# Análise Union Find

Compress Path e união por rank

# Rank:

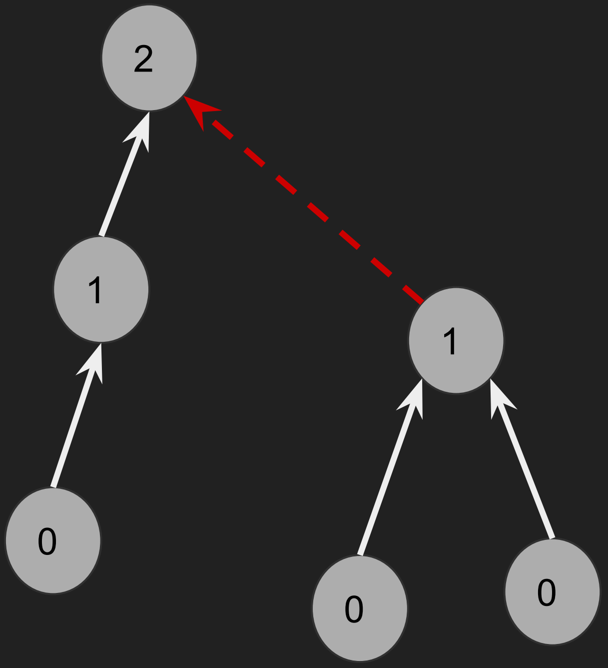
Para cada nó, nós mantemos um rank, que é um limite superior da altura daquele nó. Em outras palavras, é o número de arestas no caminho simples mais longo entre o vértice e uma folha.

## Union Operation:

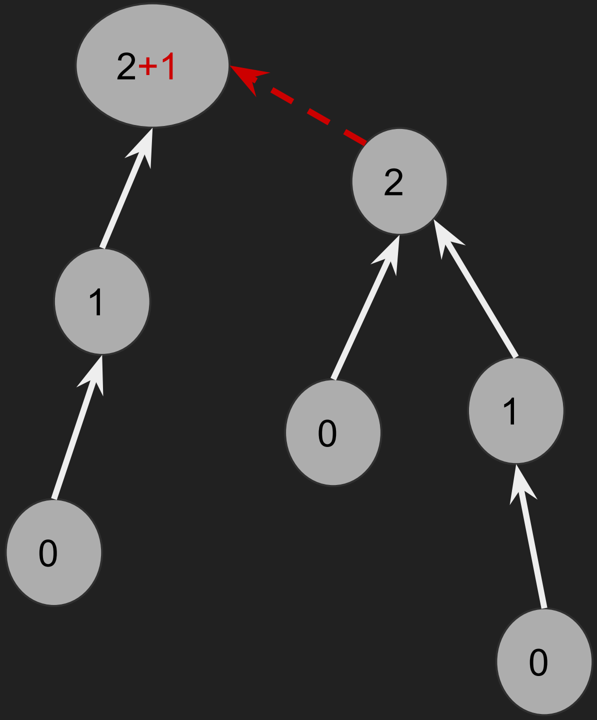
Se o rank das duas raízes são diferentes, a raiz com maior rank vira pai da raiz com menor rank. Os ranks delas não mudam.

Se o rank das duas raízes são iguais, escolhe-se arbitrariamente uma raiz para ser pai e incrementa-se o seu rank.

## Ranks diferentes




## Mesmo Rank



```
MAKE-SET(x)
    x.p = x
    x.rank = 0
```

*x.p* denota o pai de x

*x.rank* denota o rank de x

```
UNION(x, y)
    LINK(FIND-SET(x), FIND-SET(y))
```

```
LINK(x, y)
    if x.rank > y.rank
        y.p = x
    else x.p = y
        if x.rank == y.rank
            y.rank = y.rank + 1
```

```
FIND-SET(x)
    if x ≠ x.p
        x.p = FIND-SET(x.p)
    return x.p
```

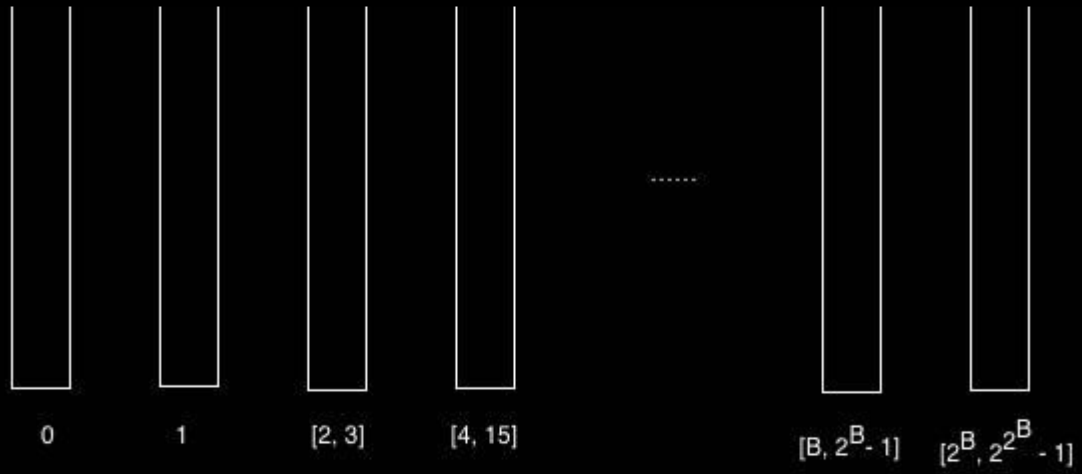
# Observações importantes

Lema 1: Na medida que a função find percorre o caminho até a raíz, o rank dos nós que ela encontra aumenta.

Lema 2: Um nó *u* que é raiz de uma subárvore com rank *r* tem no mínimo $2^r$ descendentes

Lema 3: O número de nós de rank r é de, no máximo, n/2r, onde n é o número total de elementos.

# Separando os nós em buckets por rank




- O número total de buckets é de log * n
- O número de nós dentro do bucket [B, $2^B$-1] é no máximo 2n / $2^B$
    - O número de nós dentro do bucket [B, $2^B$-1] é no máximo:

$$\frac{n}{2^B} + \frac{n}{2^{B+2}} + \frac{n}{2^{B+3}} + ... + \frac{n}{2^{2^B-1}} \leq \frac{2n}{2^B}$$

# Complexidade da operação find

Considerando *m* operações e *n* elementos.

$$T = T_1 + T_2 + T_3$$

$$T_1 = \sum (Ligacoes\ a\ raiz) = O(m)$$

$$T_2 = \sum (Numero\ de\ ligacoes\ entre\ diferentes\ buckets) = O(m\ log^* n)$$

$$T_3 = \sum (Numero\ de\ ligacoes\ entre\ o\ mesmo\ bucket)$$

# Complexidade da operação find

Para T3, suponha que temos uma transição de *u* a *v*, onde *u* e *v* estão no mesmo bucket $[B, 2^B - 1]$ e *v* não é a raiz. Fixando *u*, considere a sequência v1,v2,...,vk que fazem o papel de *v* em operações find diferentes.

- Por causa do path compression, essa sequência tem apenas nós diferentes.
- Por causa do Lema 1 os ranks dessa sequência são crescentes.
- Como todos os nós estão no mesmo bucket, o comprimento *k* da sequência (O número de vezes que o nó *u* é ligado a uma raiz diferente no mesmo bucket) é no máximo o número de ranks diferentes em B, ou seja, no máximo $2^B - 1 - B < 2^B$.

# Complexidade da operação find

$$T_3 \leq \sum_{[B, 2^B - 1]} \sum_{u} 2^B$$

$$T_3 \leq \sum_{B} 2^B \frac{2n}{2^B} \leq 2n\ log^* n$$

# Complexidade da operação find

Considerando *m* operações e *n* elementos.

$$T = T_1 + T_2 + T_3$$

$$T_1 = \sum (Ligacoes\ a\ raiz) = O(m)$$

$$T_2 = \sum (Numero\ de\ ligacoes\ entre\ diferentes\ buckets) = O(m\ log^*n)$$

$$T_3 = \sum (Numero\ de\ ligacoes\ entre\ o\ mesmo\ bucket) = O(n\ log^*n)$$

$$O(m\ log^*n)$$