Binomial Heaps

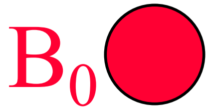
Leftist vs Binomial Heaps
(Leftist not given in this class)

	Binomial heaps	
	Actual	Amortized
Insert	O(1)	O(1)
Delete min (or max)	O(n)	O(log n)
Meld	O(1)	O(1)

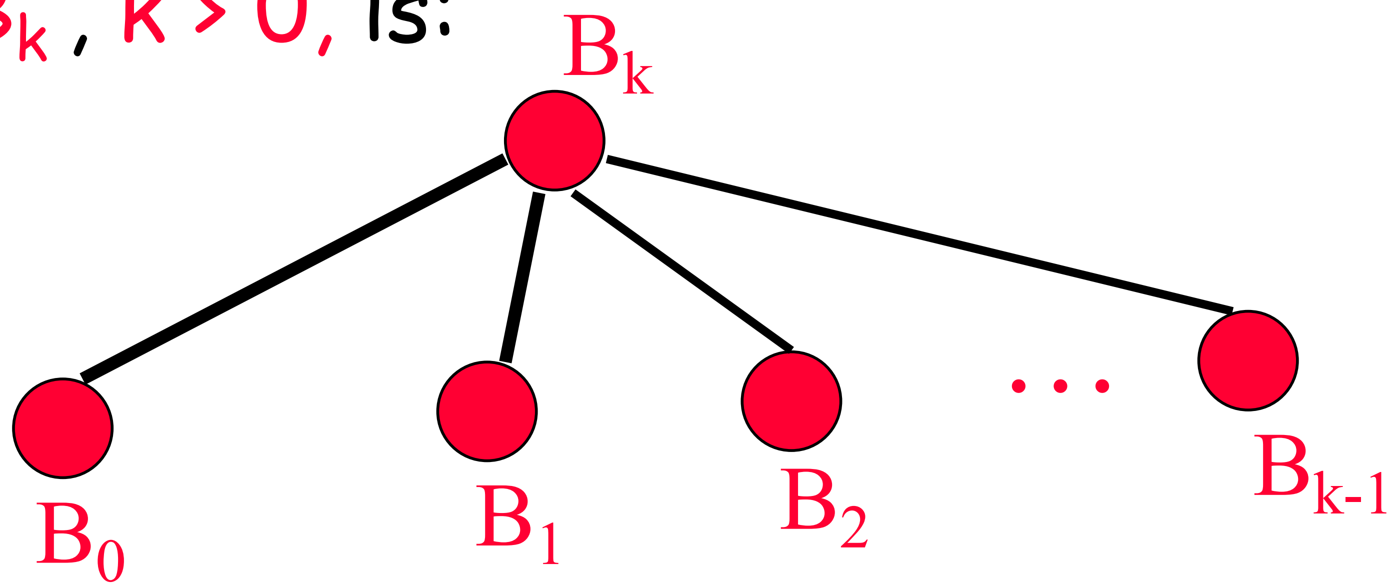
不支援search!!!

Binomial Trees

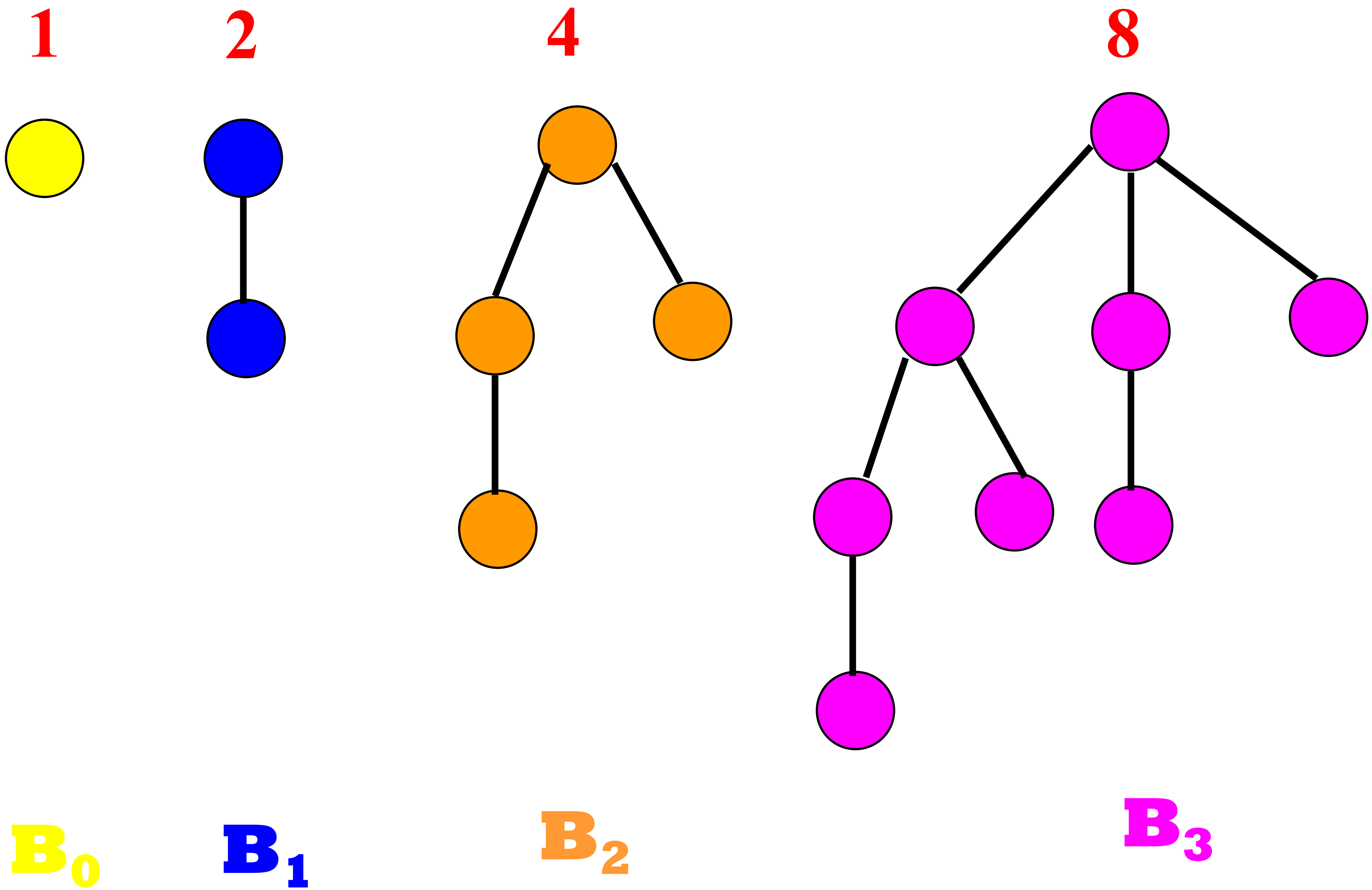
- B_k is degree k binomial tree.



- B_k , $k > 0$, is:

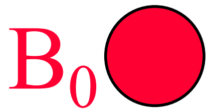


Examples



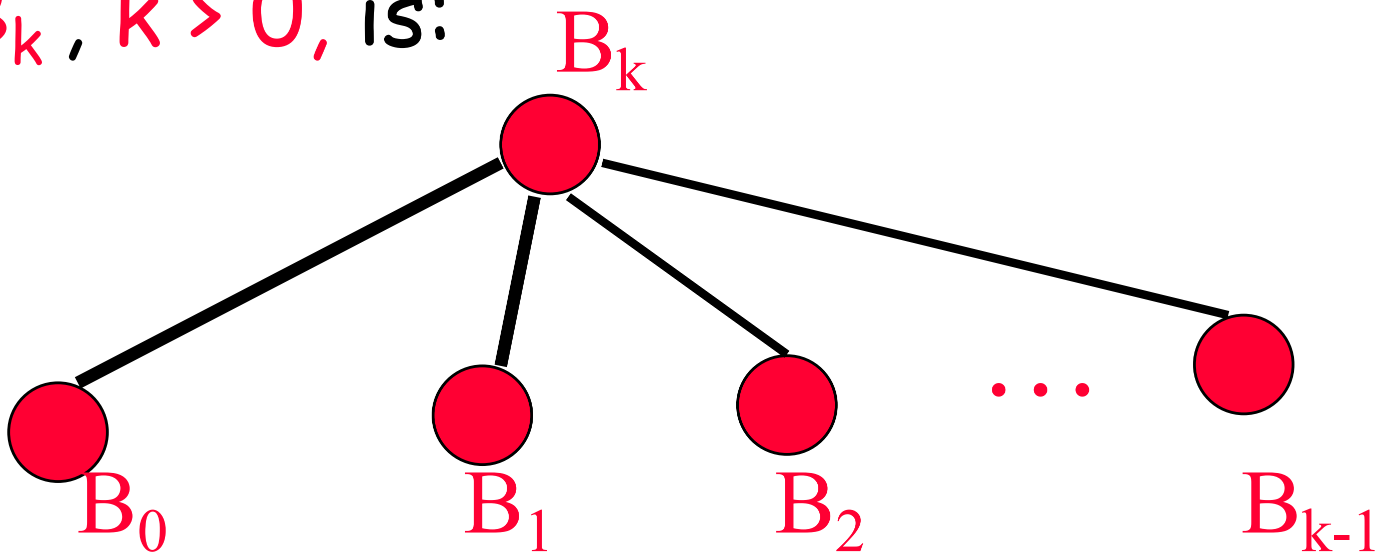
Some Properties: Number of Nodes in B_k

- N_k = number of nodes in B_k.



$N_0 = 1$

- B_k , $k > 0$, is:

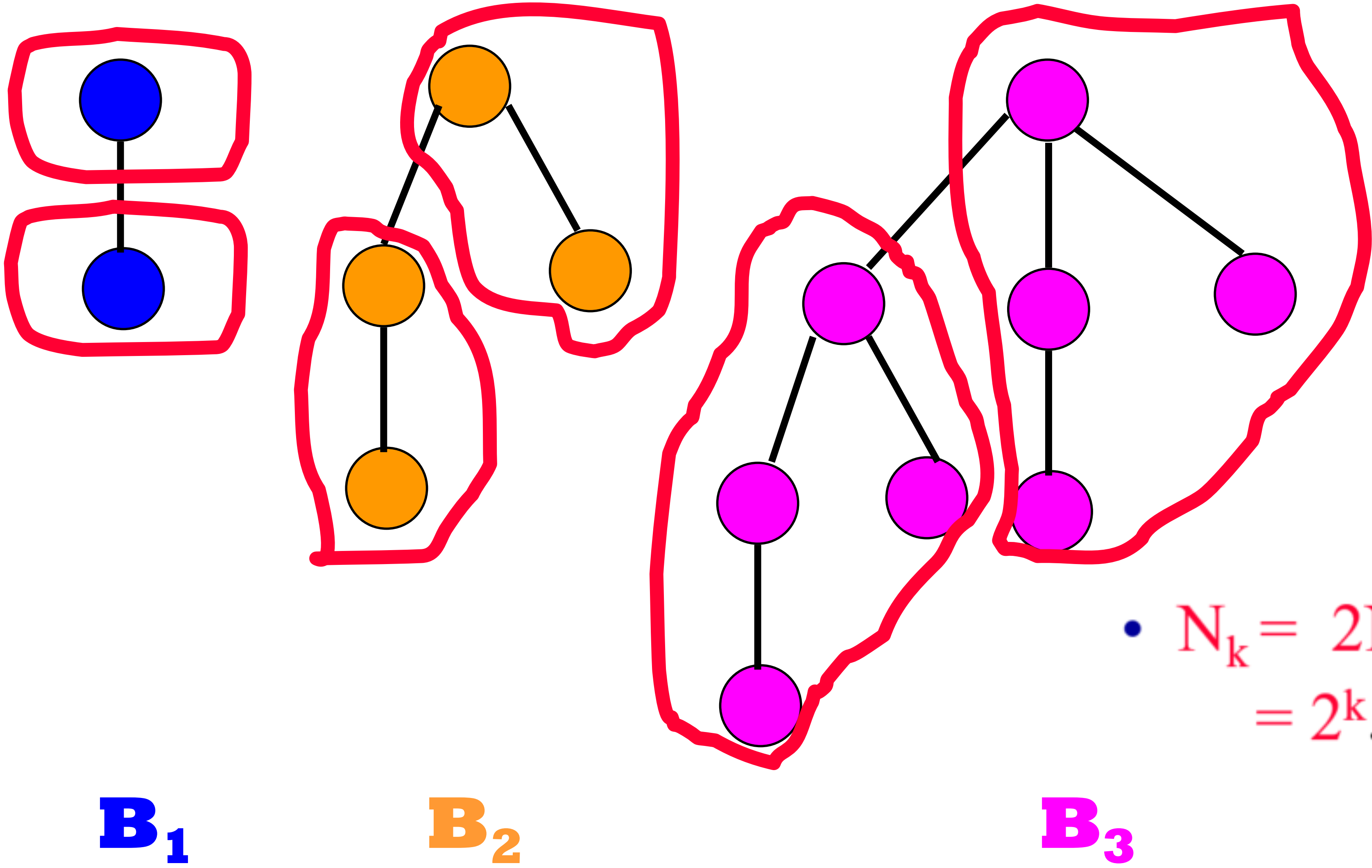


- $N_k = N_0 + N_1 + N_2 + \ldots + N_{k-1} + 1$ // 1: B_k的root
  $= 2^k$.

Equivalent Definition

(另外一個重要的角度來看Binomial Trees)

- B_k , $k > 0$, is two B_{k-1}s.
- One of these is a subtree of the other.



- $N_k = 2N_{k-1}$
  $= 2^k$.

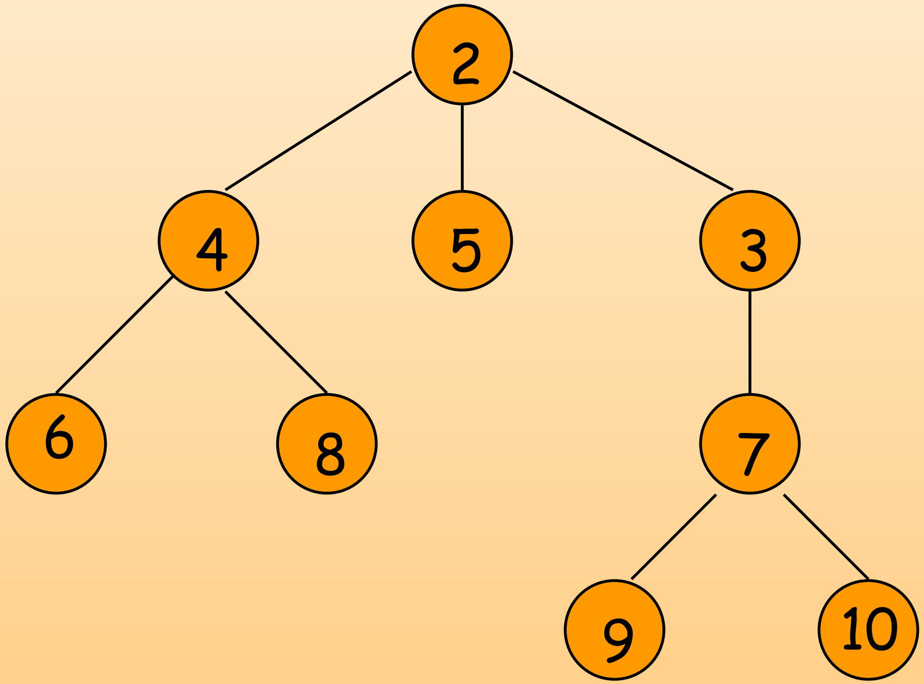
Min Binomial Heap (Definition)

- Collection of min trees.
- Each min tree is a binominal tree.
- Different binomial trees in the collection have non-identical number of degrees (of their roots)
  - Hopefully

補充：Min Tree

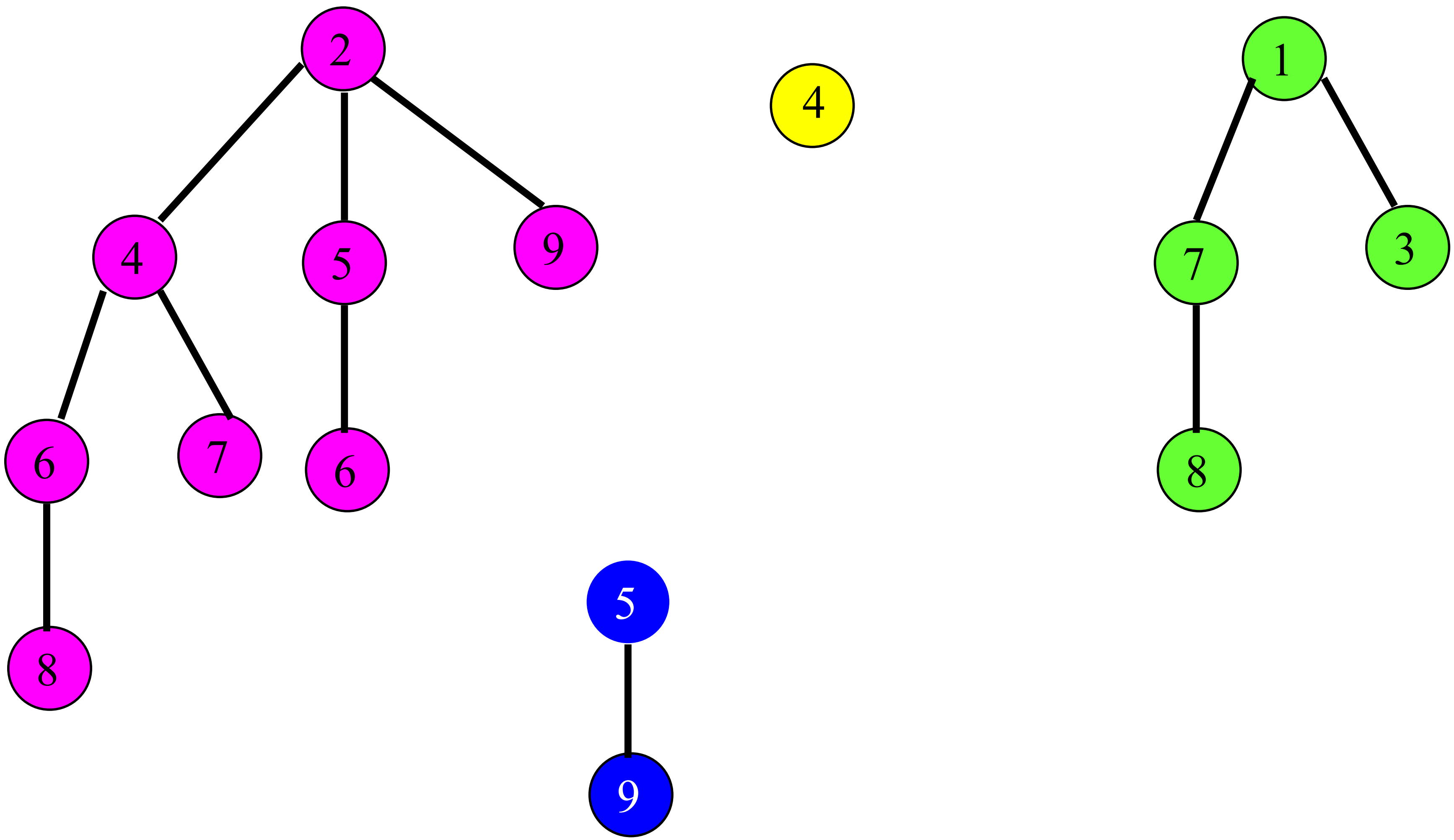
- Each tree node has a value
- Value in any node is the minimum value in the subtree

補充：Min Tree Example



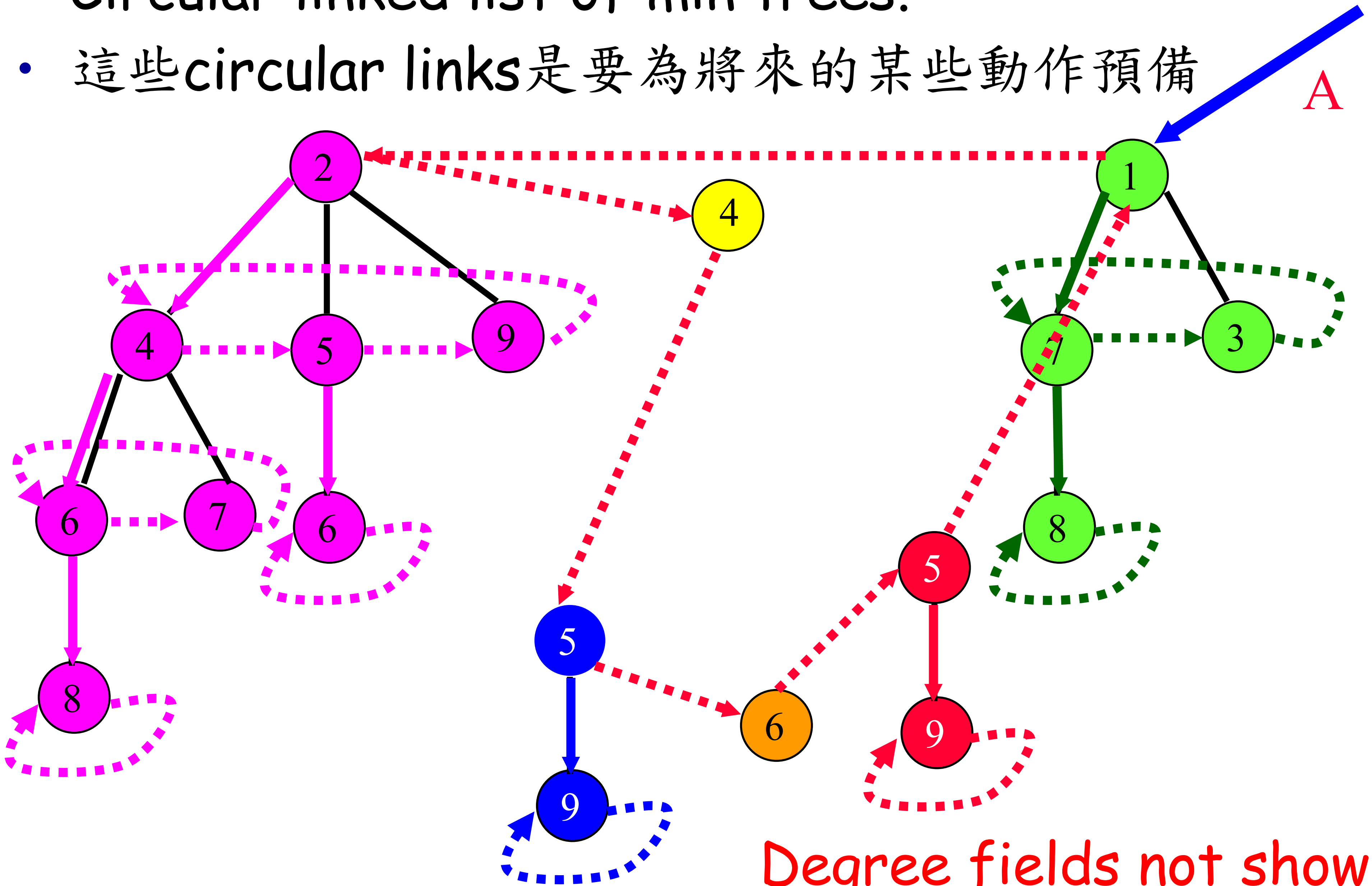
Root has minimum element.

Min Binomial Heap (Example)



Binomial Heap Representation

- Circular linked list of min trees.
- 這些circular links是要為將來的某些動作預備



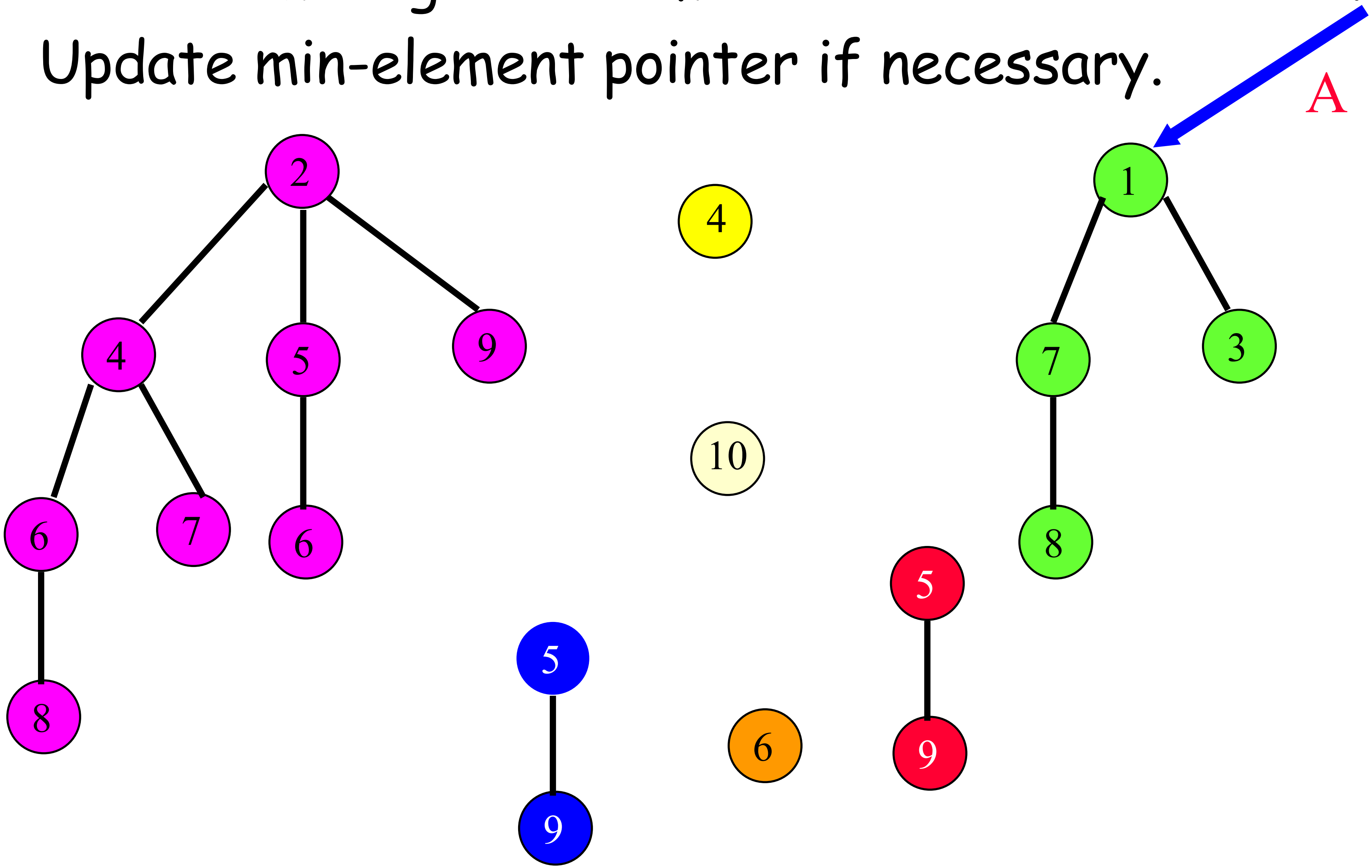
Degree fields not shown.

Per Node Structure

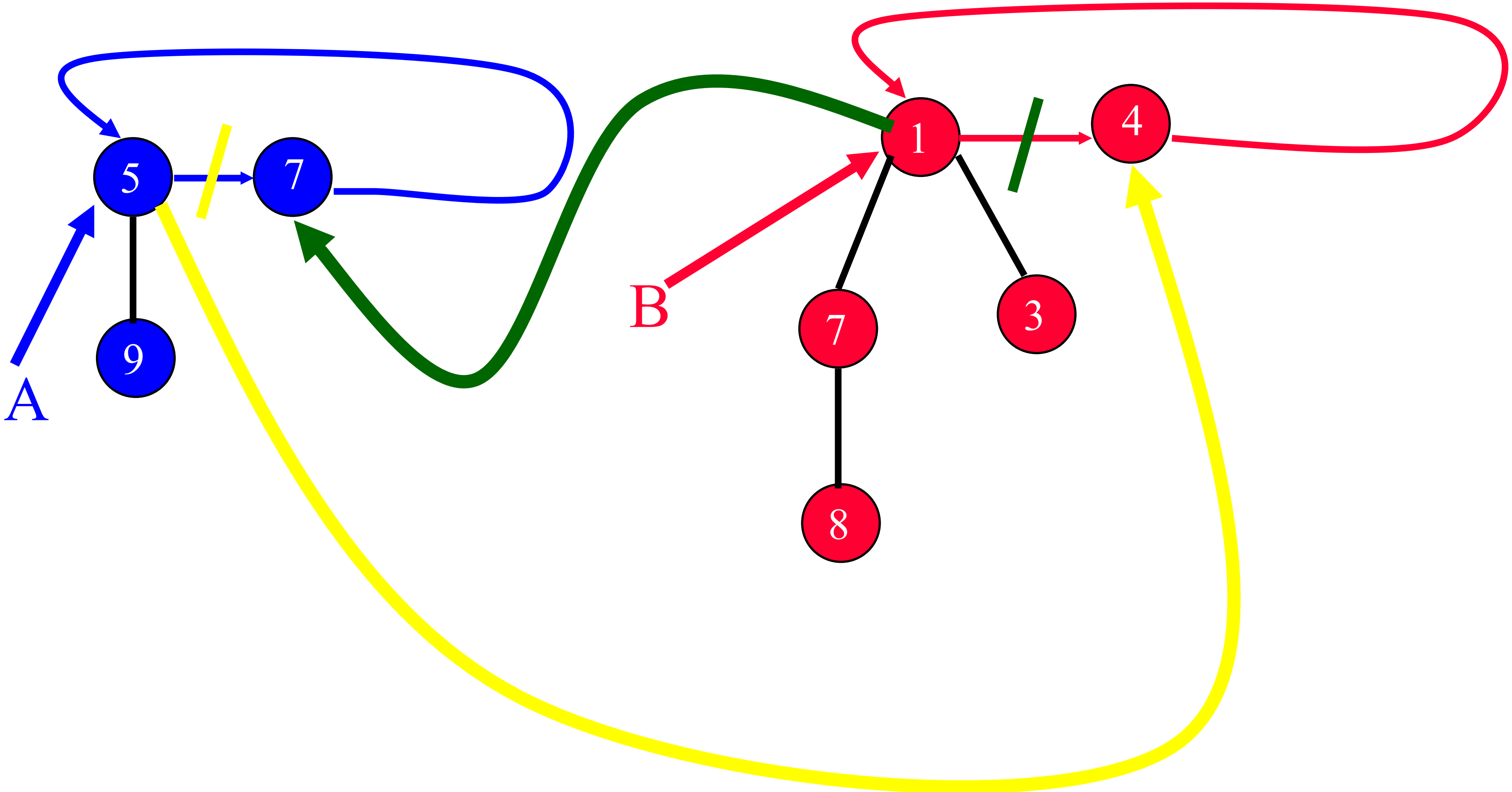
- Degree
  - Number of children.
- Child // only one pointer
  - Pointer to one of the node's children.
  - Null iff node has no child.
- Sibling // circular lists
  - Used for circular linked list of siblings.
- Data

Insert 10 (為了乾淨表達，我們省略了所有的虛線)

- Add a new single-node min tree to the collection.
- Update min-element pointer if necessary.

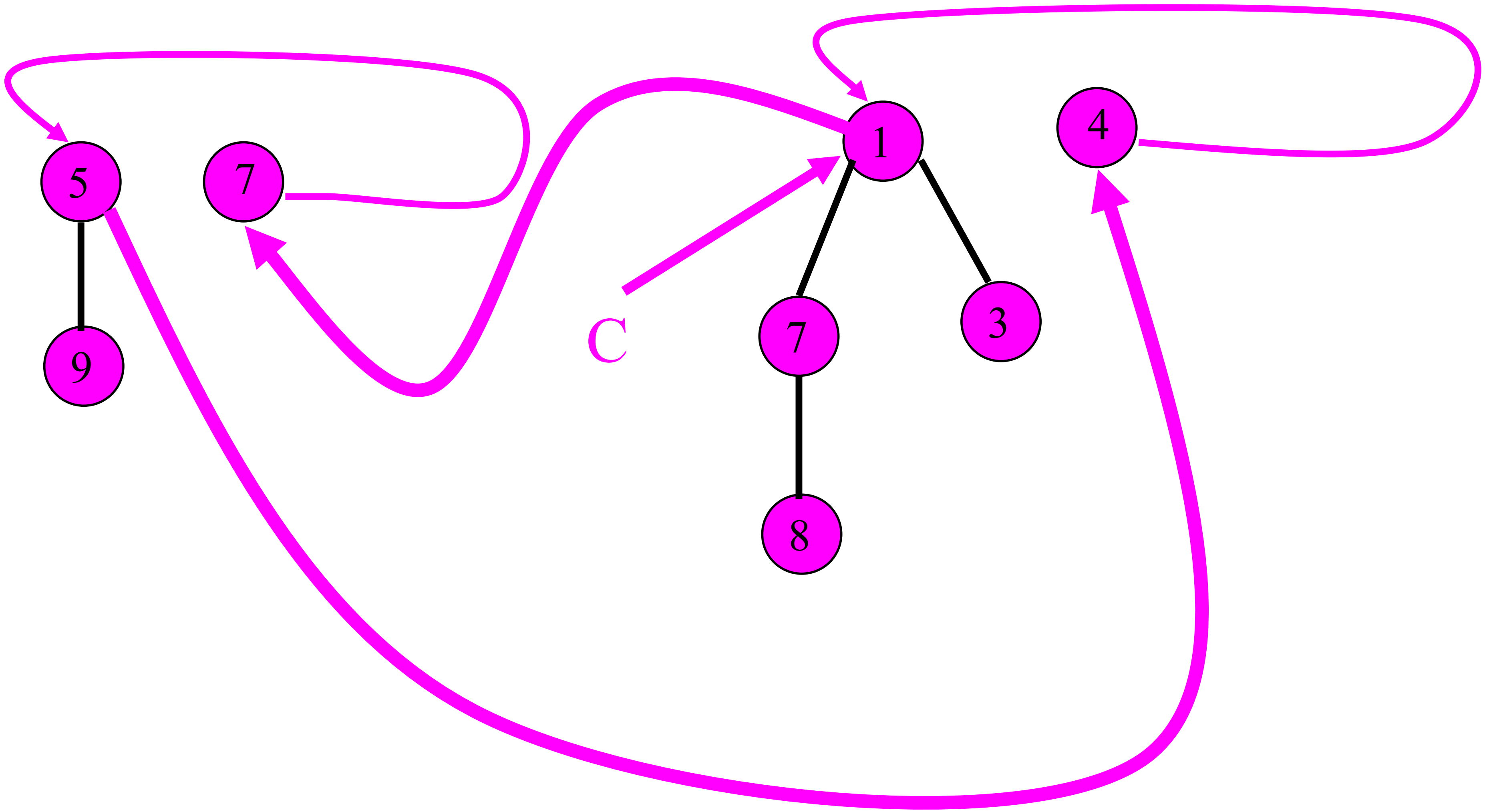


Meld



- Combine the 2 top-level circular lists.
- Set min-element pointer.

Meld

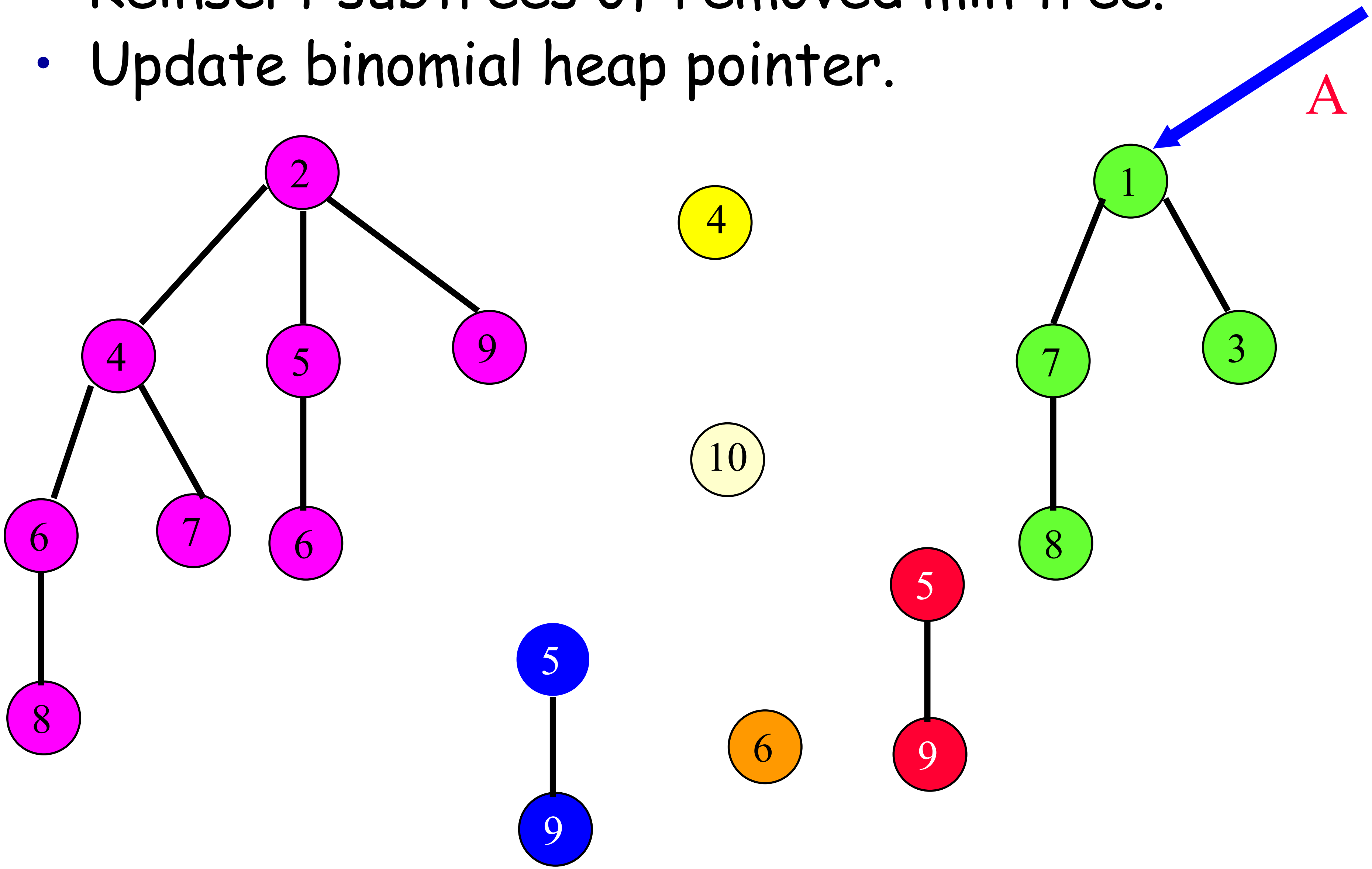


Delete Min

- Empty binomial heap => fail.

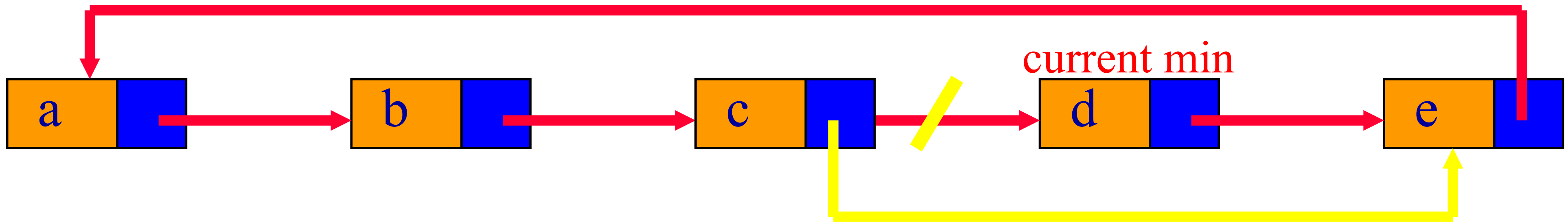
Nonempty Binomial Heap

- Remove a min tree.
- Reinsert subtrees of removed min tree.
- Update binomial heap pointer.



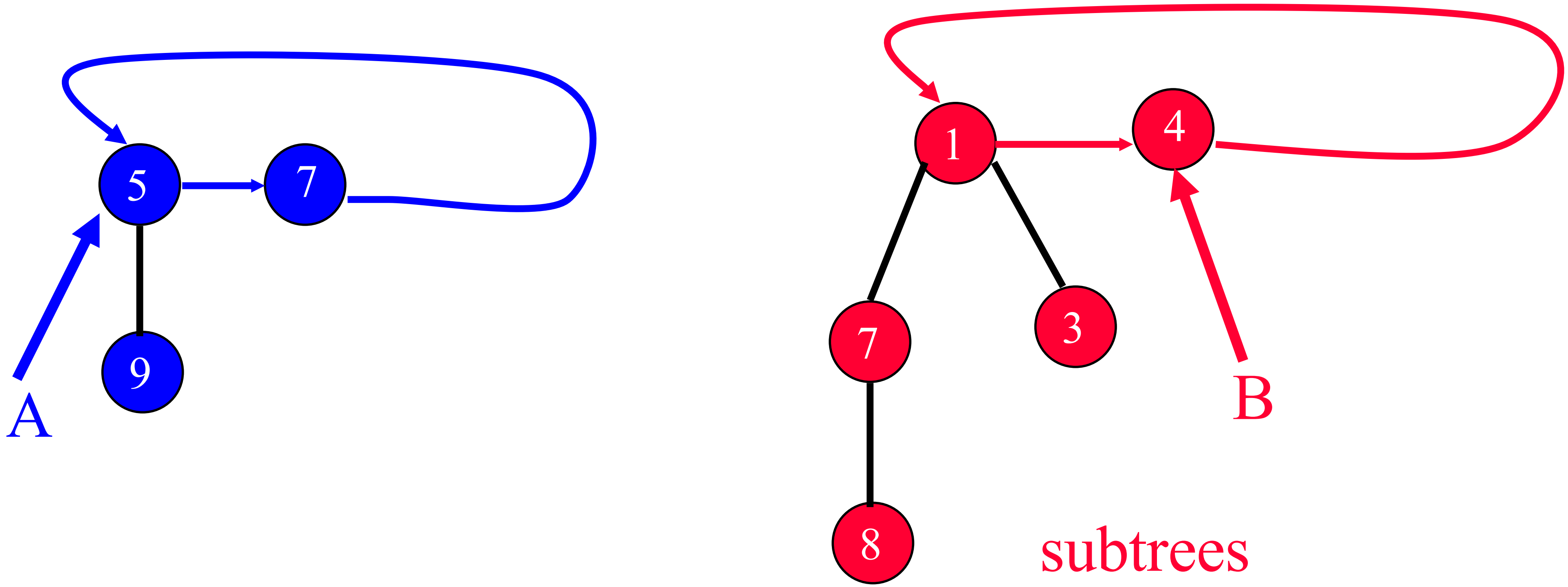
Remove Min

- Same as remove a node from a circular list.



- No next node => empty after remove.
- Otherwise, copy next-node data and remove next node.

Reinsert (Several) Several Subtrees, or Take Advantage of Merging Two Circular Lists



- Combine the 2 top-level circular lists.
  - Same as in meld operation.

Update Binomial Heap Pointer

- Must examine roots of all min trees to determine the min value.

Complexity of Delete Min

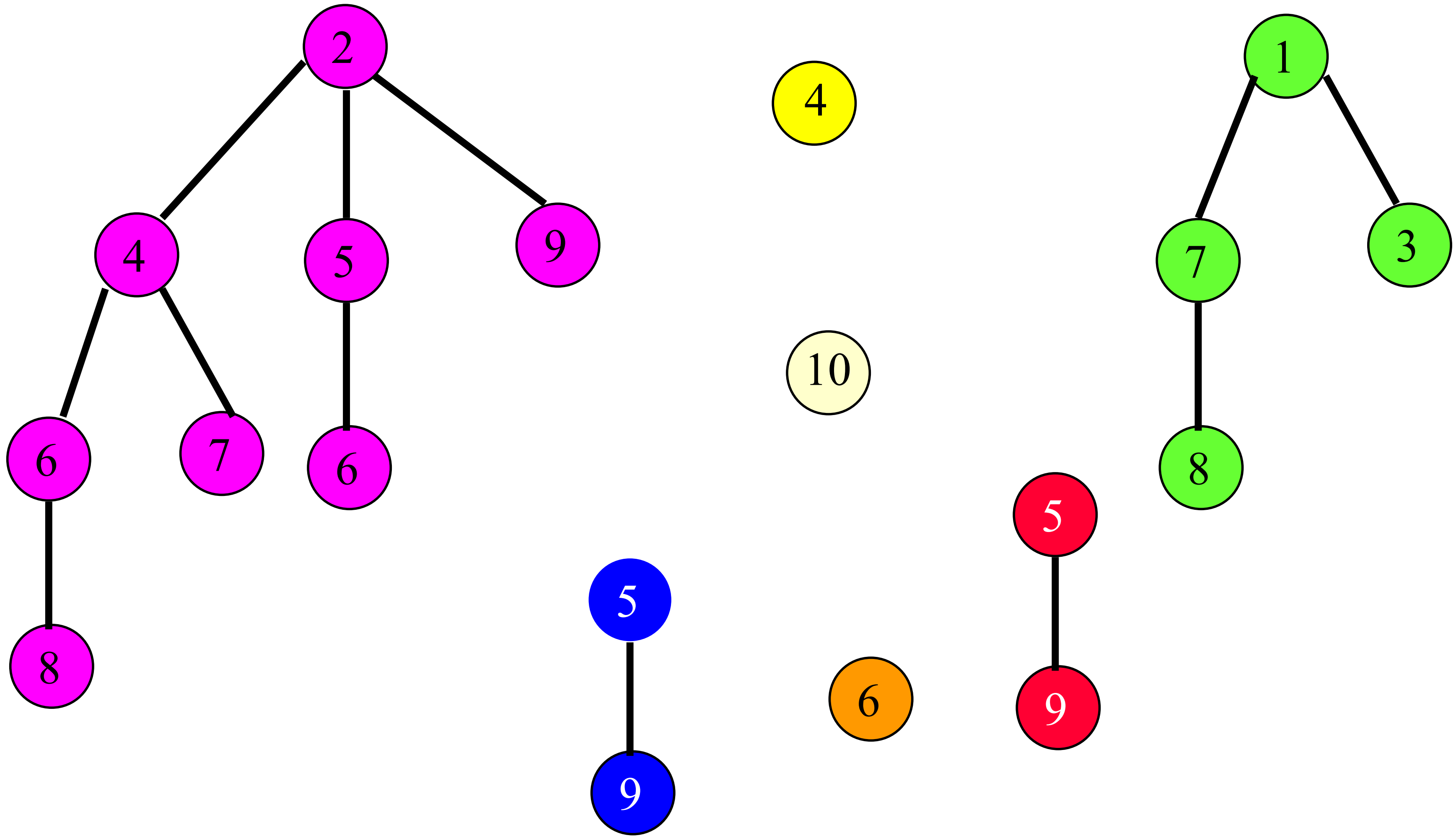
- Remove a min tree. //單純只考慮remove min
  - O(1).
- Reinsert subtrees. //兩個circular lists的join
  - O(1).
- Update binomial heap pointer. //remove min的代價
  - O(s), where s is the number of min trees in final top-level circular list.
  - s = O(n).
- Overall complexity of remove min is O(n).

Enhanced Delete Min

- During reinsert of subtrees, pairwise combine min trees whose roots have equal degree.

Pairwise Combine (Driven by Delete Min)

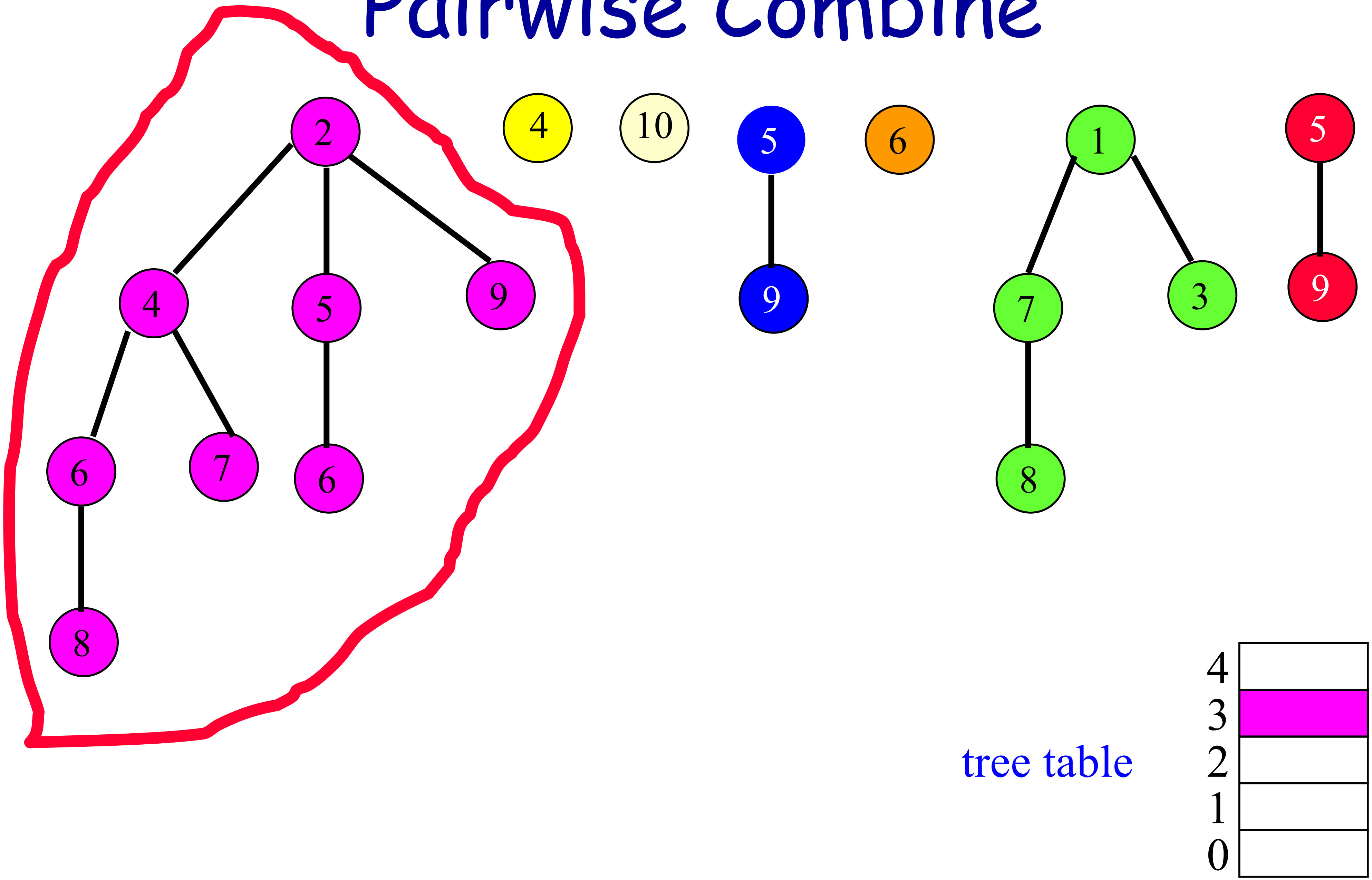
這裡假設有一個min被delete了，而餘如下min trees



Examine the s = 7 trees in some order. //比方照circular list的次序走訪

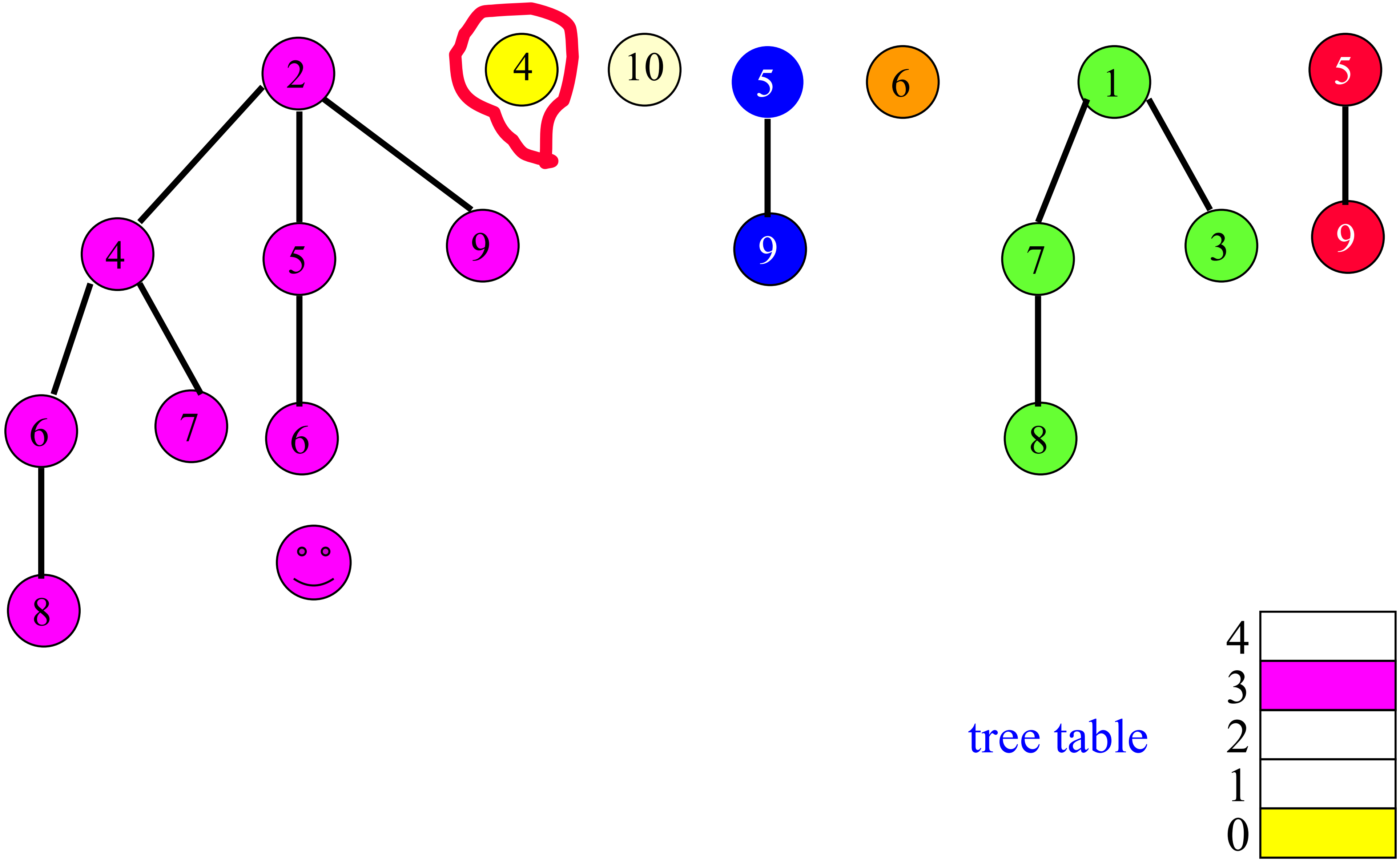
Determined by the 2 top-level circular lists.

Pairwise Combine

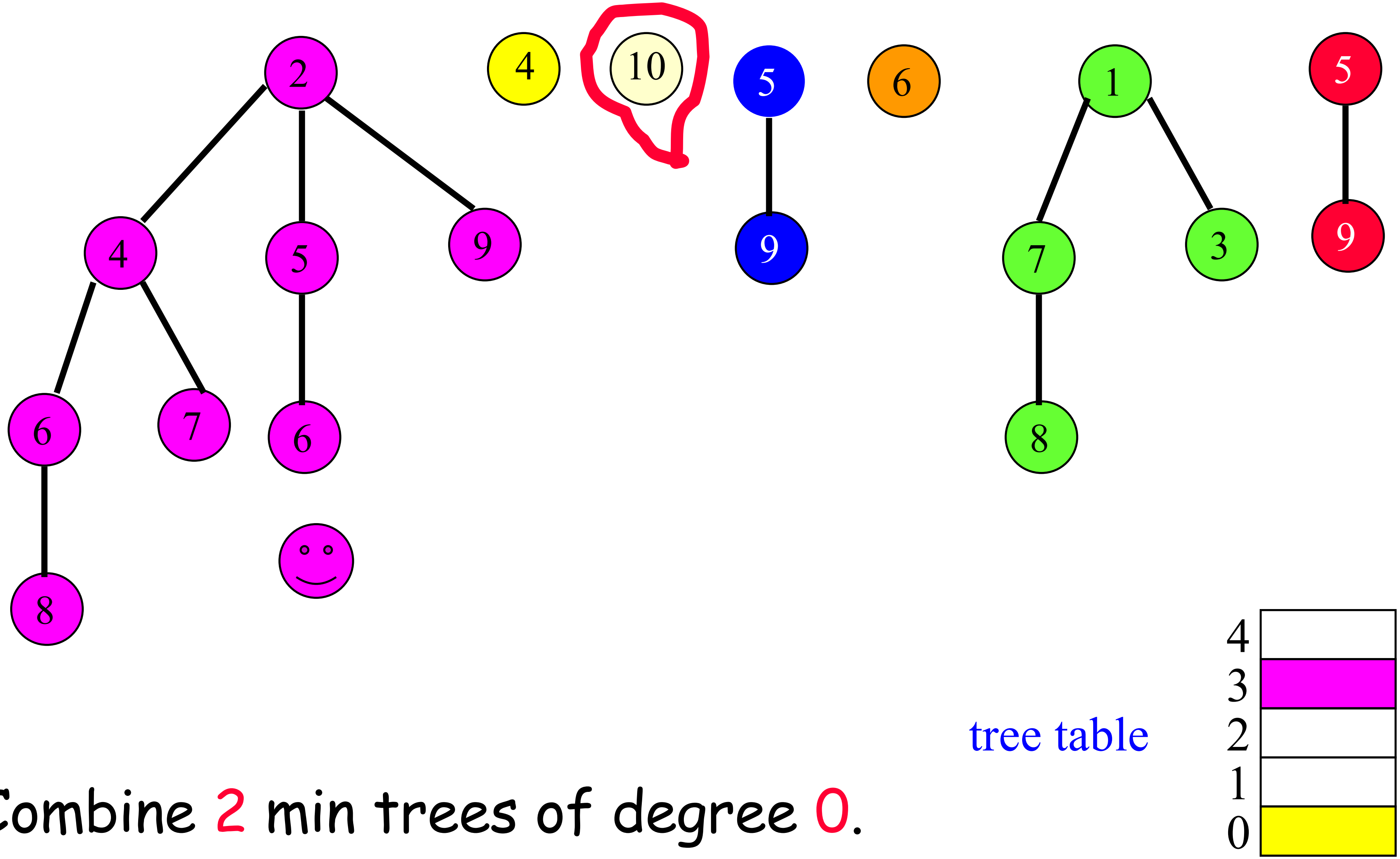


Use a table to keep track of trees by degree.

Pairwise Combine



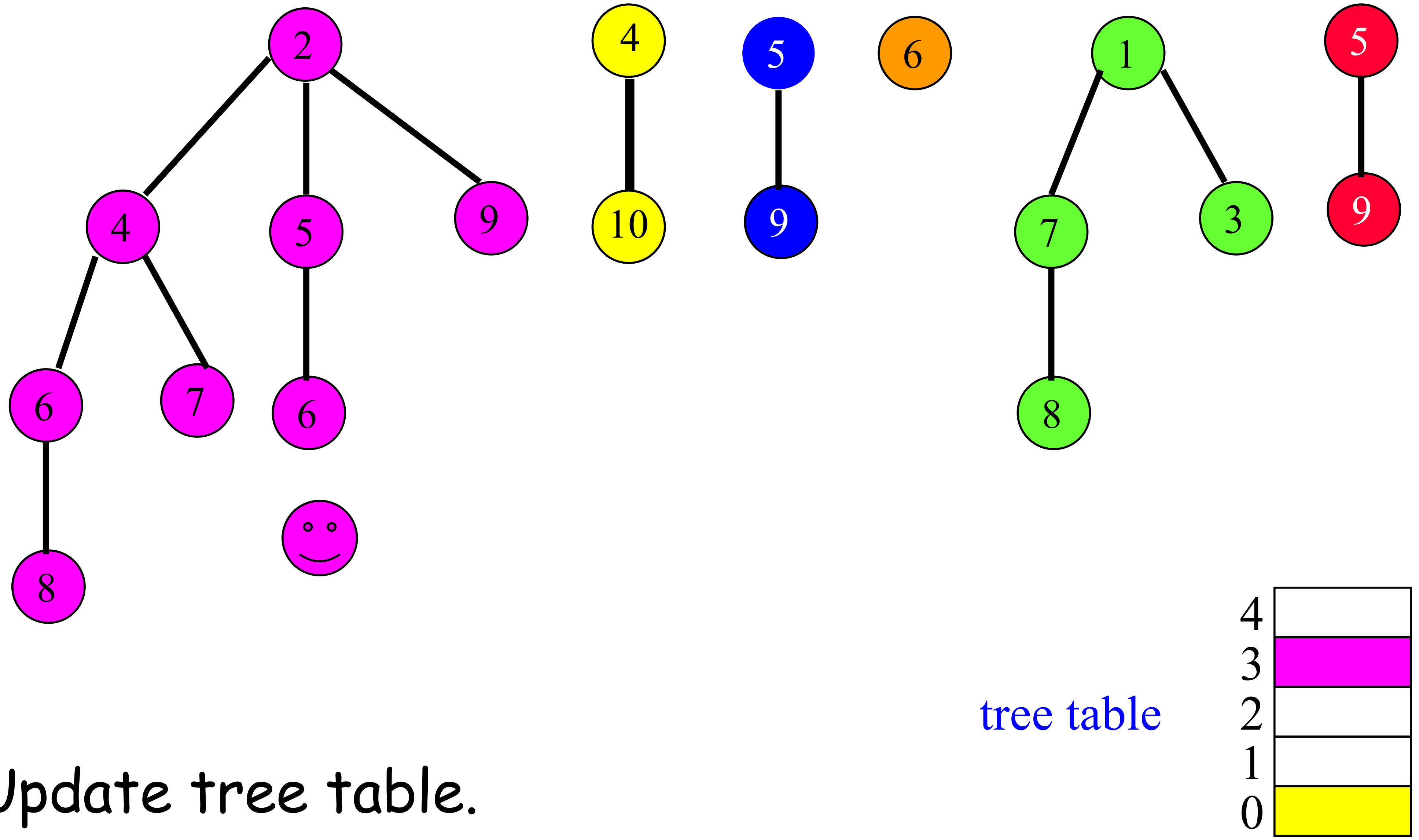
Pairwise Combine



Combine 2 min trees of degree 0.

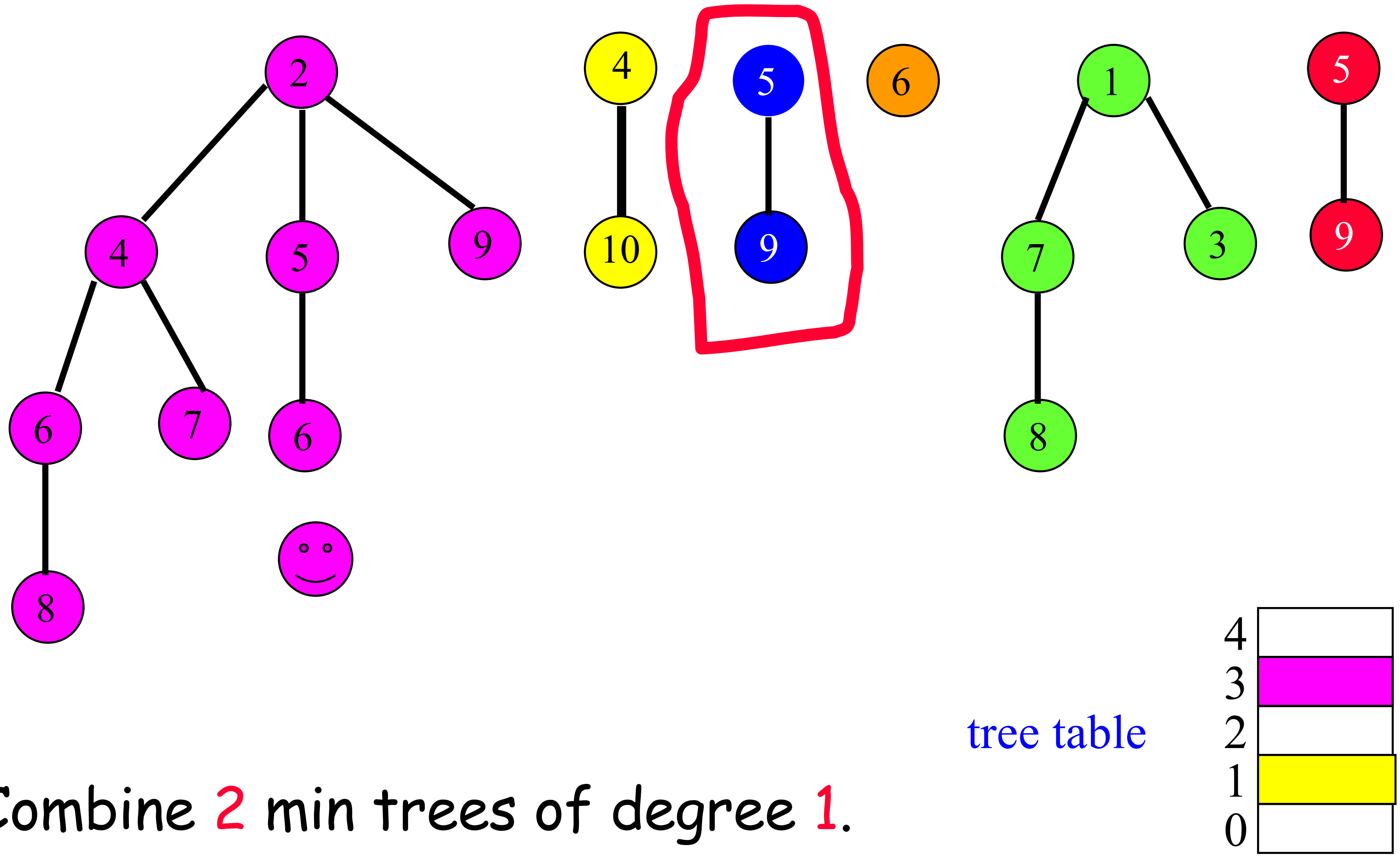
Make the one with larger root a subtree of other.

Pairwise Combine



Update tree table.

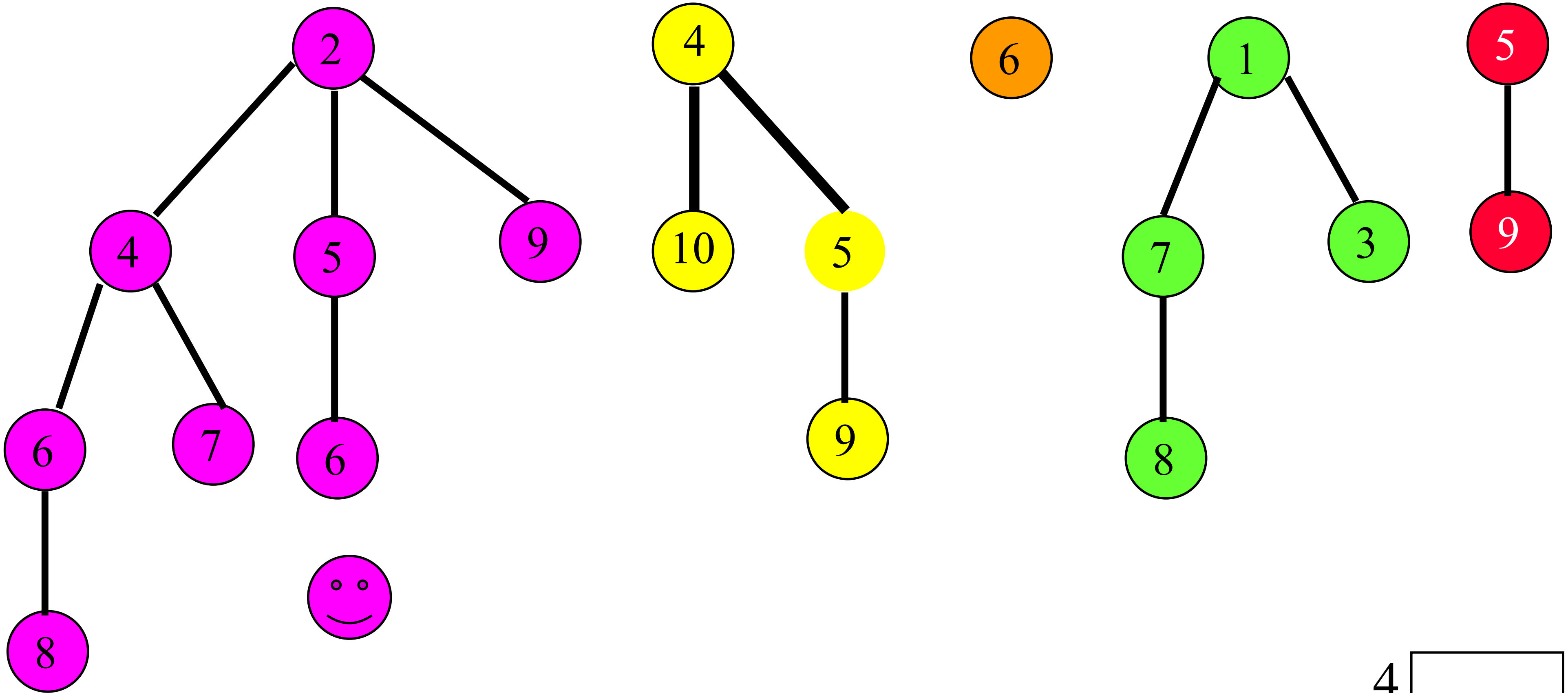
Pairwise Combine

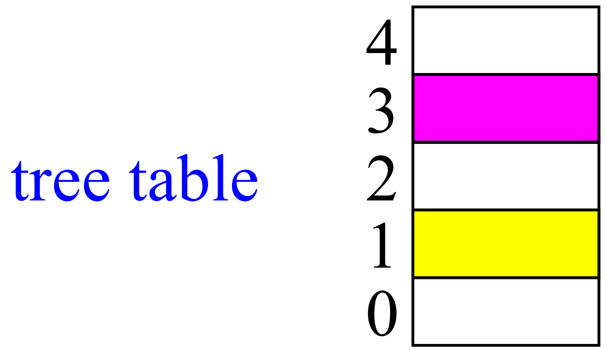


Combine 2 min trees of degree 1.

Make the one with larger root a subtree of other.

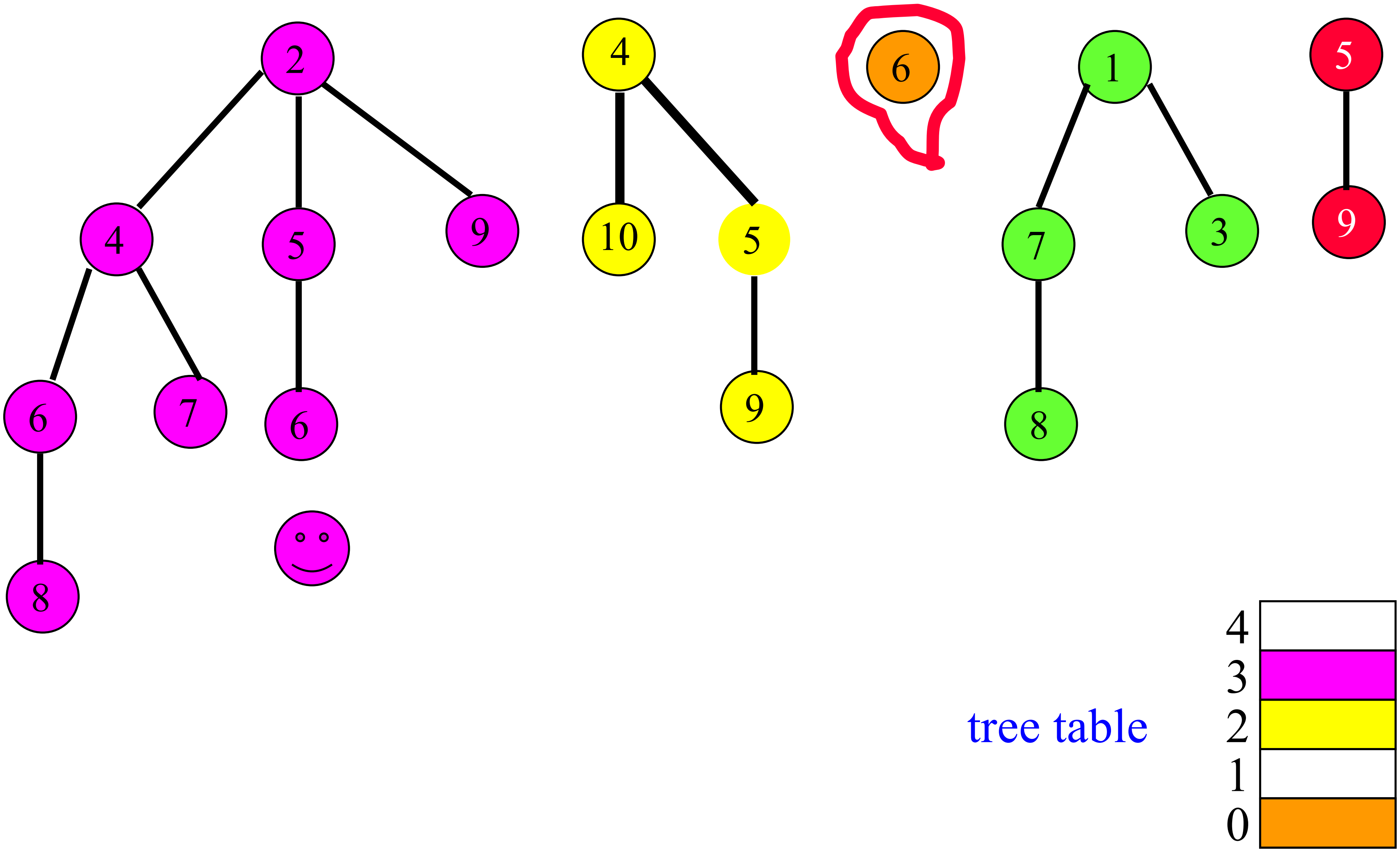
Pairwise Combine



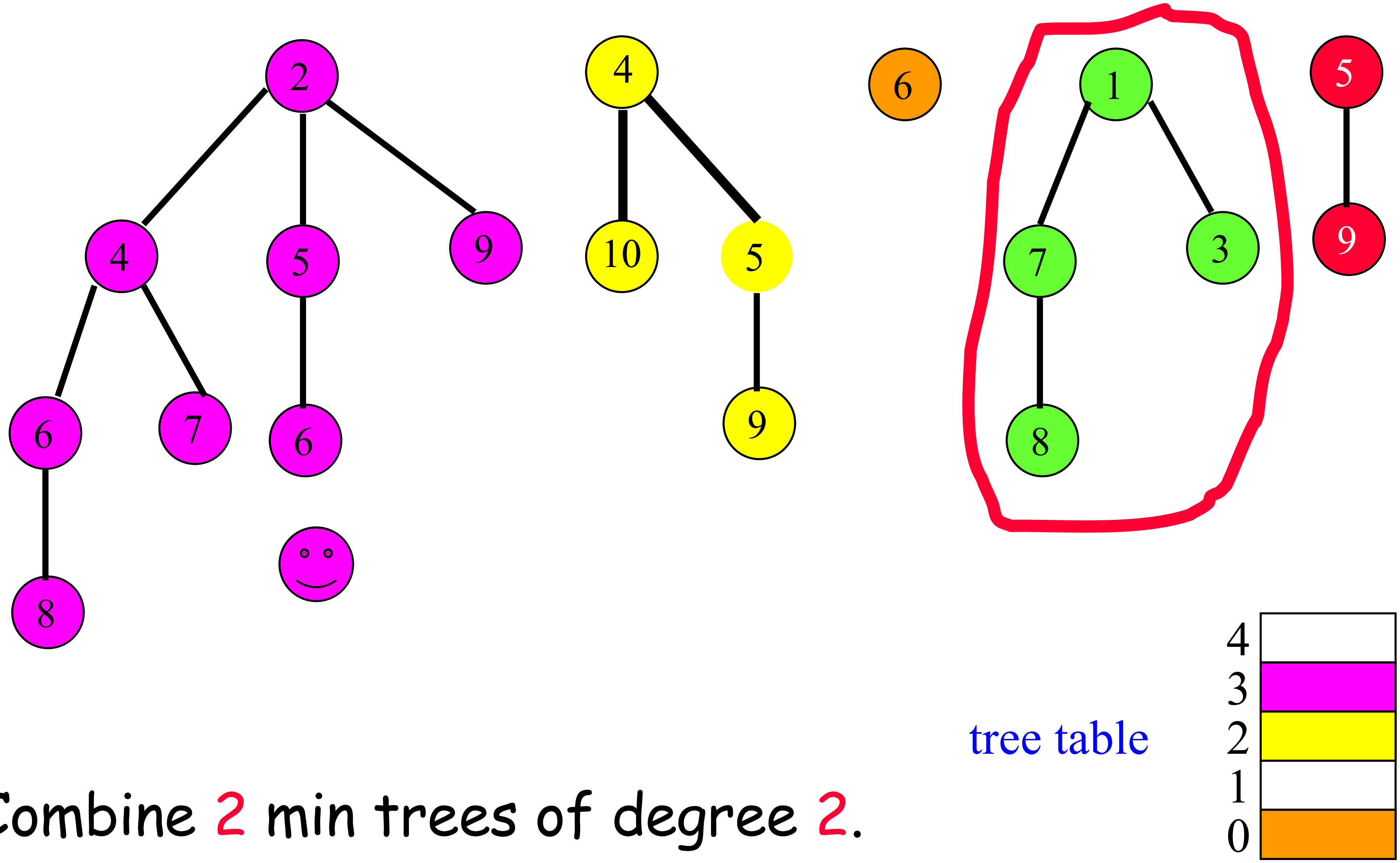


Update tree table.

Pairwise Combine



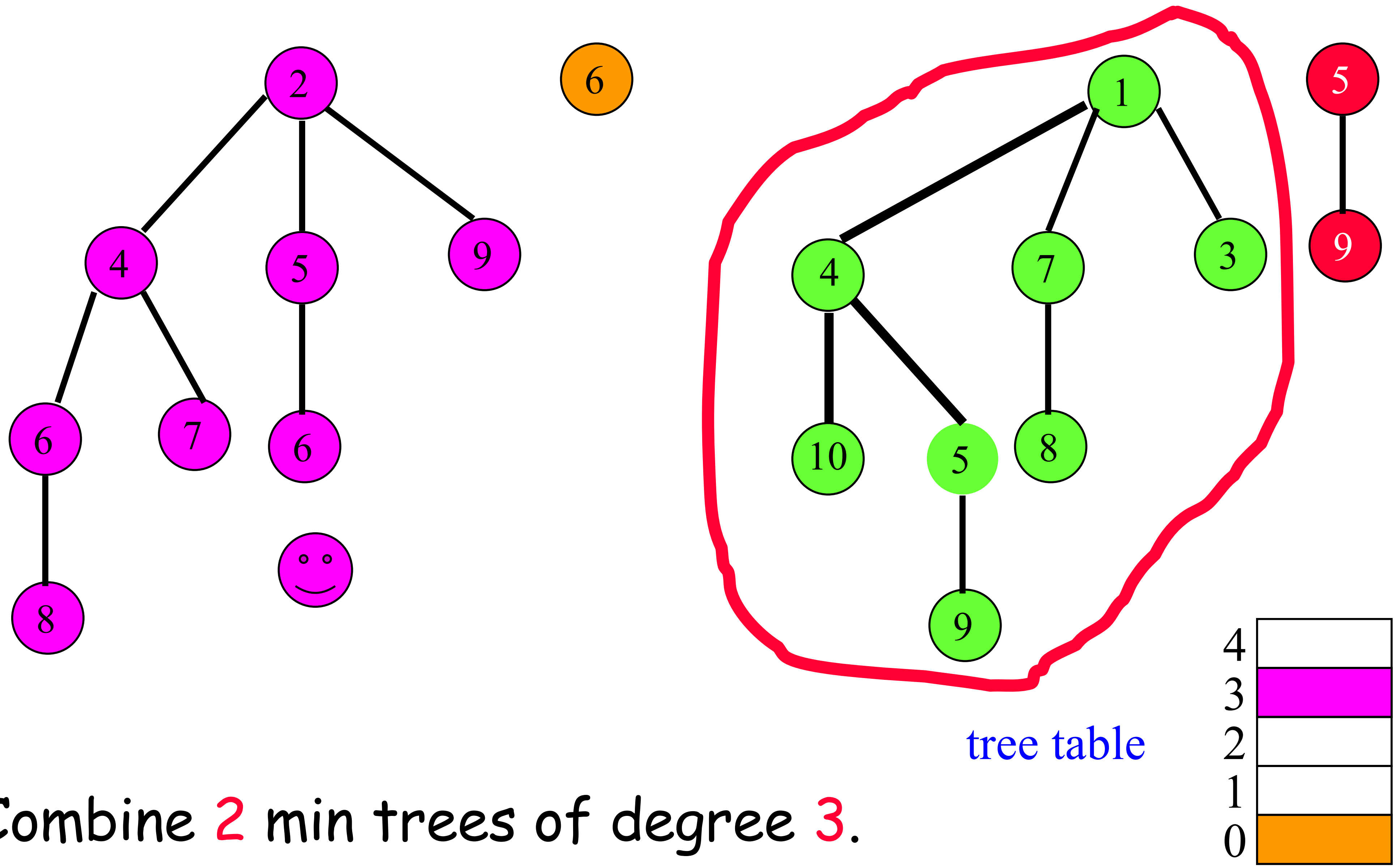
Pairwise Combine



Combine 2 min trees of degree 2.

Make the one with larger root a subtree of other.

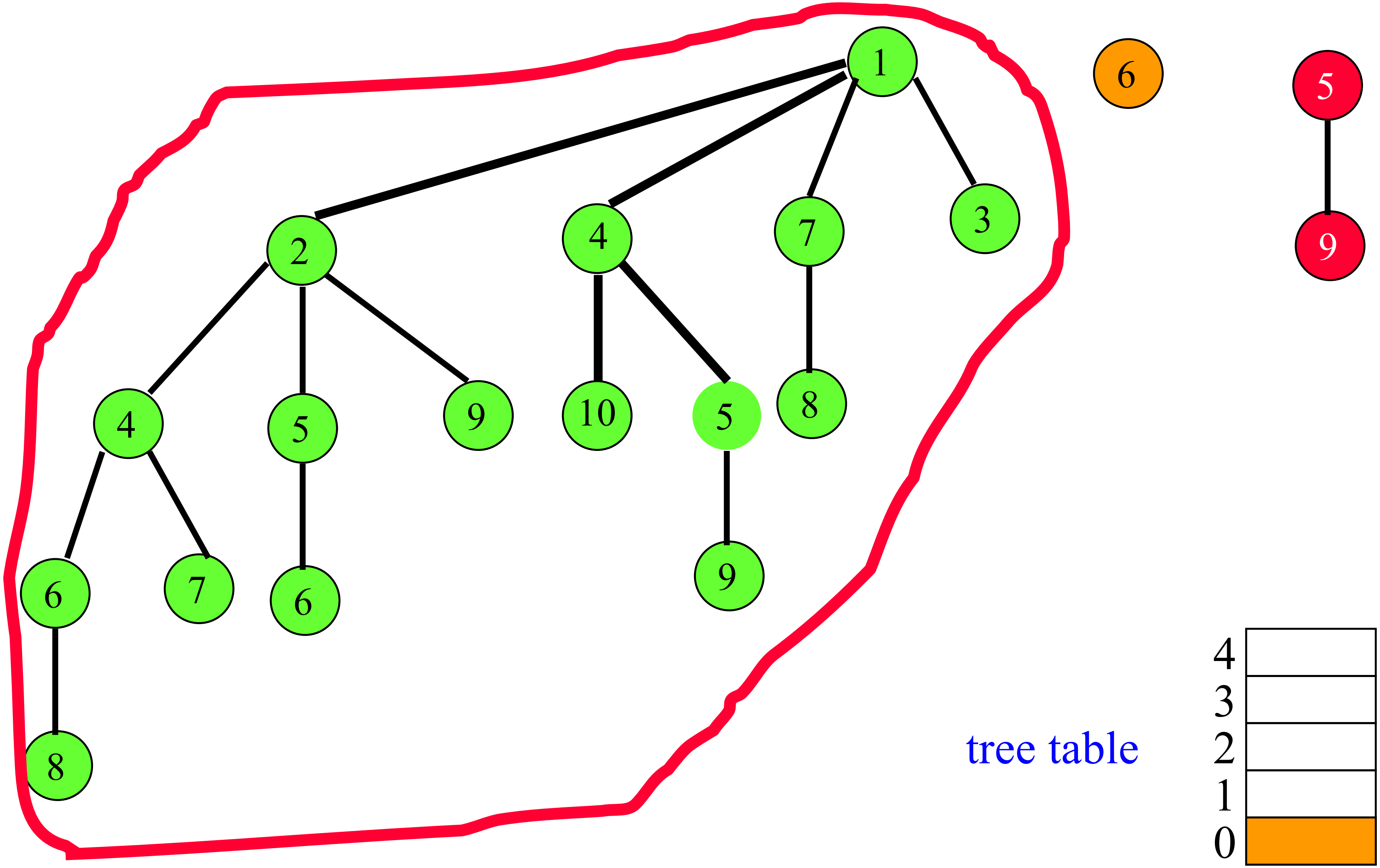
Pairwise Combine



Combine 2 min trees of degree 3.

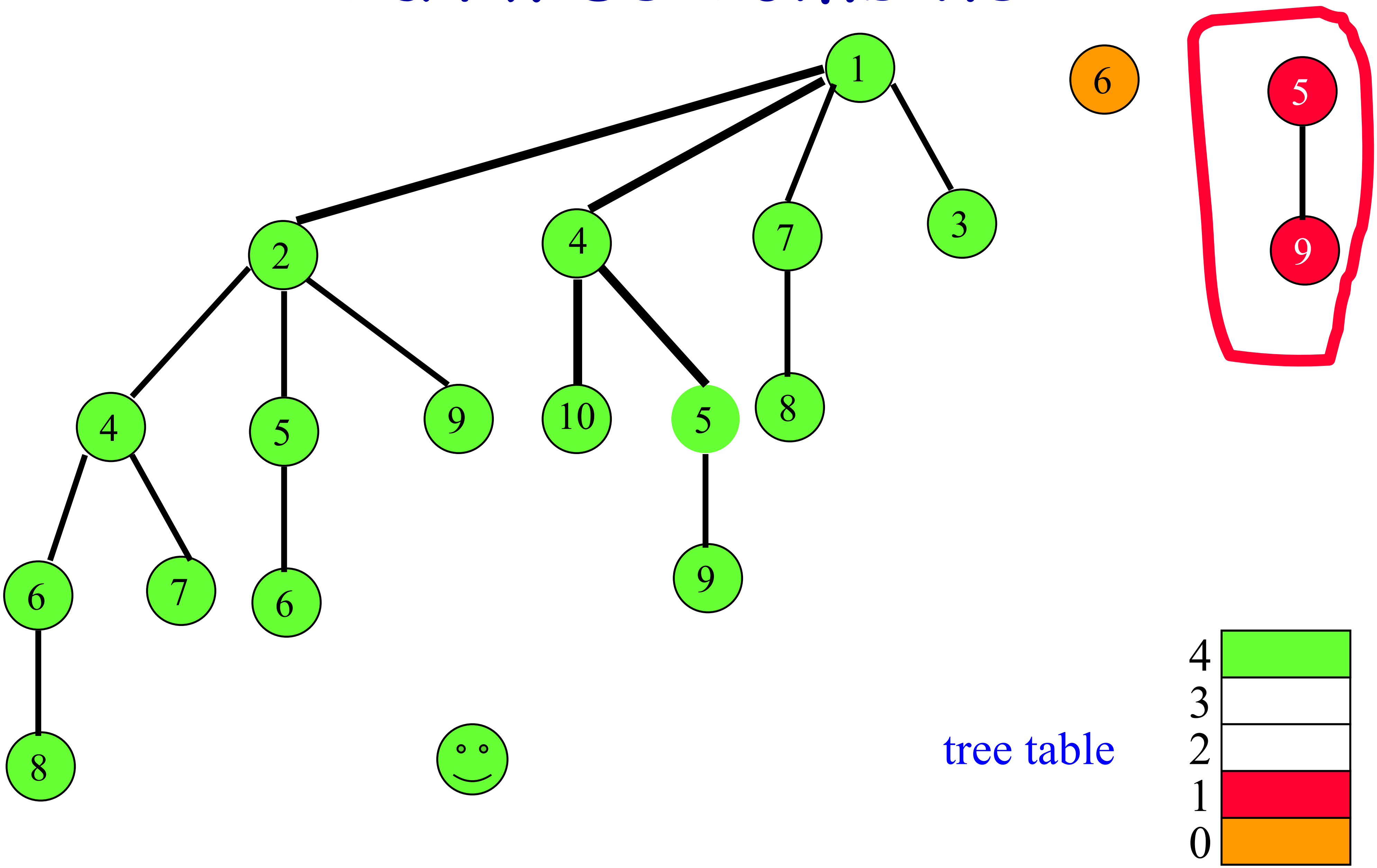
Make the one with larger root a subtree of other.

Pairwise Combine

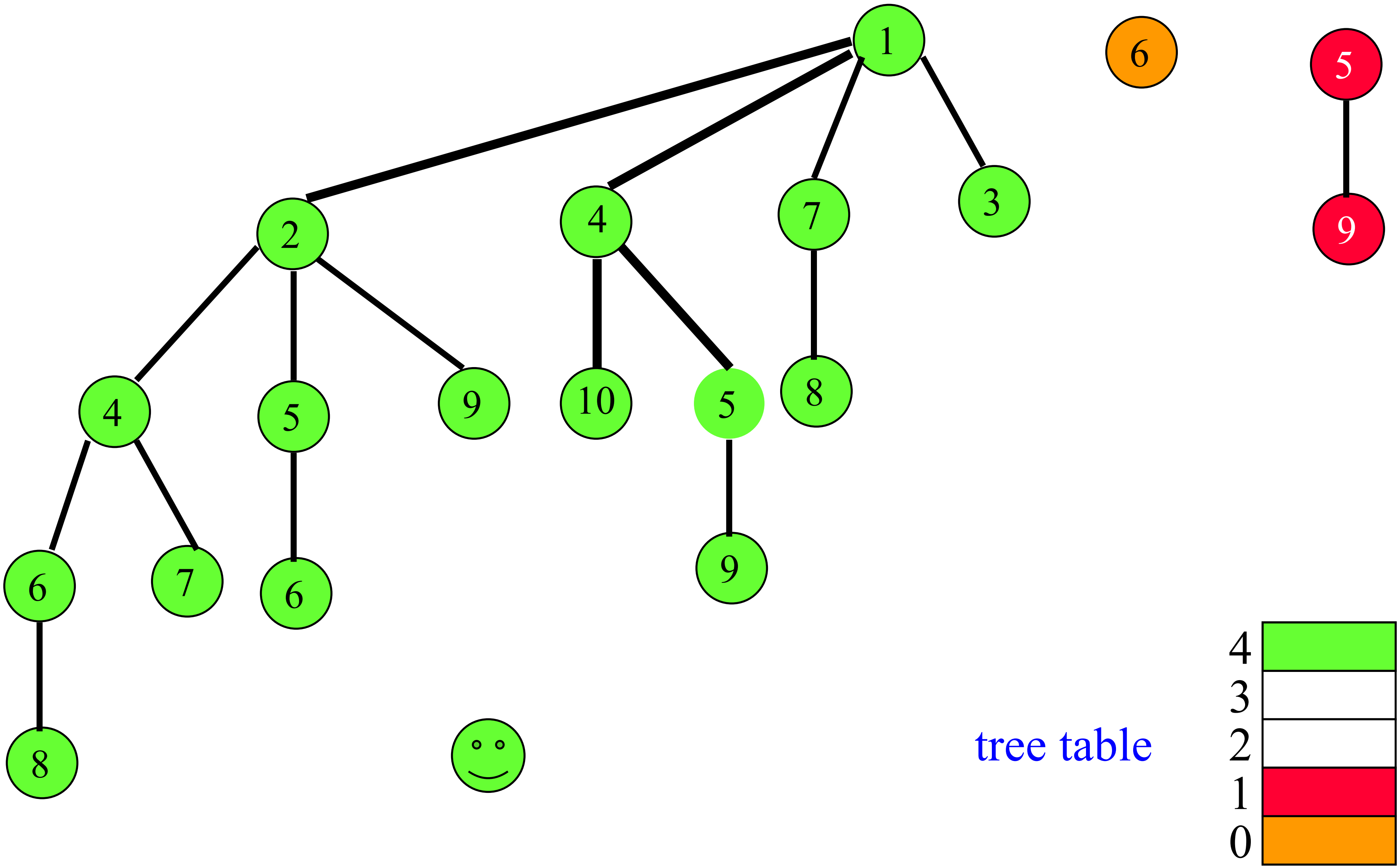


Update tree table.

Pairwise Combine



Pairwise Combine



Create circular list of remaining trees.

Complexity of Delete Min

- Create and initialize tree table.
  - O(MaxDegree).
  - Done once only.
- Examine s min trees and pairwise combine.
  - O(s). //每個s中的min binomial tree最多只會“被” meld一次
- Collect remaining trees from tree table, reset table entries to null, and set binomial heap pointer. //收尾
  - O(MaxDegree).
- Overall complexity of remove min.
  - O(MaxDegree + s).

N_k and MaxDegree

- $N_0 = 1$
- $N_k = 2N_{k-1}$
  $= 2^k$.
- If we start with "independently single elements" (一堆B_0) and perform operations as described, then all trees in all binomial heaps are binomial trees.
- So, MaxDegree = $O(\log n)$.

Performance Analysis

(說直白了：當時insert的時候沒有做任何的最佳化，“amortized cost”因此主張insert要來分攤delete min的成本)

	Binomial heaps	
	Actual	Amortized
Insert	O(1)	O(1)
Delete min (or max)	O(n)	O(log n)
Meld	O(1)	O(1)

Amortized cost:
1)當有 “整批” (batched) 的操作 (有連續一堆單筆資料的inserts) 時才會有意義
2)“Enhanced” delete min的操作成本 O(MaxDegree + s)
3)將s攤銷到之前的每個insert操作 O(1+1)=O(1)，該些insert's為連續兩次delete min當中的那些insert's
4)Amortized “enhanced delete min” 為 O(MaxDegree) = O(log n)