Disjoint sets

Ch. 5.10

Disjoint sets

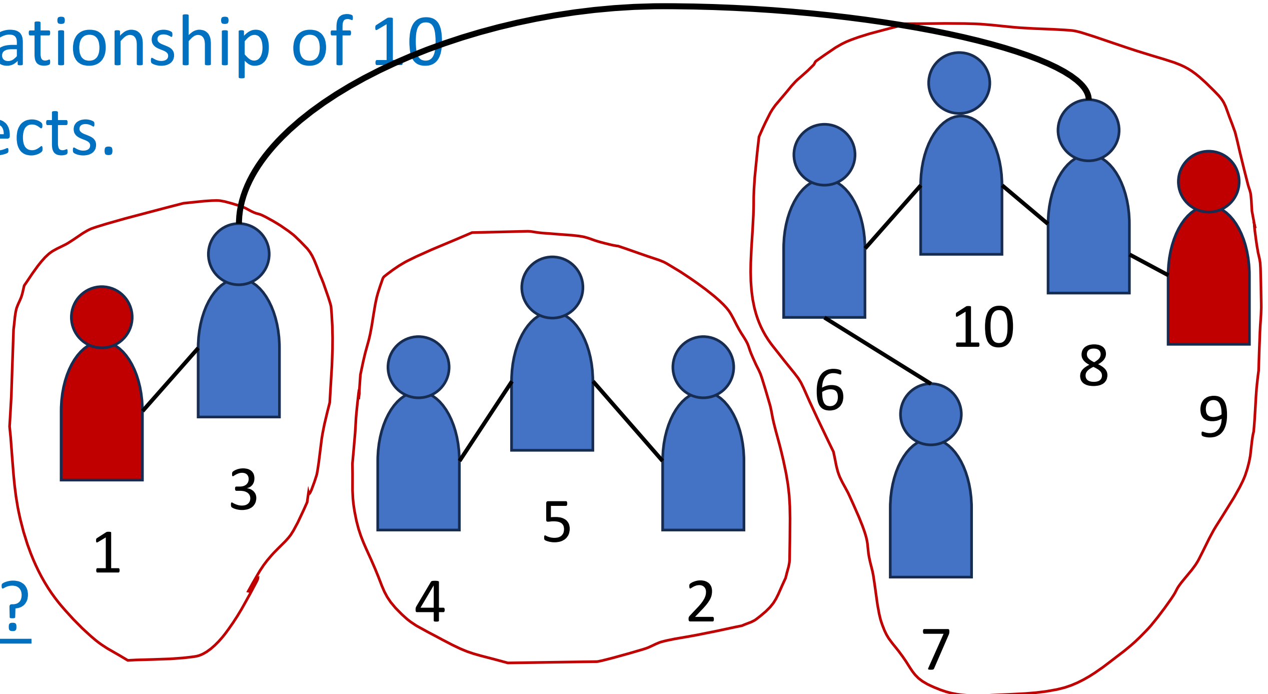
- No element appears in multiple sets.
- Given 10 elements, from 0 to 9,

S1 = {0, 6, 7, 8}	S2 = {1, 4, 9}	S3 = {2, 3, 5}	Disjoint sets
S1 = {0, 6, 7, 8, 9}	S2 = {0, 1, 4, 9}	S3 = {1, 2, 3, 5}	**NOT** disjoint sets

EXAMPLE

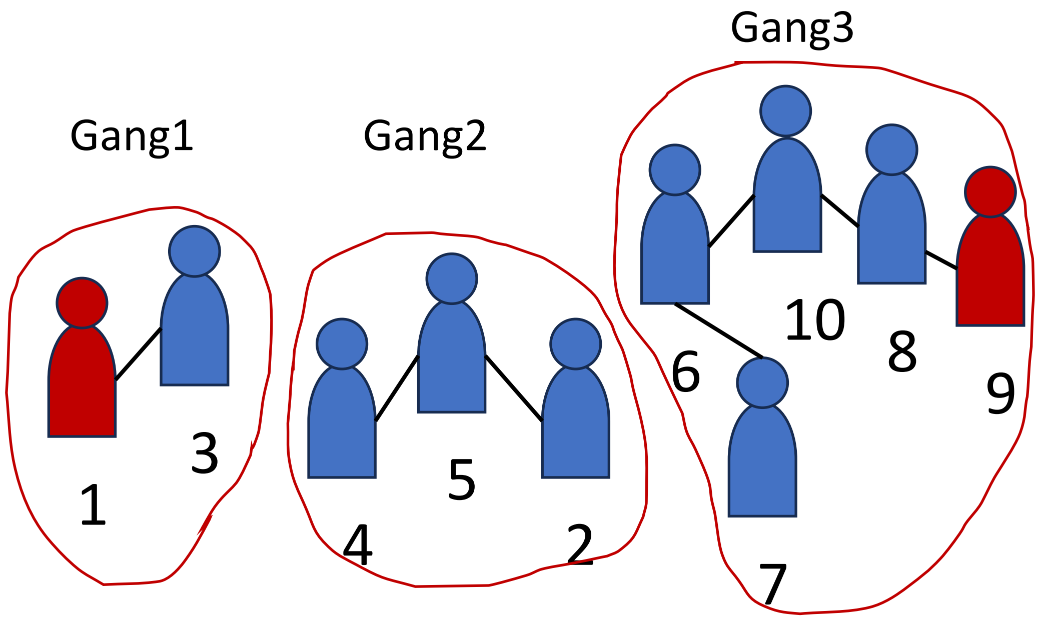
You are police officers. You know the relationship of 10 gang members. No.1 and No.9 are suspects.
<u>Do they belong to the same gang?</u>

Then you find evidence showing that gangs of No.8 and No.3 are merged. <u>Do No.1 and No.9 belong to the same gang?</u>

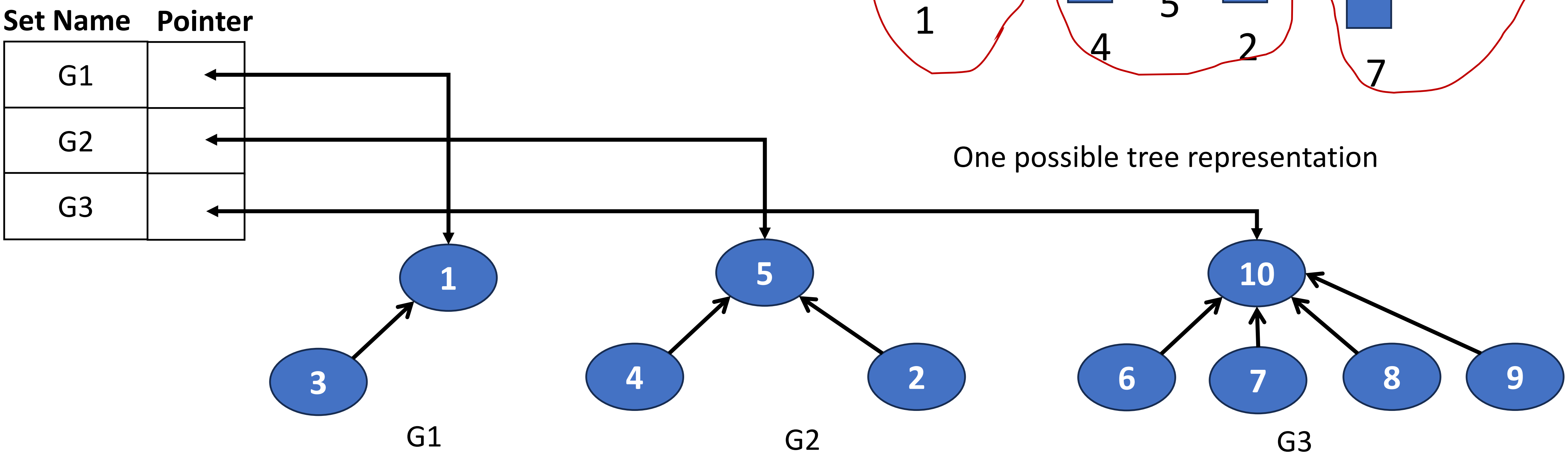


Data representation of disjoint sets

- Link the nodes from the children to the parent.
- Each root has a pointer to the set name.



One possible tree representation



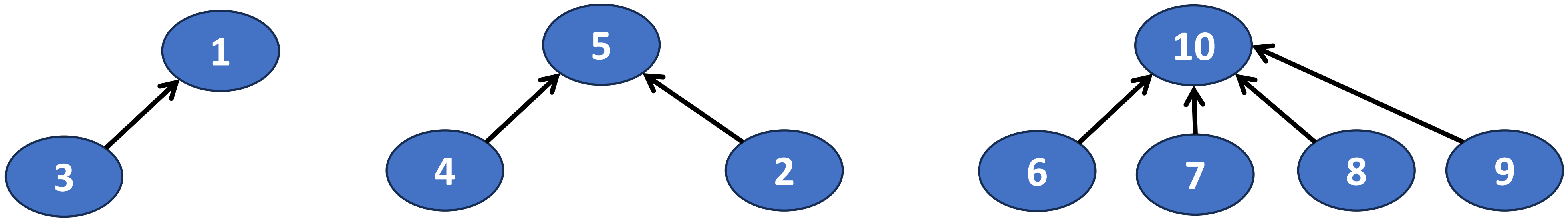
Operation: find(i)

- Find the set containing the targeted element

  S1 = {0, 6, 7, 8} S2 = {1, 4, 9} S3 = {2, 3, 5} Example: 8 is in set S1.

- Start at the node representing element *i* and climb up the tree until the root is reached. Then return the element in the root.

- Using an integer array to store the parent of each element.



	0	1	2	3	4	5	6	7	8	9	10	11		*N*
parent[]		-1	5	1	5	-1	10	10	10	10	-1		...	

Root node

Find the set containing 9.

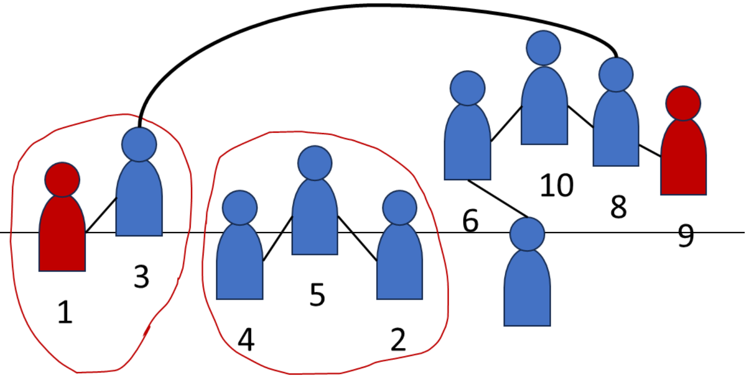
Operation: disjoint set union

- Combine two disjoint set into one

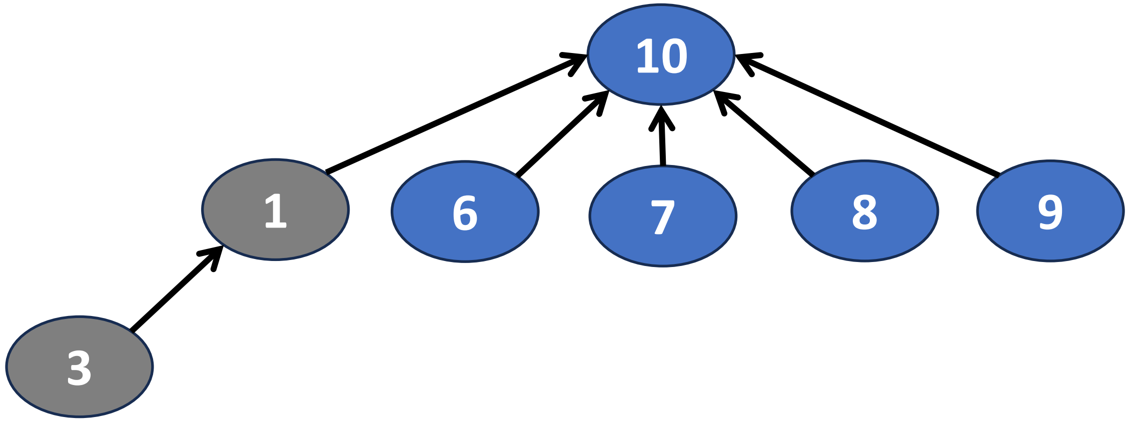
$G1 = \{1, 3\} \qquad G3 = \{6,7,8,9,10\} \xrightarrow[\text{union}]{} G1 \cup G3 = \{1,3,6,7,8,9,10\}$

- For tree representation, we set the parent field of one root to the other root.

EXAMPLE

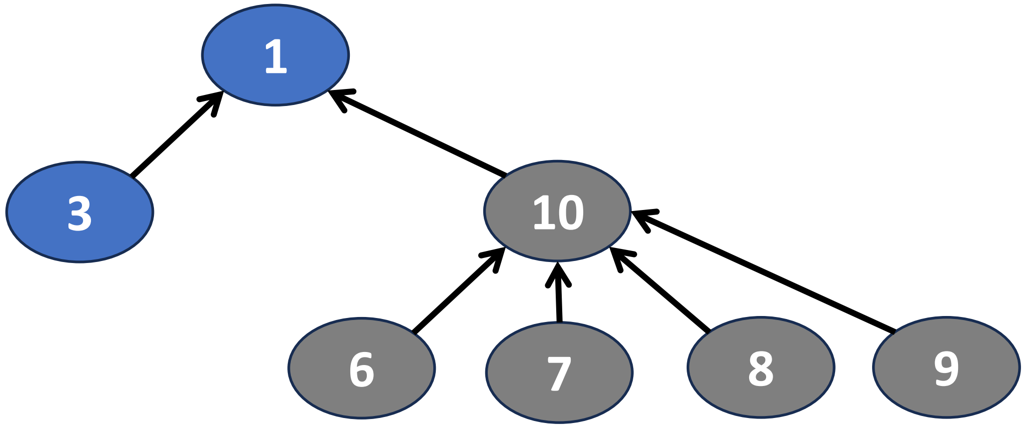


Merge the gangs of No.8 and No.3.



G1 U G3

OR



G1 U G3

Operation: union(i,j)

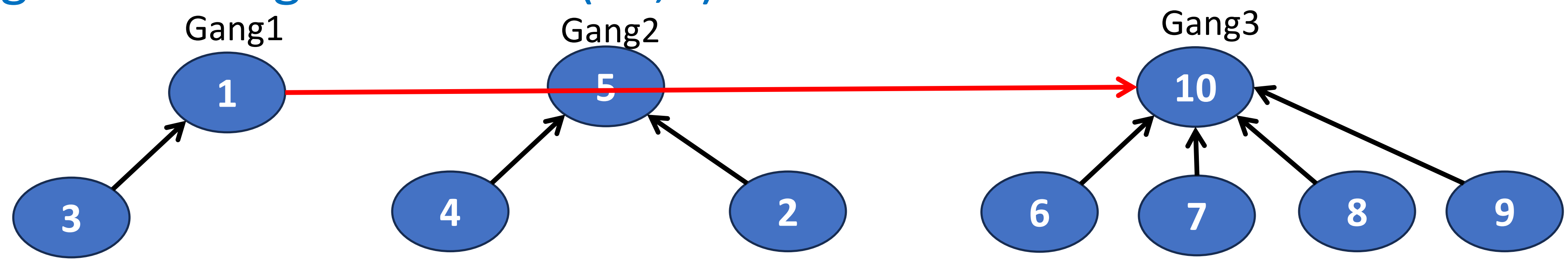
- i and j are the roots of two different trees, i != j
- To unite the trees, make one tree as a subtree of the other.

parent[j] = i

- Time complexity: O(1)

EXAMPLE

Merge Gang1 and Gang3. → Union(10,1)



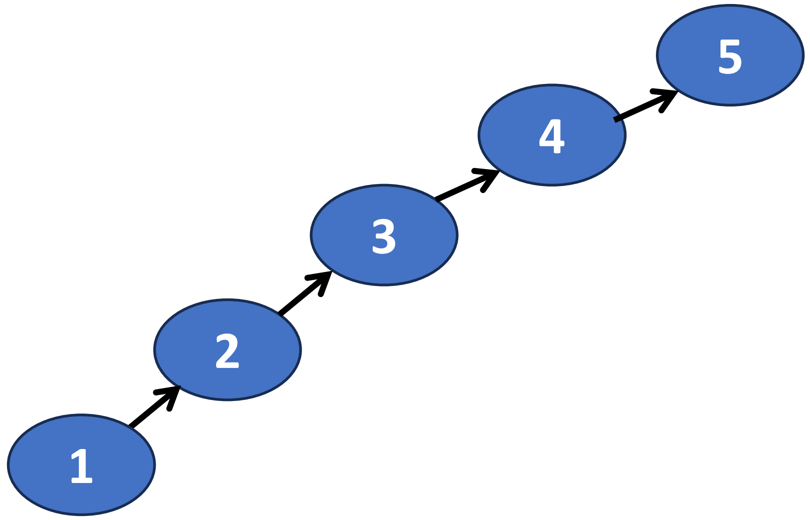
	0	1	2	3	4	5	6	7	8	9	10	11	...	*N*
parent[]		10	5	1	5	-1	10	10	10	10	-1		...	

Operation: find(i)

```
while (parent[i] >= 0)
        i = parent[i];  // move up the tree
return i;
```

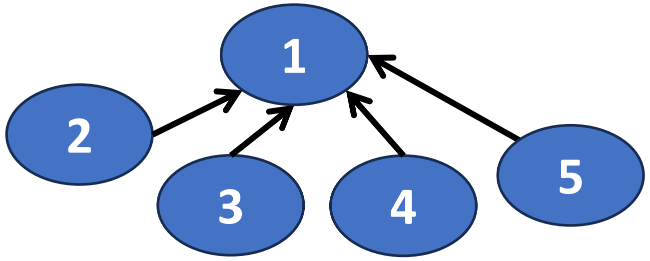
- Time complexity? Depending on the level of i

After union(2,1), union(3,2), union(4,3), union(5,4)



Find(2): climbing up 3 times.

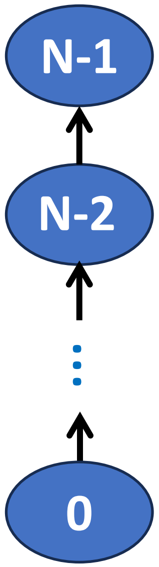
After union(1,2), union(1,3), union(1,4), union(1,5)



Find(2): climbing up once.

Sequence of union-find operations

union(1,0), find(0)
union(2,1), find(0)
⋮
union(N-1,N-2), find(0)



It produces a degenerate tree.

For each find(0), we trace from 0 to the root.

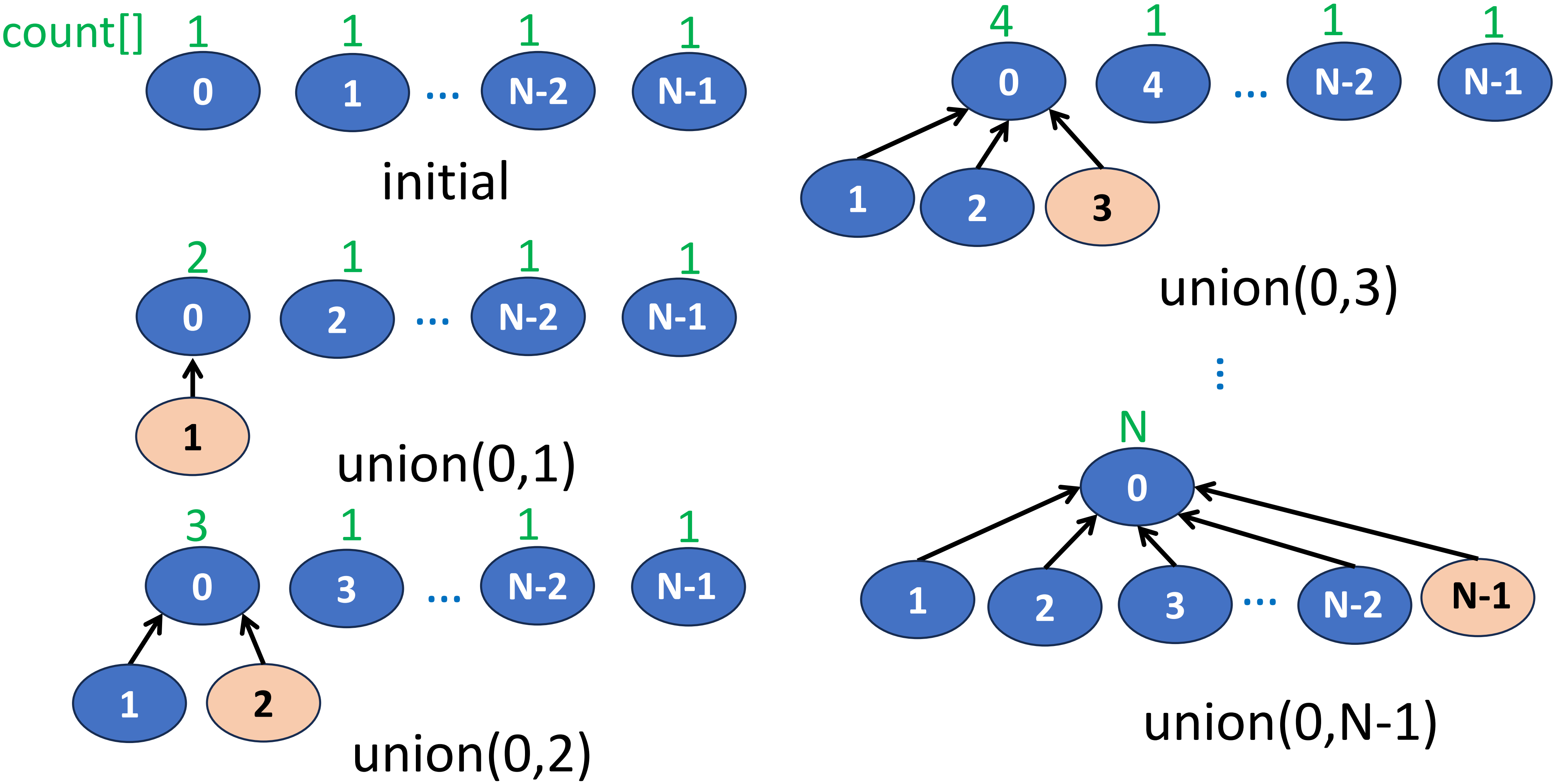
- Time complexity
  - Time to initialize parent[i]=0 for all elements: O(n)
  - n-1 times of union: Each union takes O(1). Total time is O(n).
  - n-1 times of find: Each find takes i. Total time is $\sum_{i=2}^{n} i$ =O(n^2).

Total time: O(n^2)

How to avoid the creation of degenerate tree?

Weight rule for union(i,j)

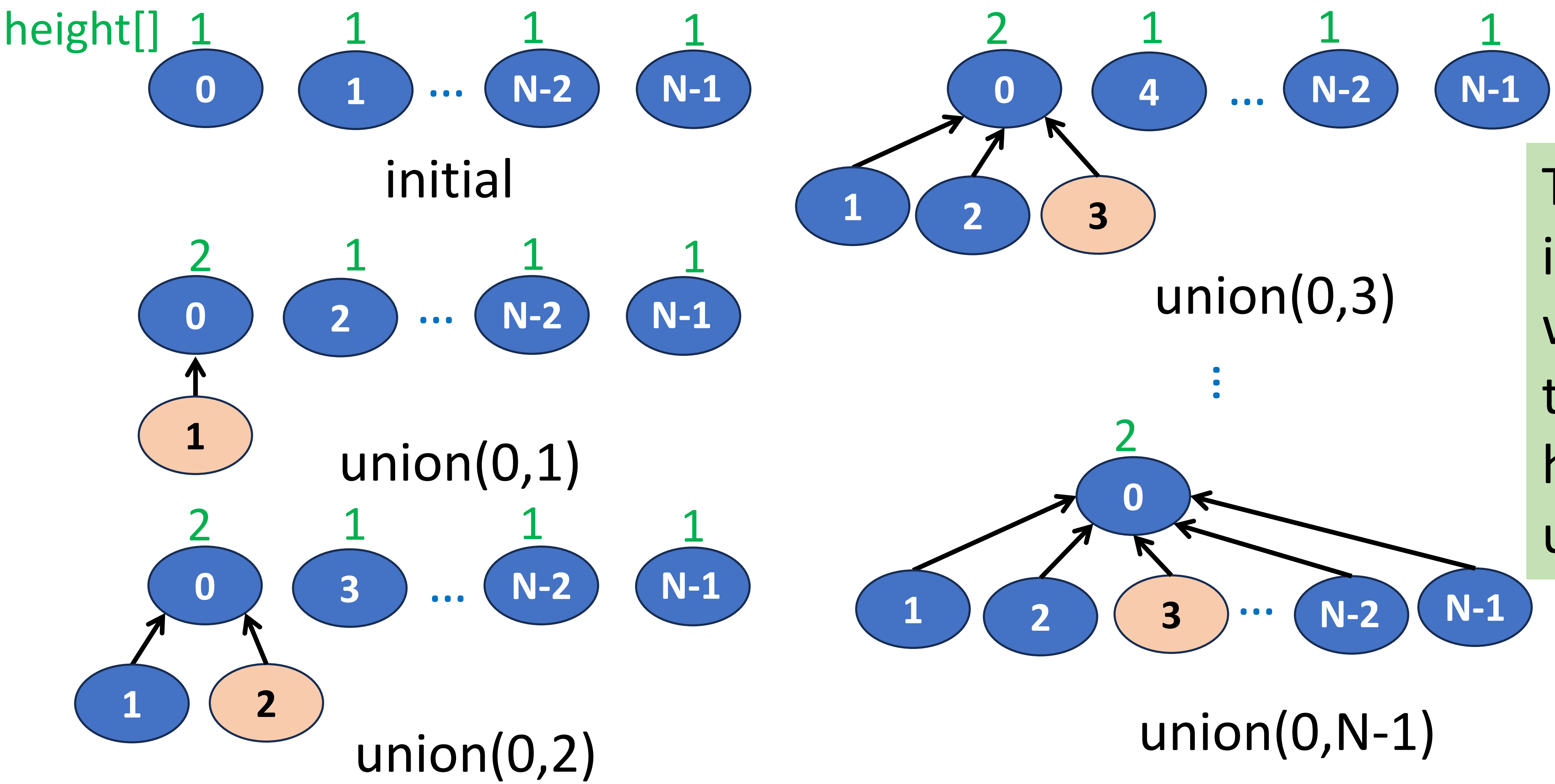
- Make tree with fewer number of elements a subtree of the other tree.



The count of the new tree is the sum of the counts of the trees that are united.

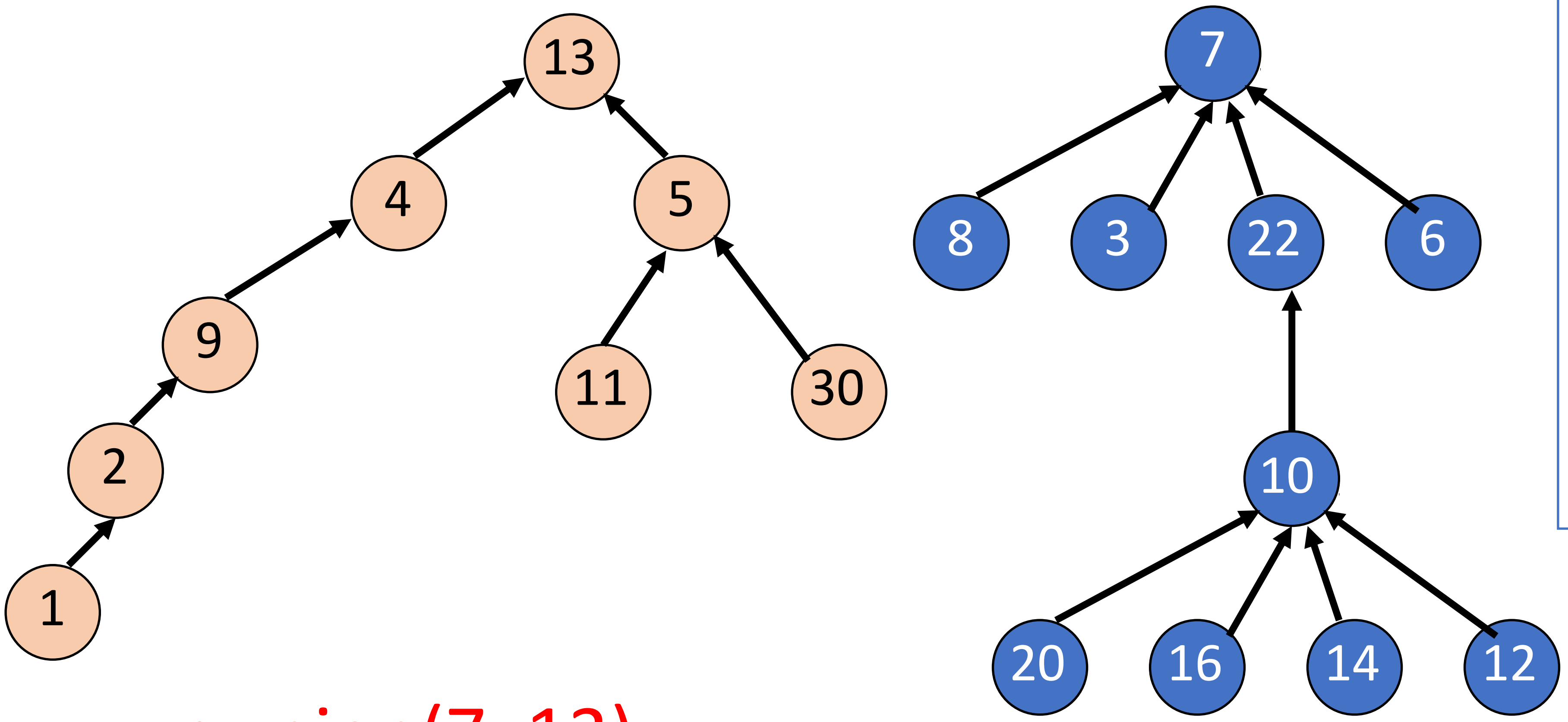
Height rule

- Make tree with smaller height a subtree of the other tree.



The height increases only when two trees of equal height are united.

Exercise



Please reply your answers of Q4 and Q5 via the following link:

Group members: 1~3 people

- union(7, 13)

Which tree should become a subtree of the other?

Q4: Write out the answer based on **weight** rule.

Q5: Write out the answer based on **height** rule.

Height of a tree

Lemma 5.5

- Suppose we start with single element trees and perform unions using either the **height rule** or the **weight rule**.

- The height of a tree with p elements is at most floor $(\log_2 p) + 1$

Processing an intermixed sequence of u-1 unions and f finds

Time complexity is $O(u + f \log u)$

Requires at most f*[floor($\log_2 u$) + 1]

Generating a tree with u nodes.

Further improving find(i)

- Shorten the path of nodes by ***collapsing rule***:

Make all nodes on find path point to tree root.

EXAMPLE

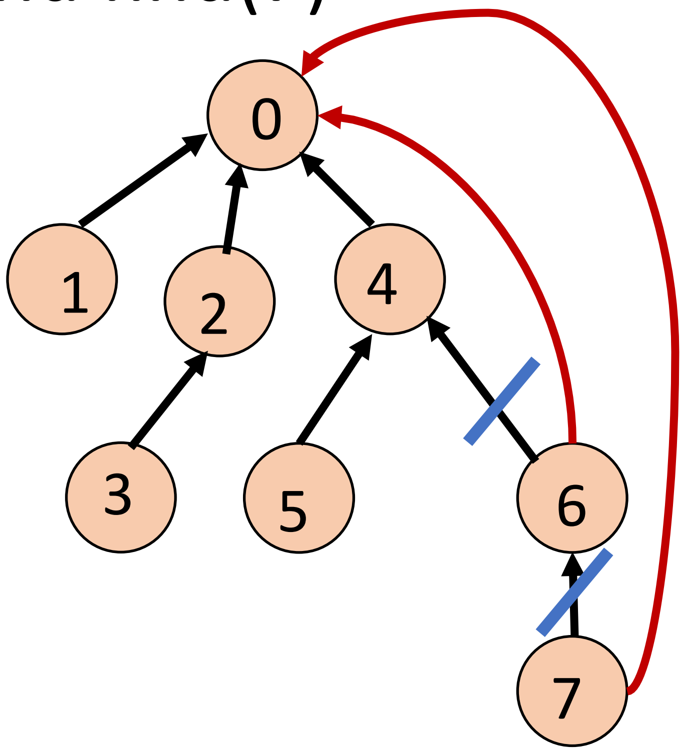
A sequence of find(7), find(7), find(7), find(7) and find(7)

- <u>Without</u> collapsing rule:

A find(7) requires climbing up 3 times.
Total: 5*3 = 15 moves.

- <u>With</u> collapsing rule:

The first find(7) requires climbing up 3 times.
The remaining 4 find(7) requires climbing up once.
Total: 3+4*1=6 moves



Time complexity

Lemma 5.6 [Tarjan and Van Leeuwen]

- Let $T(f,u)$ be the maximum time required to process any intermixed sequence of f finds and u unions. Assume that $u \geq n/2$.

$$k_1 * (n + f * \underline{\alpha(f+n, n)}) \leq T(f,u) \leq k_2 * (n + f * \underline{\alpha(f+n, n)})$$

where k_1 and k_2 are constants.

A slowly growing function related to Ackermann's function

- These bounds apply when we start with singleton sets and use either the weight or height rule for unions and collapsing rule for a find.

Time complexity is close to $O(n + f)$

Summary

- What are disjoint sets?

- Disjoint set union and find

    Time complexity for u unions and f finds: $O(nf)$

  + Weight rule or height rule for union

    Time complexity for u unions and f finds: $O(n + f \log n)$

  + Collapsing rule for find

    Time complexity for u unions and f finds: close to $O(n + f)$