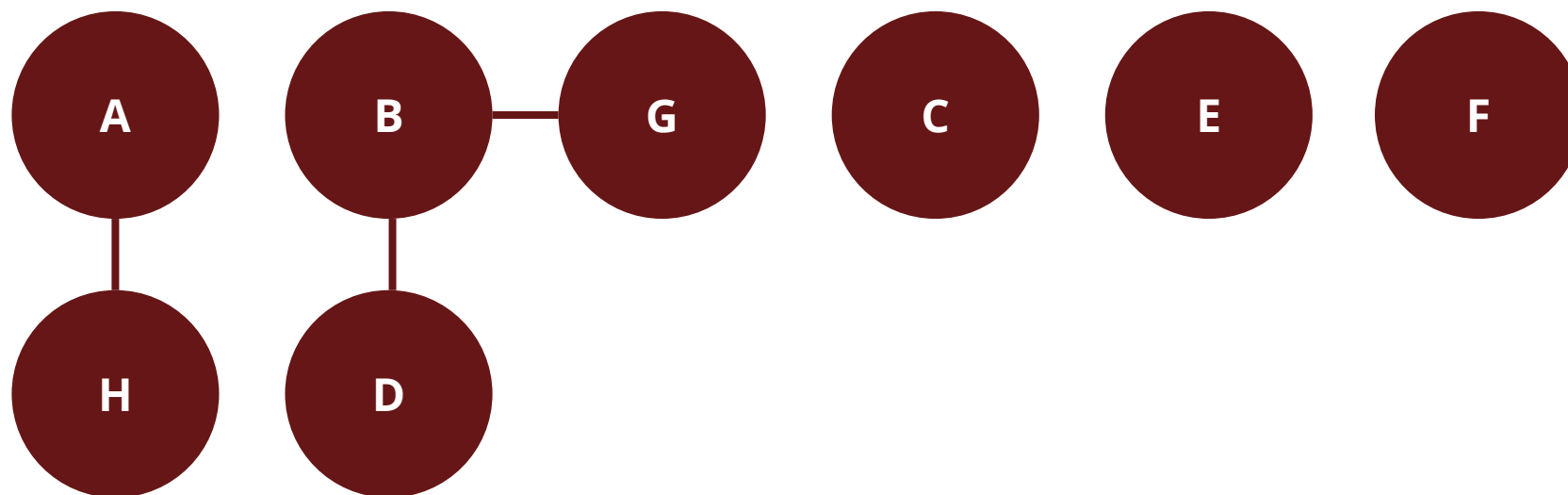
union(B, D)



union(G, D)

Animação - União (sem Path Compression/Union by Rank)

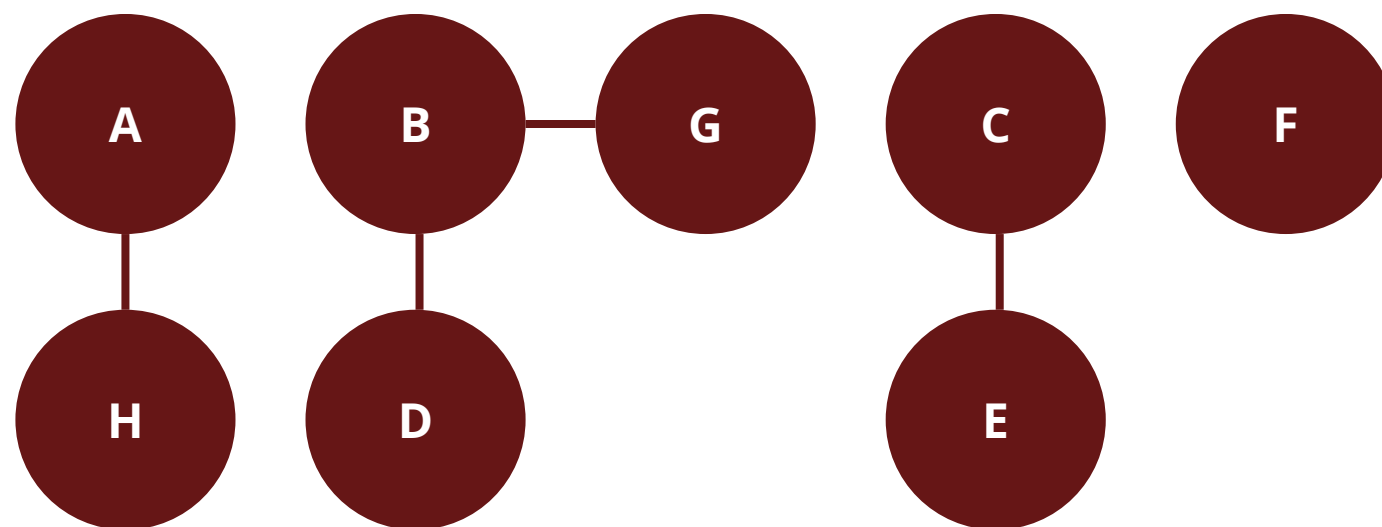
union(G, D)



union(E, C)

Animação - União (sem Path Compression/Union by Rank)

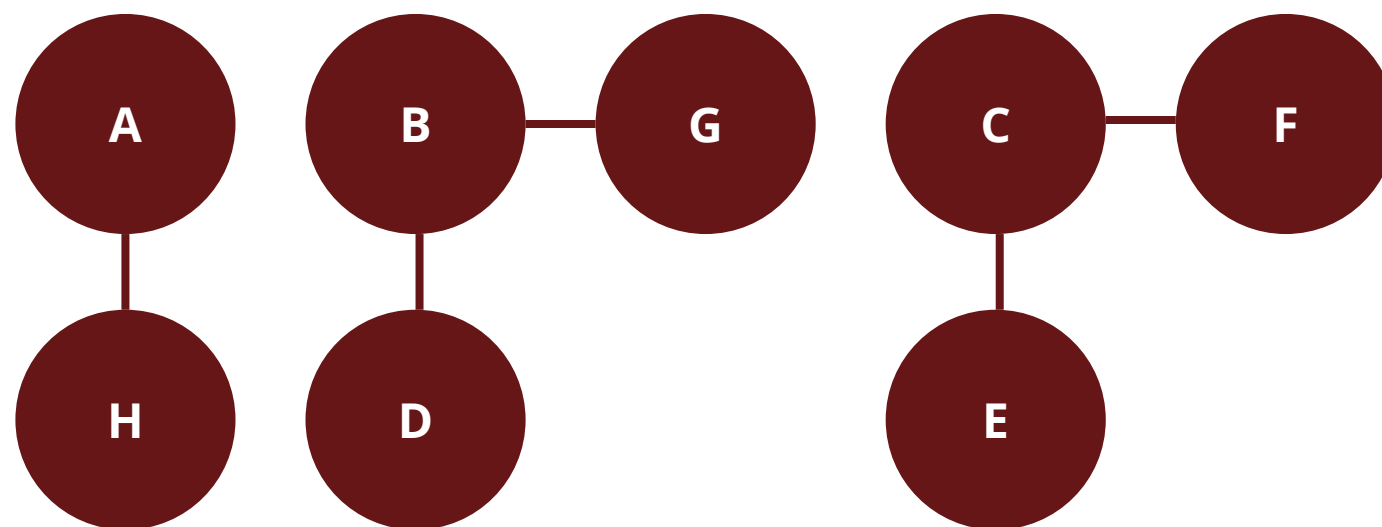
union(E, C)



union(E, F)

Animação - União (sem Path Compression/Union by Rank)

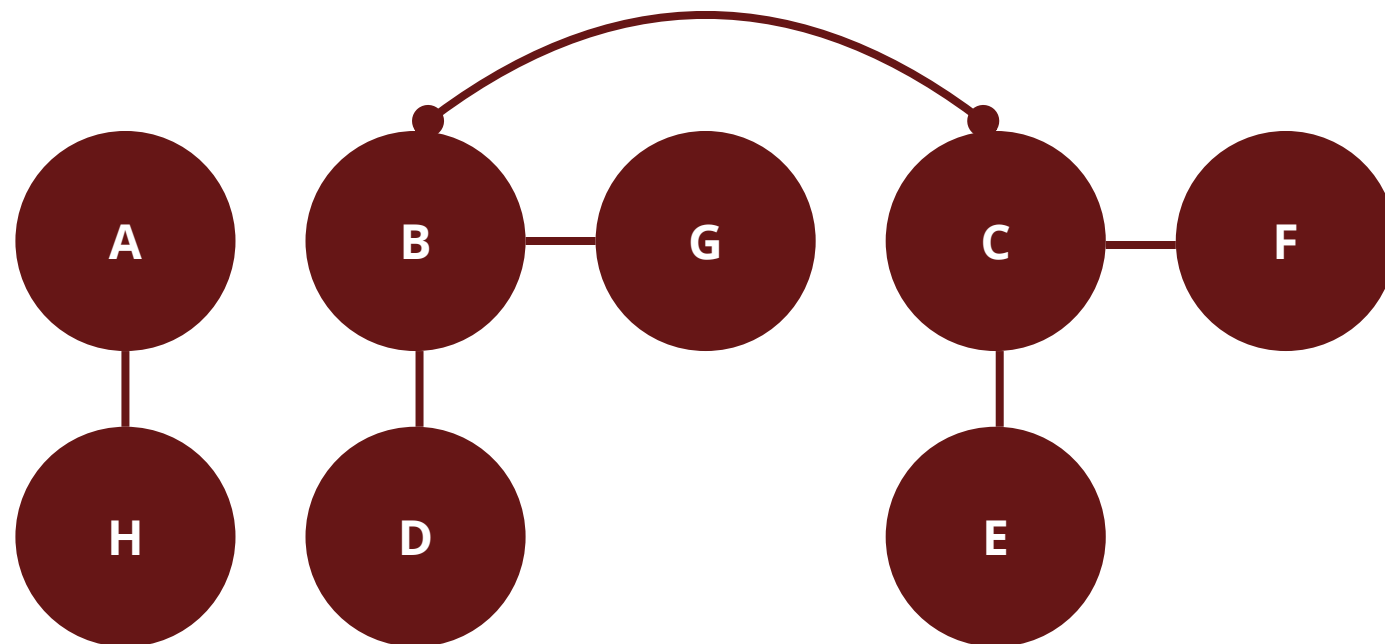
union(E, F)



union(D, F)

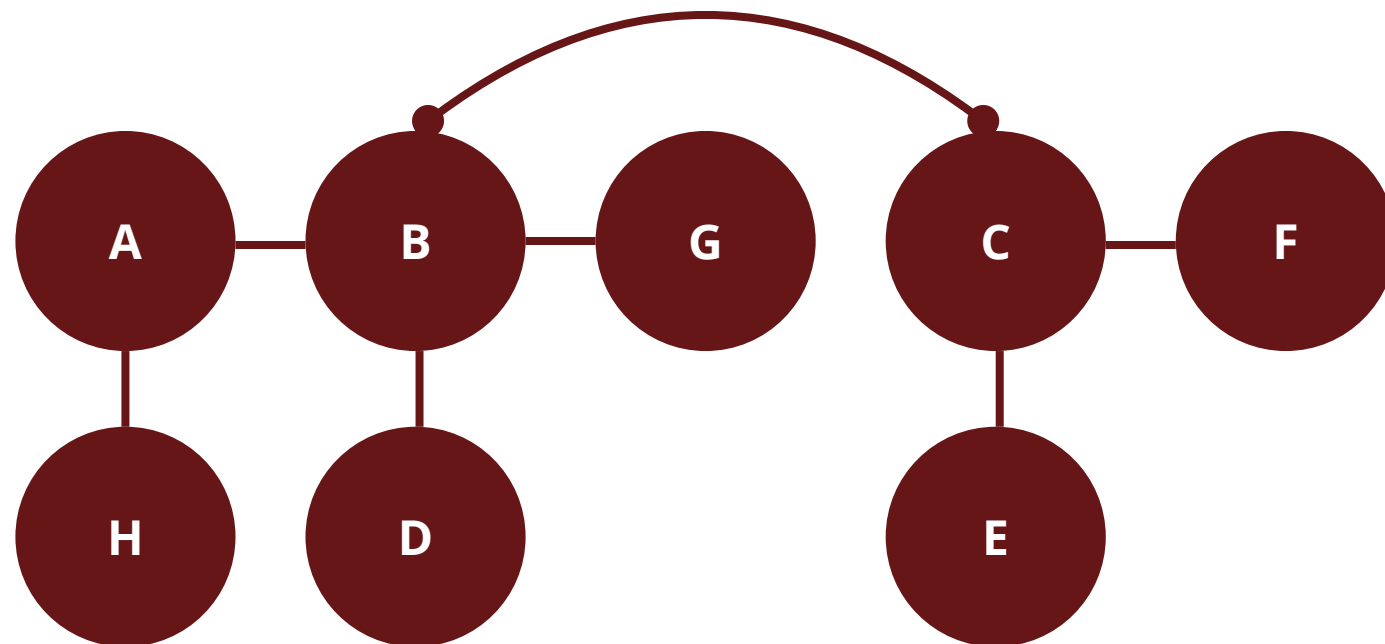
Animação - União (sem Path Compression/Union by Rank)

union(D, F)



Animação - União (sem Path Compression/Union by Rank)

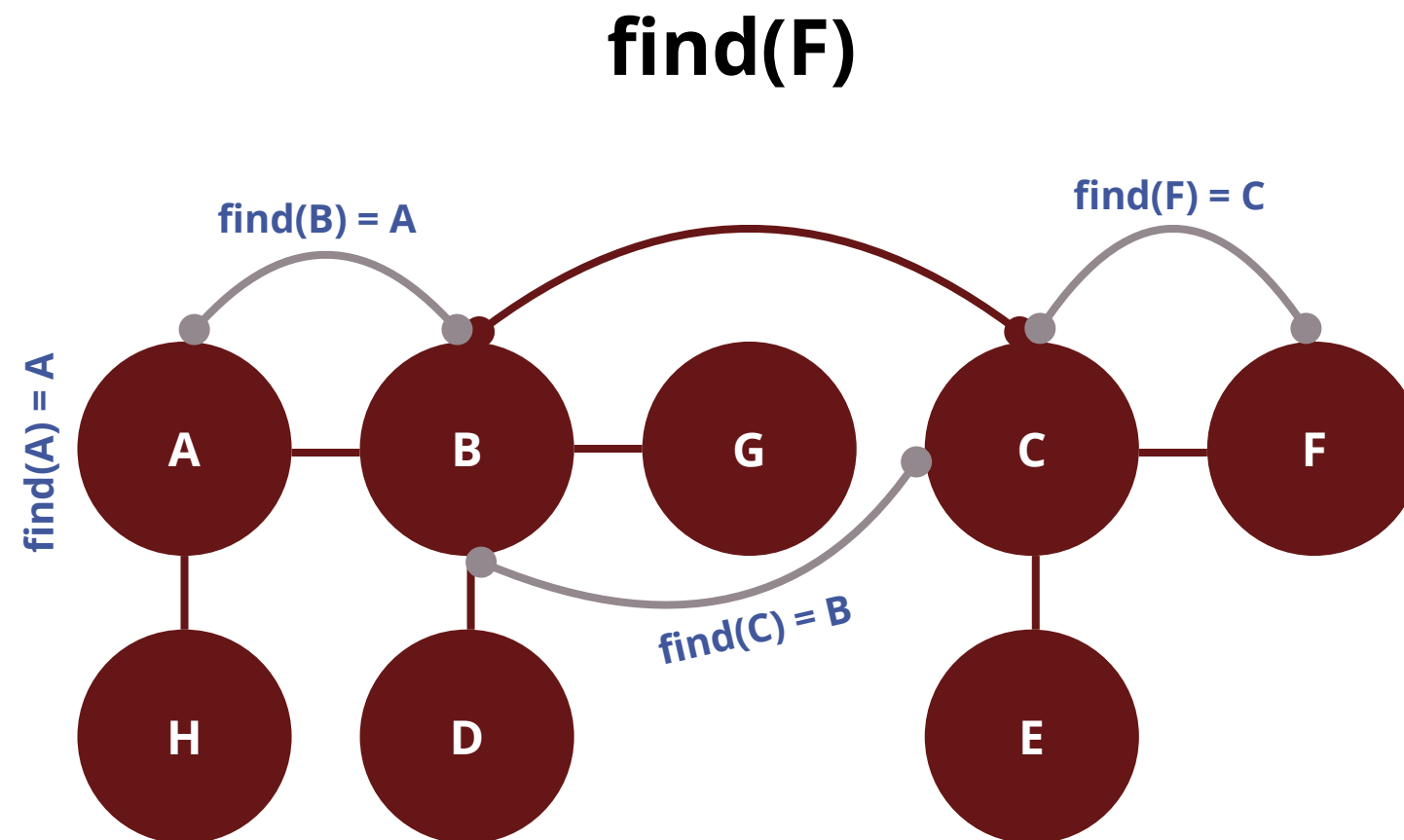
union(A, B)



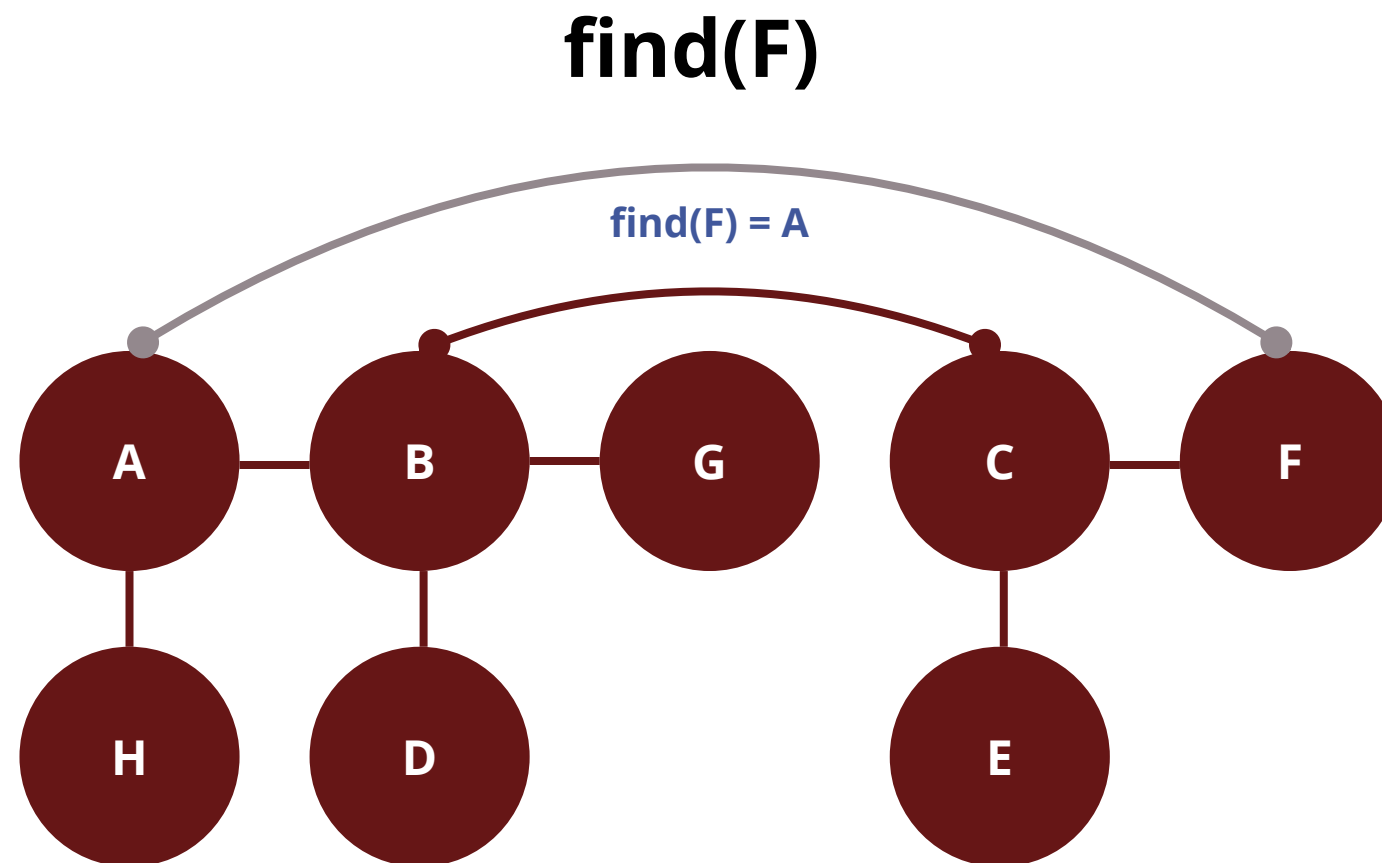
Find

```
int find(UnionFind *uf, int elemento) {  
    if (uf->raiz[elemento] != elemento) {  
        uf->raiz[elemento] = find(uf, uf->raiz[elemento]);  
    }  
    return uf->raiz[elemento];  
}
```


Animação - Find (sem Path Compression)

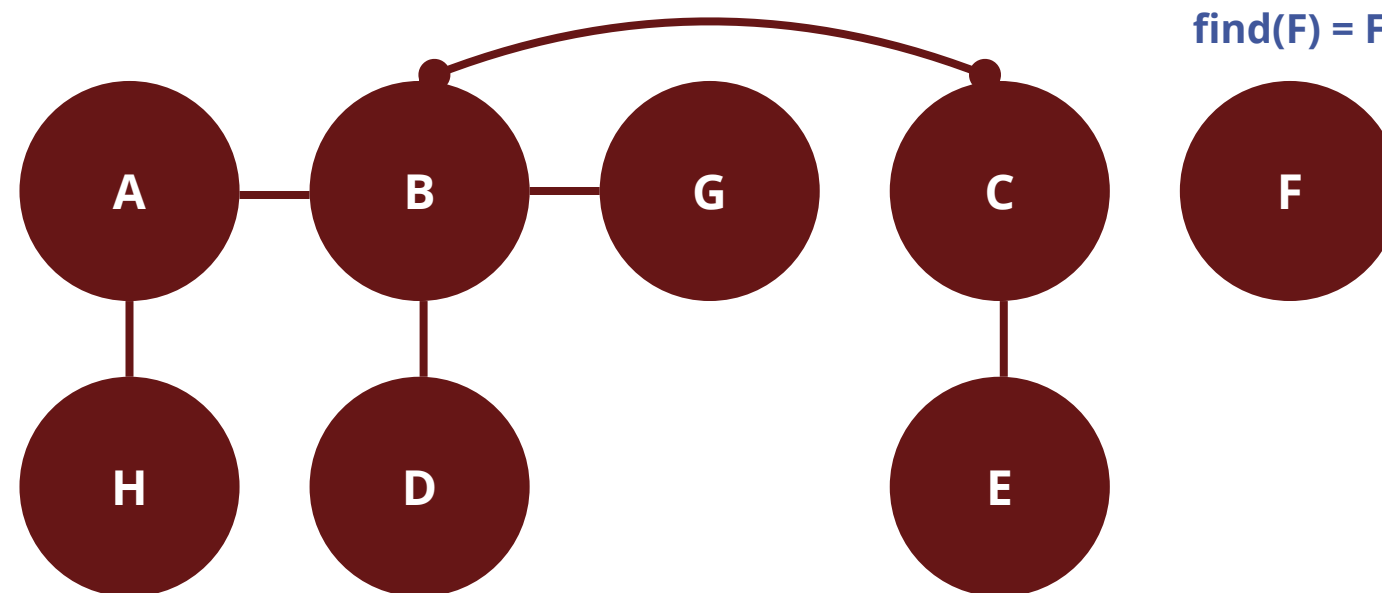


Animação - Find (sem Path Compression)

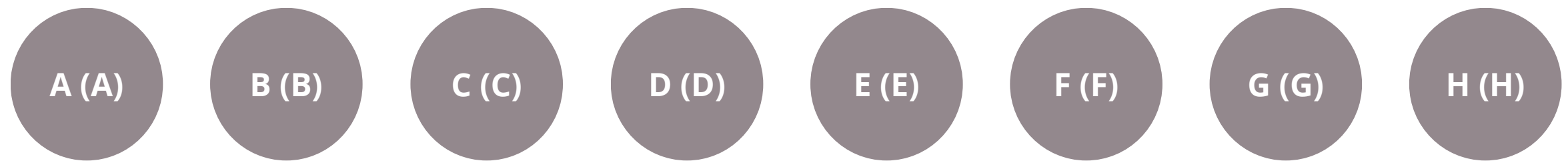


Animação - Find (sem Path Compression)

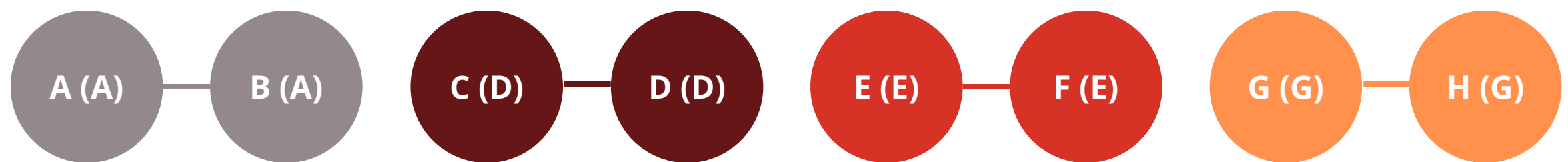
find(F)



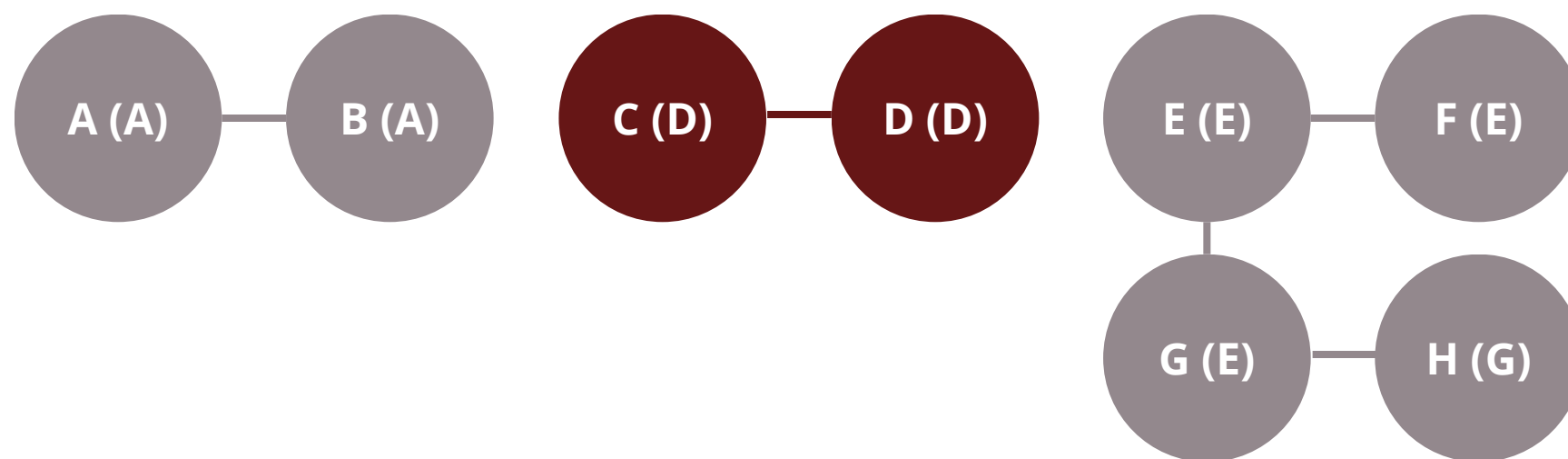
Animação - União (c/ Path Compression/Union by Rank)



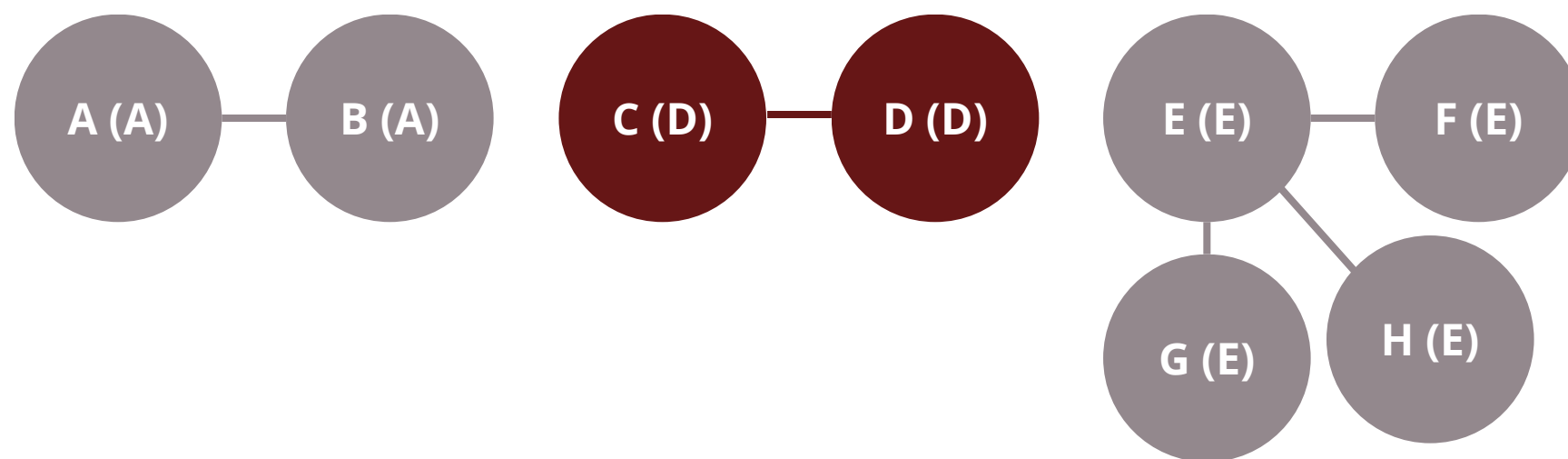
Animação - União (c/ Path Compression/Union by Rank)



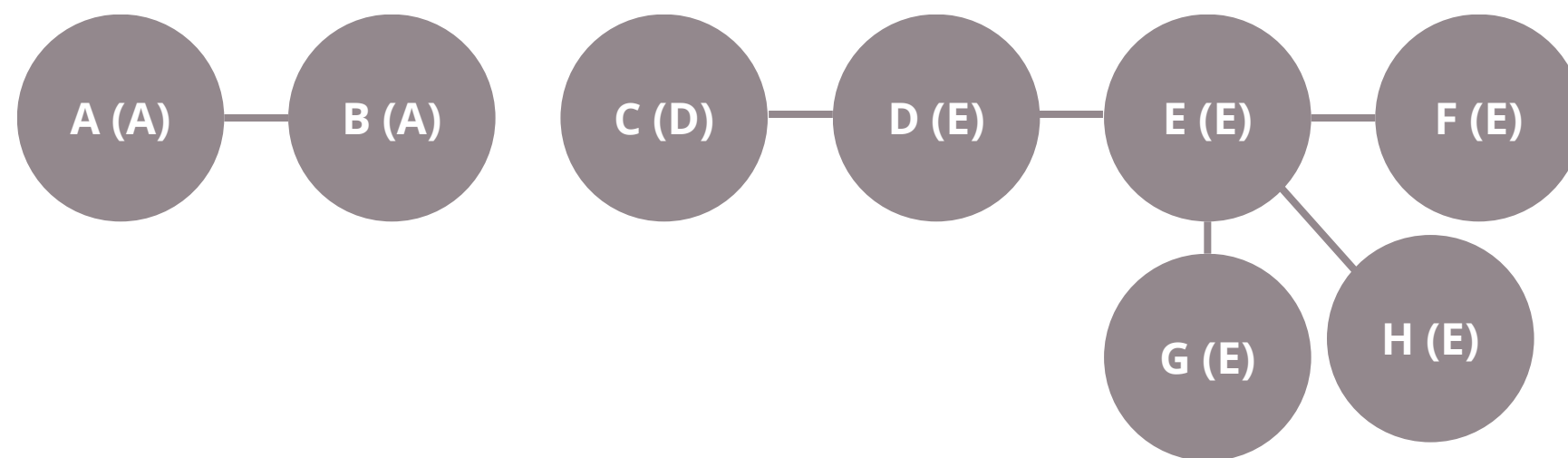
Animação - União (c/ Path Compression/Union by Rank)



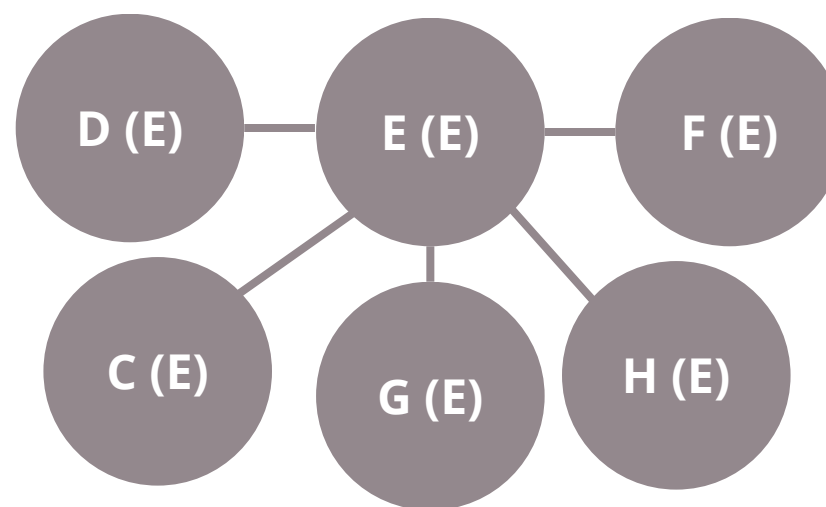
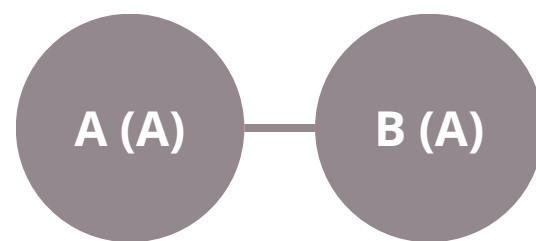
Animação - União (c/ Path Compression/Union by Rank)



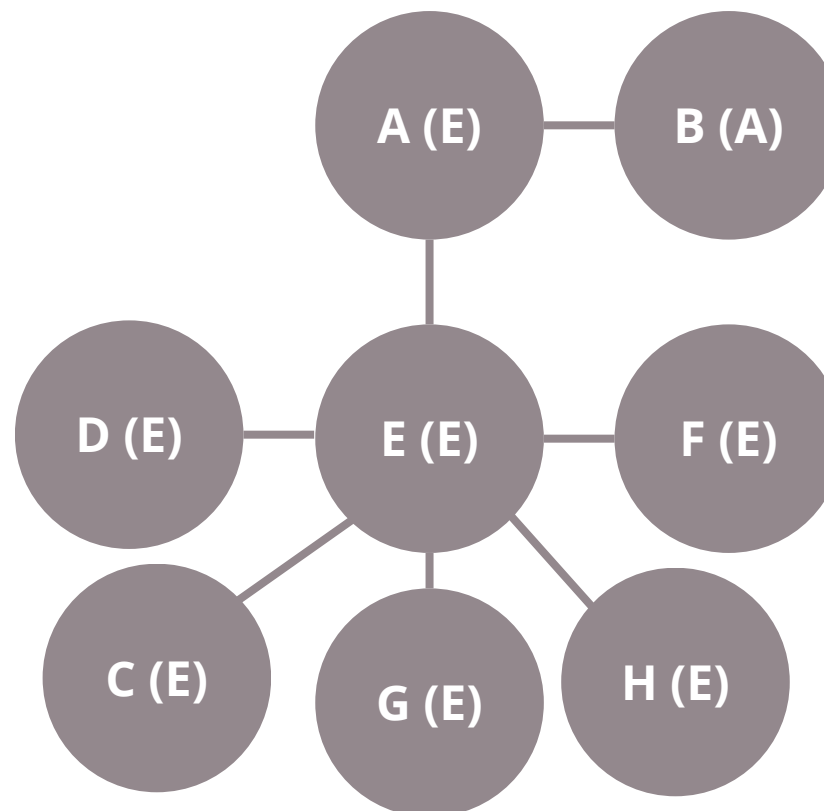
Animação - União (c/ Path Compression/Union by Rank)



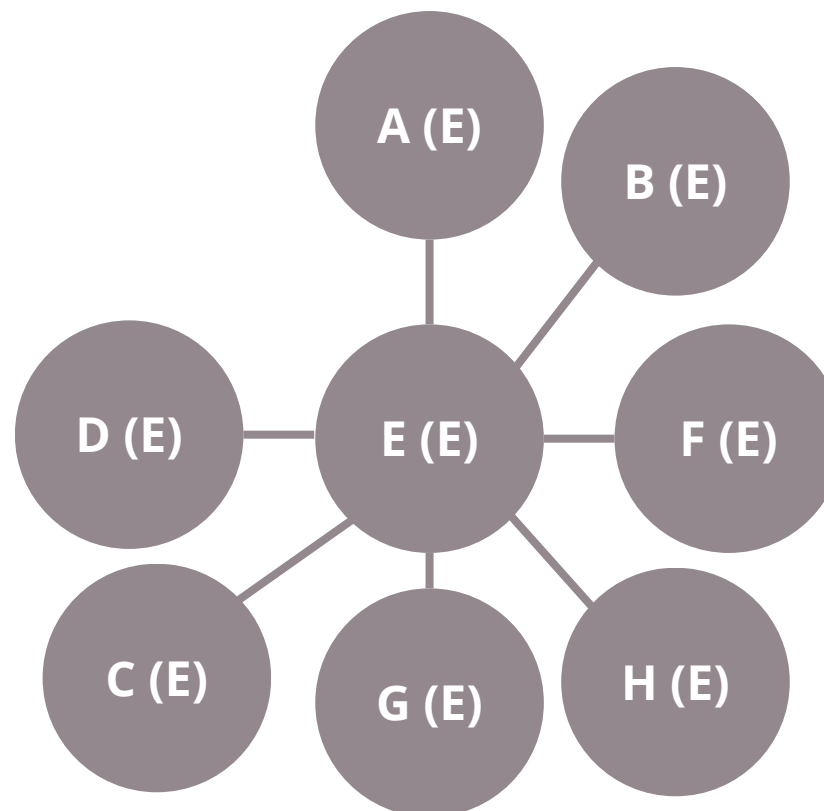
Animação - União (c/ Path Compression/Union by Rank)



Animação - União (c/ Path Compression/Union by Rank)



Animação - União (c/ Path Compression/Union by Rank)



De volta à Motivação...

- O problema das ilhas conectadas por pontes pode ser efetivamente resolvido utilizando a estrutura de dados chamada Union-Find devido à sua capacidade de agrupar elementos em conjuntos disjuntos e verificar se dois elementos pertencem ao mesmo conjunto.