Disjoint-set/Union-find Forest

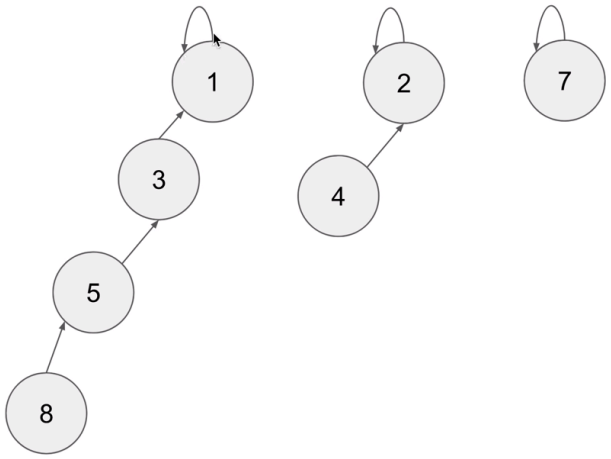
Find(x): find the root/cluster-id of x
Union(x, y): merge two clusters

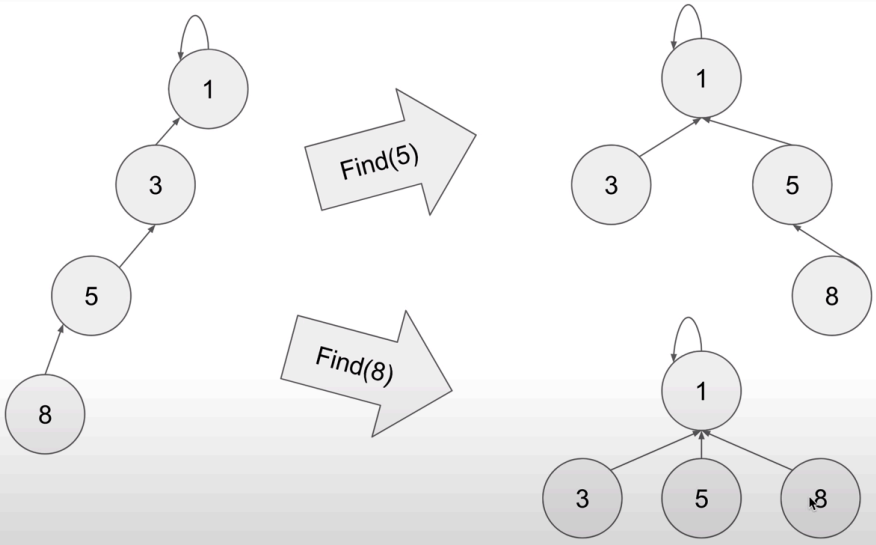
Check whether two elements are in the same set or not in O(1)*.

Find: O(α(n))* ≈ O(1)
Union: O(α(n))* ≈ O(1)
Space: O(n)

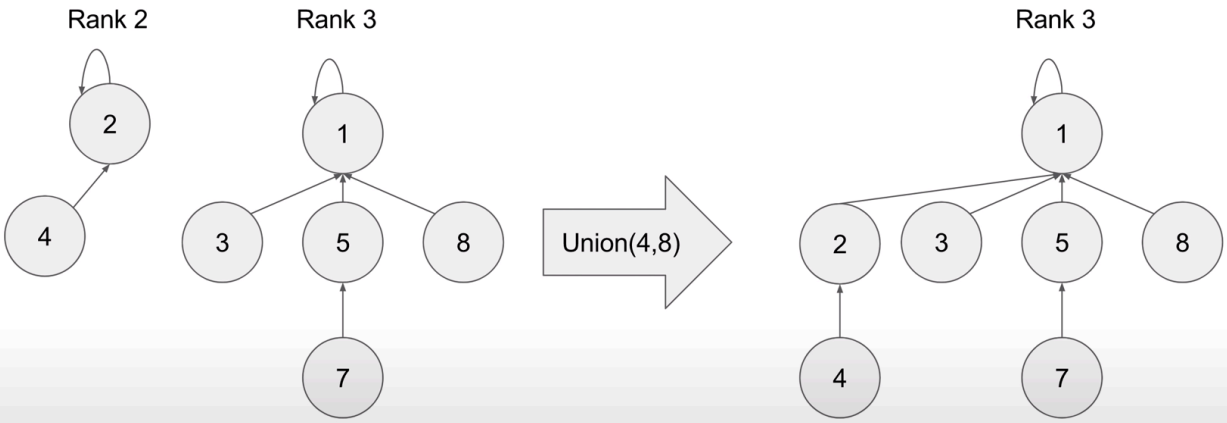
Without optimization: Find: O(n)
Two key optimizations:
1) Path compression: make tree flat
2) Union by rank: merge low rank tree to high rank one

*: amortized
α(.): inverse Ackermann function





Optimization 1: Path compression, happened during Find



Optimization 2: Union by rank, merge low rank tree into high rank one
If two sub-tree has the same rank, break tie arbitrarily and increase the merged tree's rank by 1
Reduce path compression overhead

Pseudo code:

```
class UnionFindSet:
  func UnionFindSet(n):
    parents = [1..n]
    ranks = [0..0] (n zeros)

  func Find(x):
    if x != parents[x]:
      parents[x] = Find(parents[x])
    return parents[x]

  func Union(x, y):
     px, py = Find(x), Find(y)
     if ranks[px] > ranks[py]: parents[py] = px
     if ranks[px] < ranks[py]: parents[px] = py
     if ranks[px] == ranks[py]:
       parents[py] = px
       ranks[px] ++
```