

Binomial Heaps

Ch. 9.3

Leftist vs. Binomial heaps

- Time complexity for different operations

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

Toy example of amortization

- Performing a sequence of insert and delete-min operations:

I1, I2, D1, I3, I4, I5, I6, D2, I7

Actual cost of each insertion: 1

Actual cost of D1: 8

Actual cost of D2: 10

Total cost: 25

- Cost transferring (amortization) scheme:

Transfer one unit of cost of a delete-min to the prior insertions.

	I1	I2	D1	I3	I4	I5	I6	D2	I7
Actual	1	1	8	1	1	1	1	10	1
Amortized	1 +1	1 +1	8 -2	1 +1	1 +1	1 +1	1 +1	10 -4	1

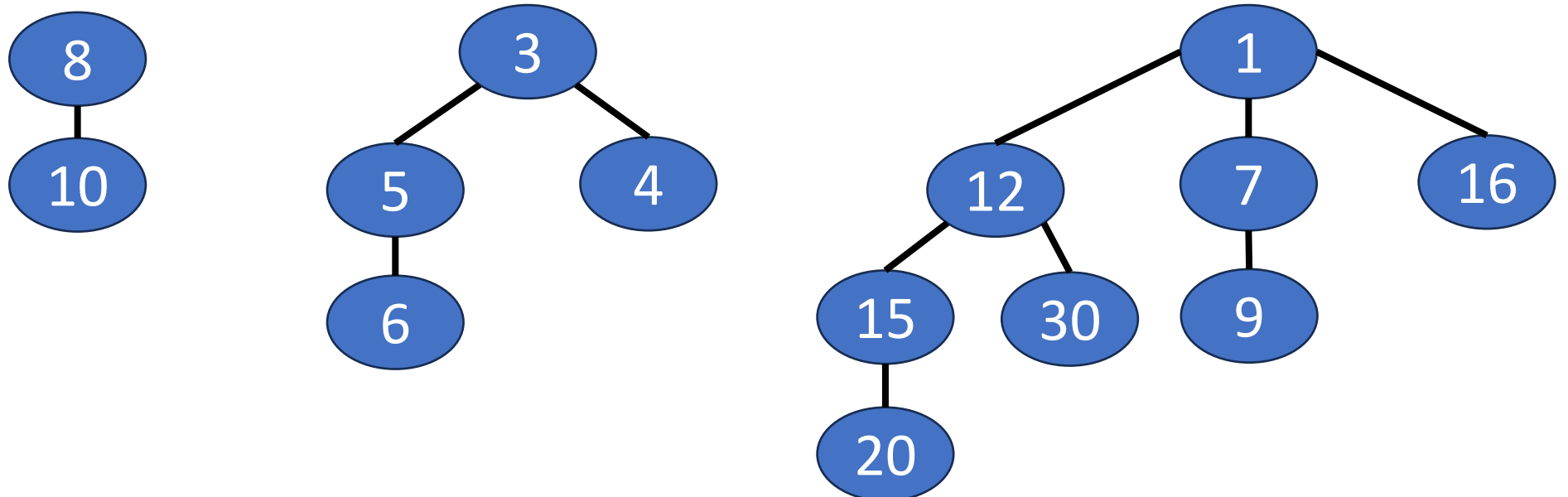
Amortization

- Sometimes, we care more about the overall time to perform a sequence of operations.
 - The time to perform a single operation is less important.
 - For example: sorting.

Binomial heaps (B-heap)

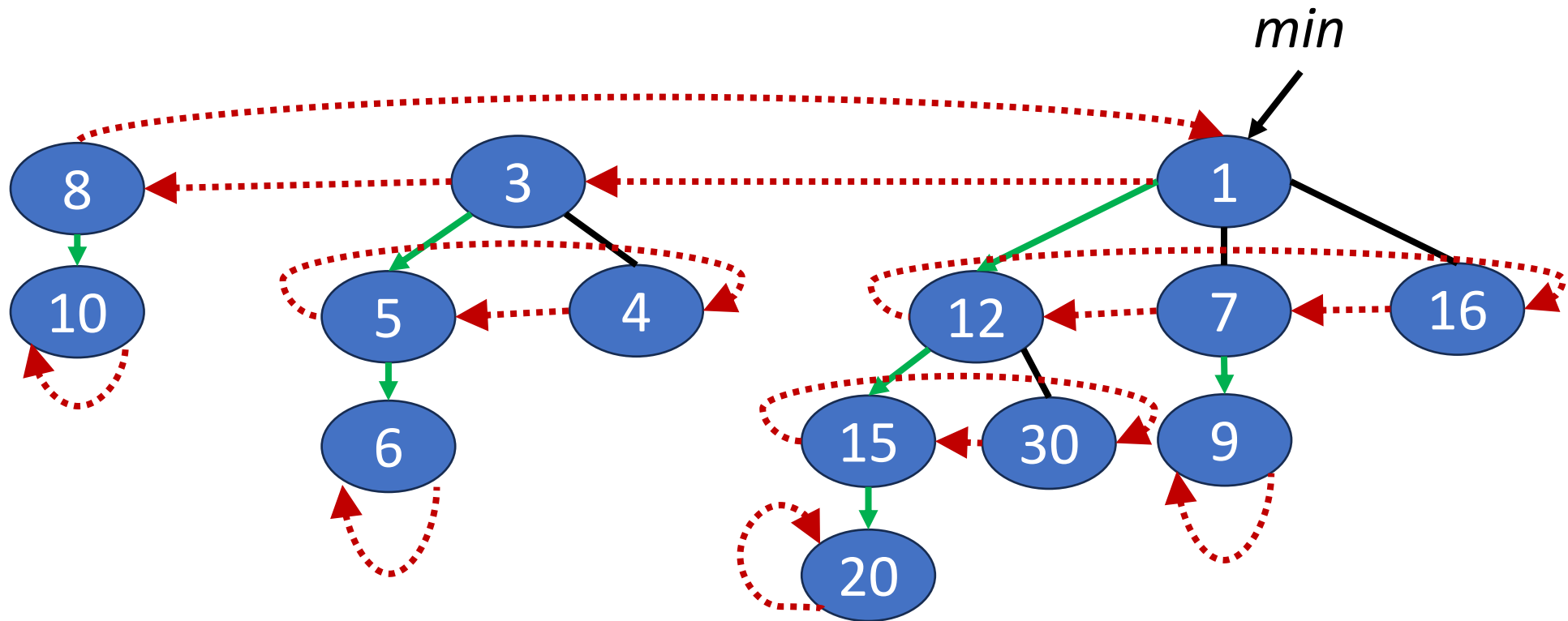
- **Max** binomial heap
 - A collection of **max** trees
 - **Min** binomial heap
 - A collection of **min** trees
- (In this lecture, we will consider **min** binomial heaps only.)

Example: A B-heap with 3 min trees.



Binomial heap representation

- All roots of min trees form a **circular linked list**.
 - All the children of a node form a **circular linked list**.
- {10}, {6}, {5,4}, {20}, {15,30}, {9}, {12,7,16}, {8,3,1}
- *min*: pointer to the B-heap.



Node structure

- Child

- Pointer to one of the node's children.
- Null iff node has no child.

- Link

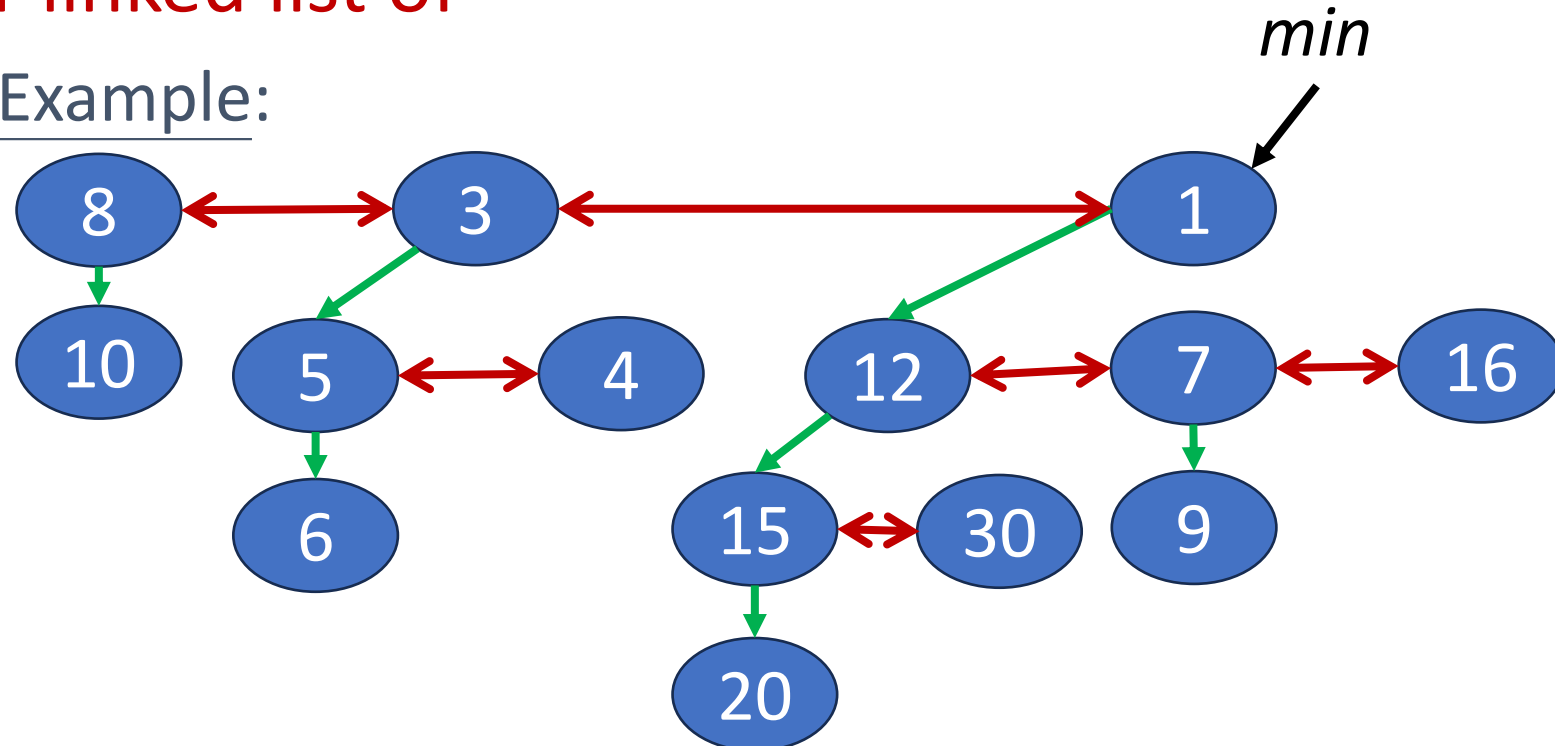
- Used for singly circular linked list of siblings.

- Degree

- Number of children.

- Data

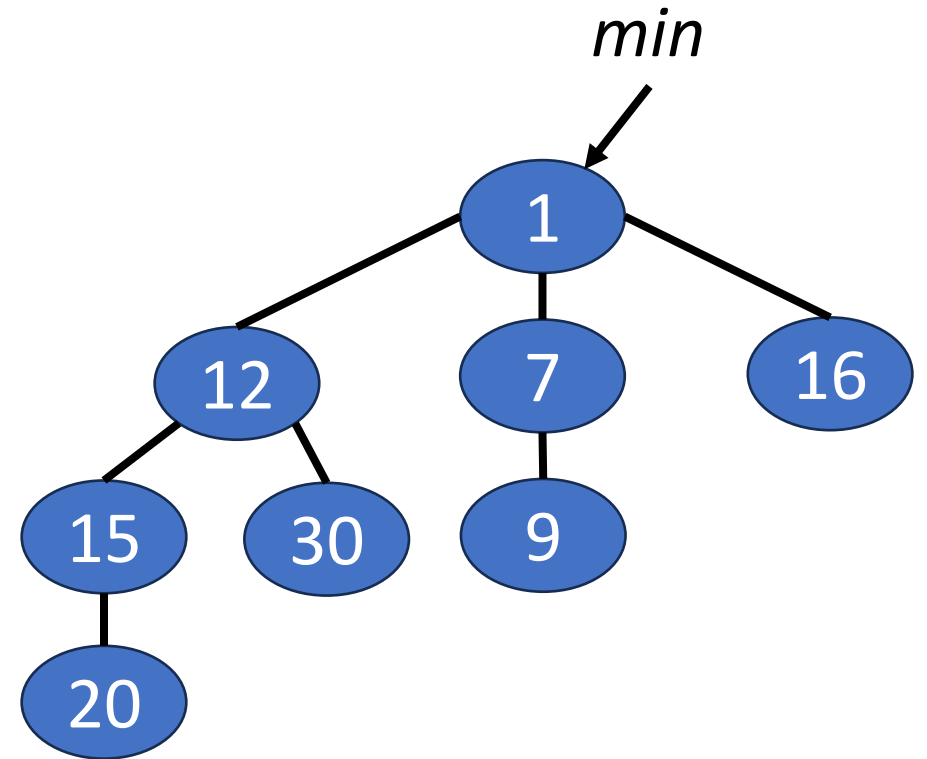
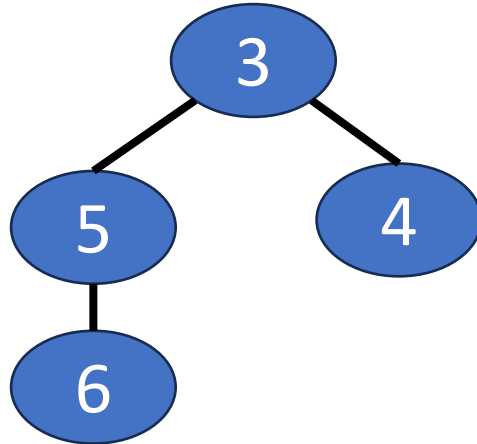
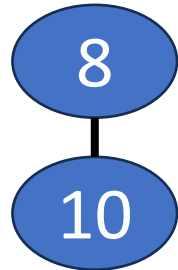
Example:



Operation: Insertion

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

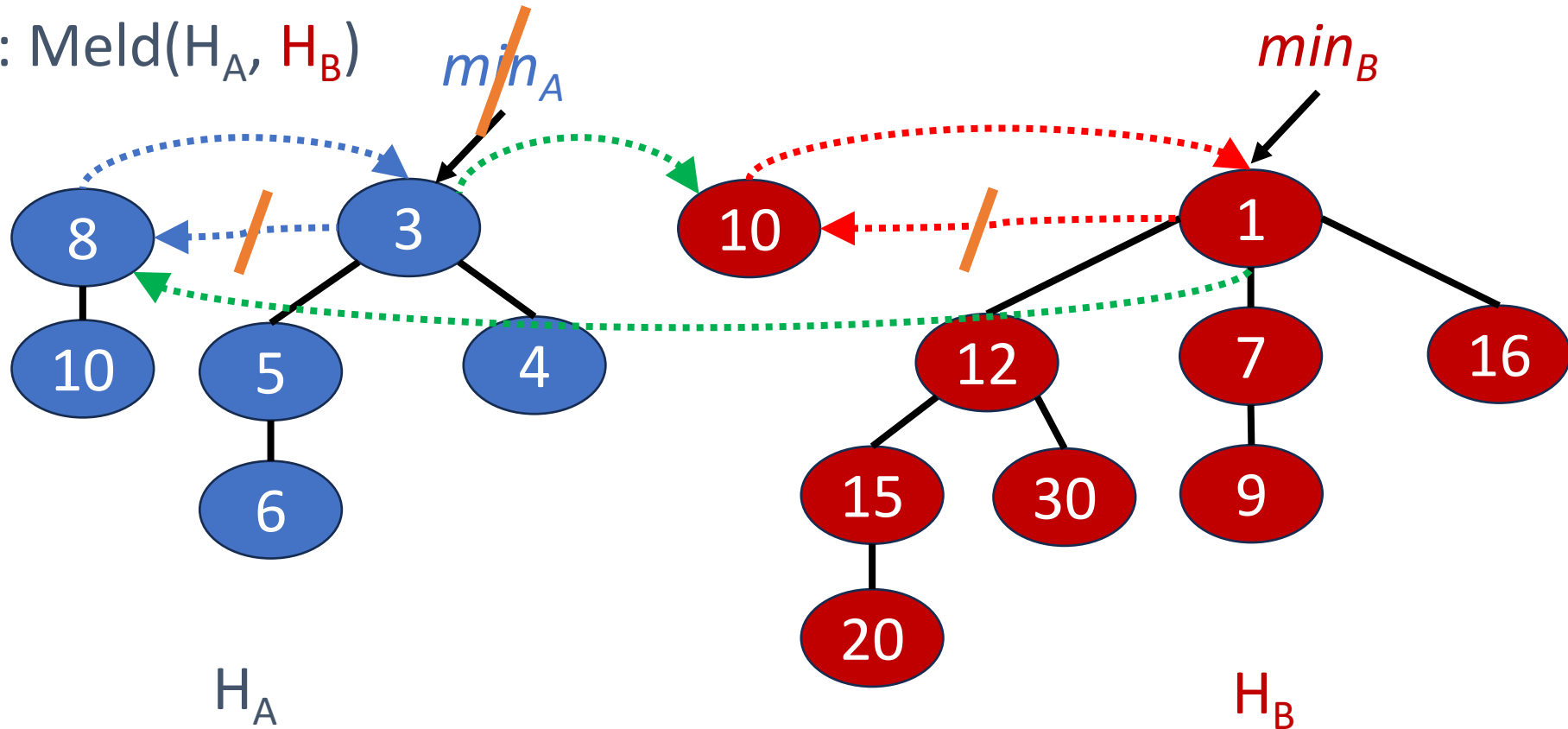
Example: Insert 10



Operation: Meld

- Meld the top circular lists of two B-heaps into one circular list
- Keep only the min pointer having the smaller key

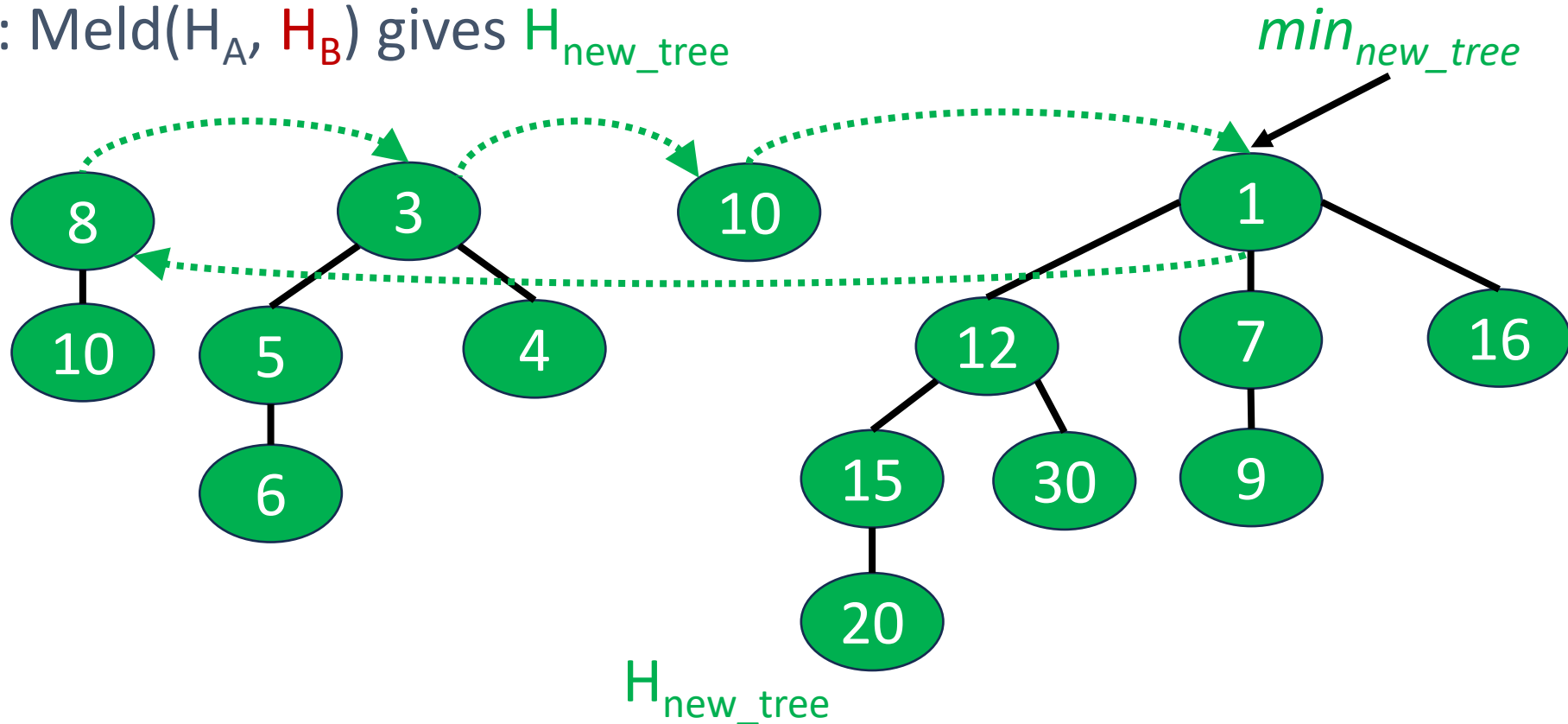
Example: $\text{Meld}(H_A, H_B)$



Operation: Meld

- Meld the top circular lists of two B-heaps into one circular list
- Keep only the min pointer having the smaller key

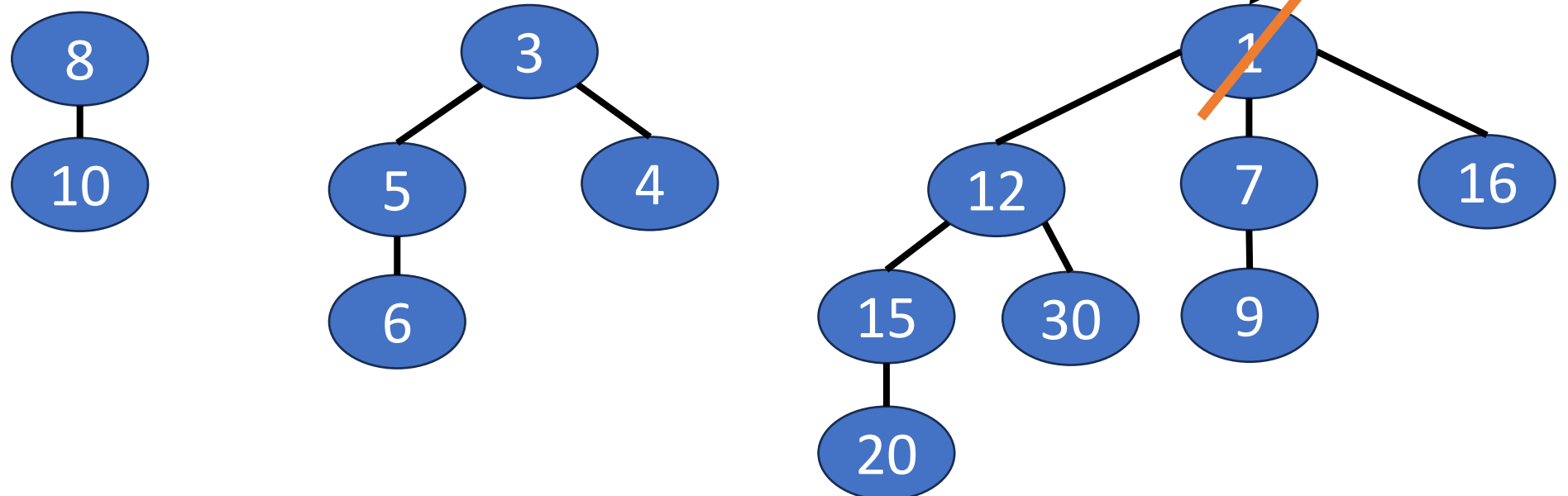
Example: Meld(H_A , H_B) gives $H_{\text{new_tree}}$



Operation: Delete-min

- If min is 0, the B-heap is empty. → Cannot perform deletion.
- If min points to a node, the B-heap is not empty.
 - 1) Remove the node pointed by min.
 - 2) Reinsert subtrees of the removed node.
 - 3) Update the pointer min.

Example:

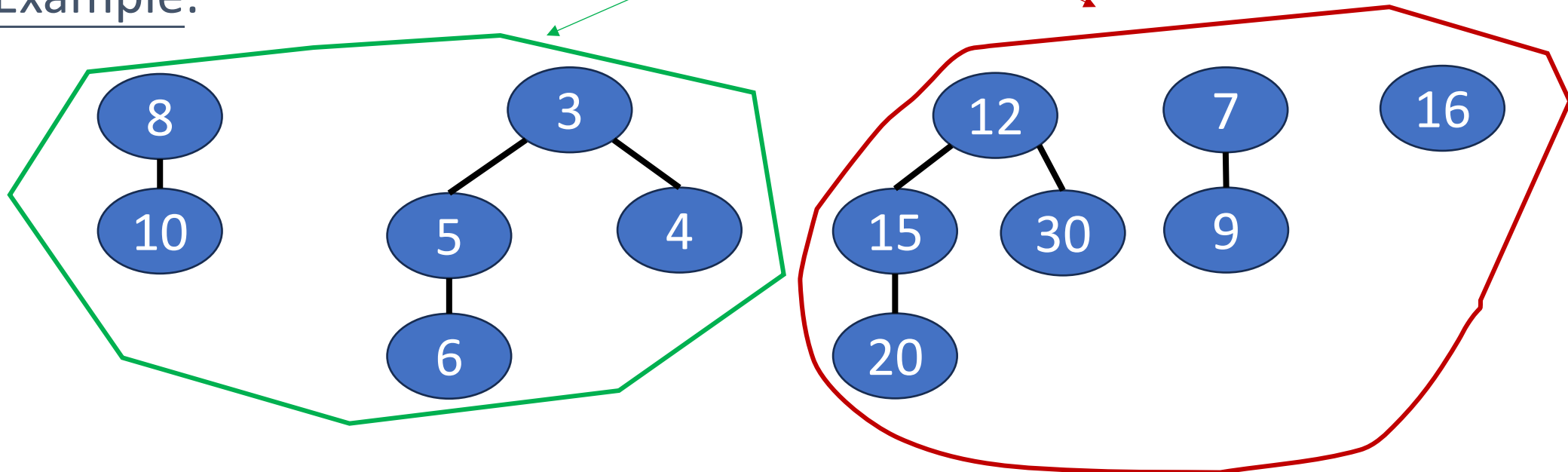


Operation: Delete-min

1) Remove the node pointed by min.

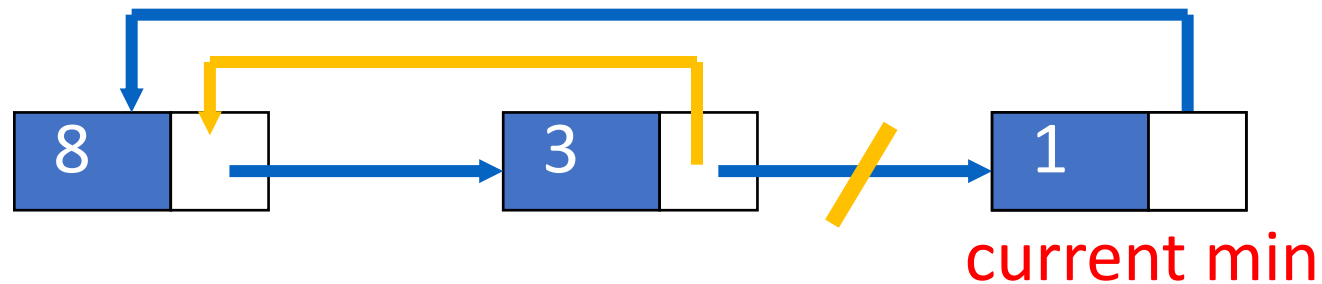
➔ New B-heap consists of the **remaining min trees** and the **sub-min tree of the deleted node**.

Example:



Remove min node from B-heap

- Delete the min node from its circular list.
- Same as removing a node from a circular list.



- If no next node → The list is empty after remove min node.

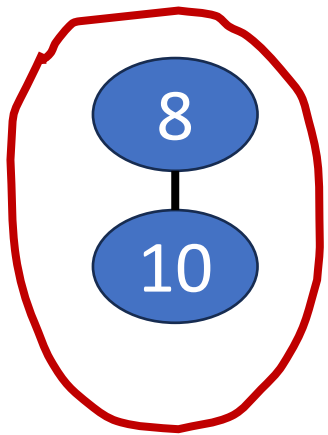
Operation: Delete-min

2) Min-tree joining.

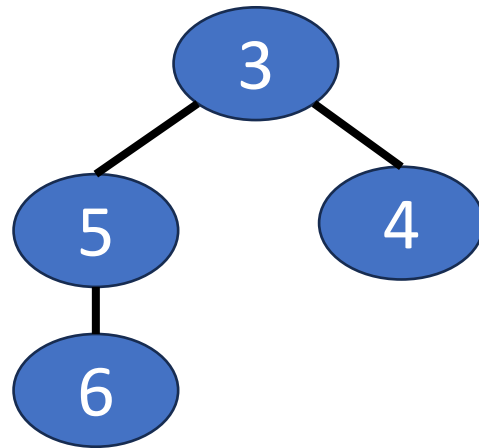
- Pairs of min trees having the same degree must be joined.

The degree of a min tree is the degree of its root.

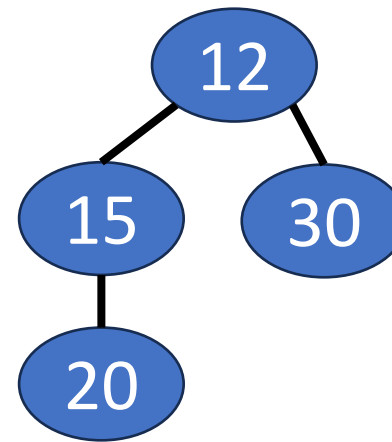
Example:



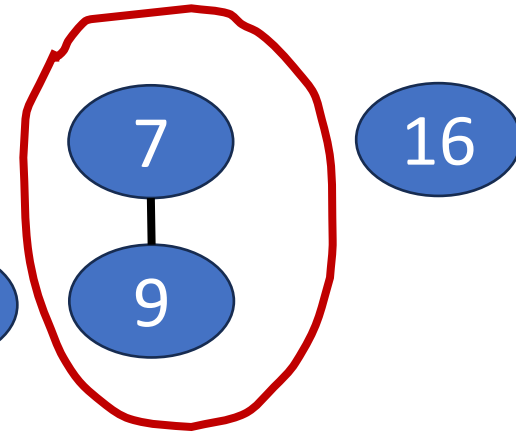
Degree = 1



Degree = 2



Degree = 2



Degree = 1 Degree = 0

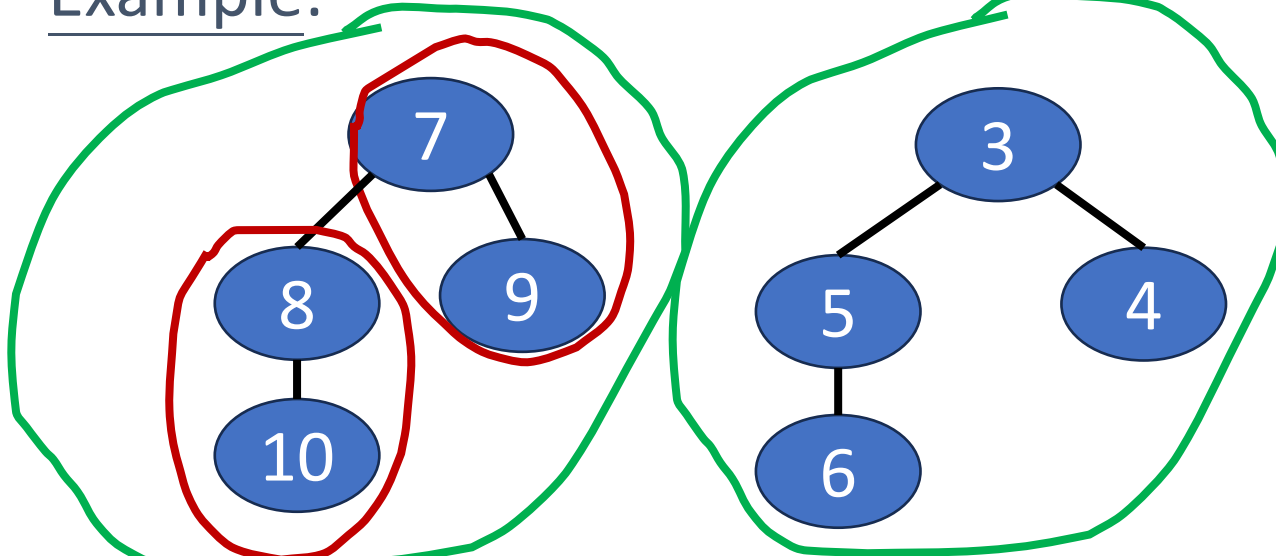
Operation: Delete-min

2) Min-tree joining.

- Pairs of min trees having the same degree must be joined.
- The min tree(root has a larger key) becomes a subtree of the min tree(root has a smaller key) . **To maintain property of min tree.**

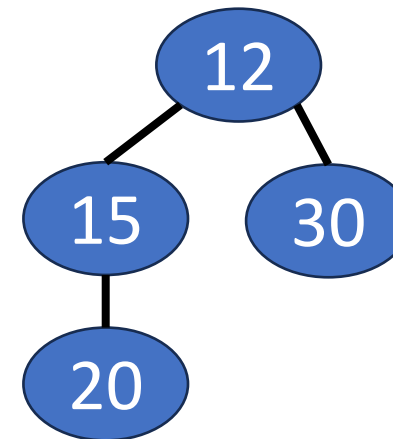
Example:

Find a pair of degree-two min trees



Degree = 2

Degree = 2



Degree = 2



Degree = 0

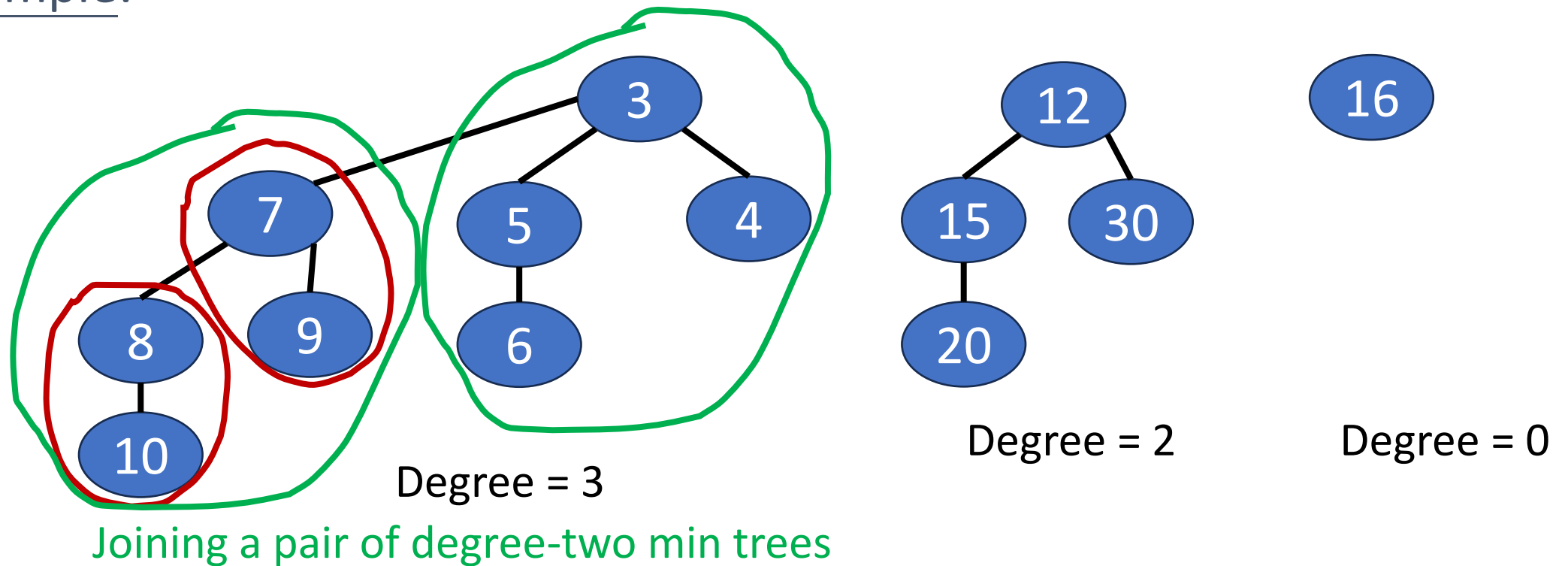
Joining a pair of
degree-one min trees

Operation: Delete-min

2) Min-tree joining.

- Pairs of min trees having the same degree must be joined.
- The min tree(root has a larger key) becomes a subtree of the min tree(root has a smaller key) .

Example:



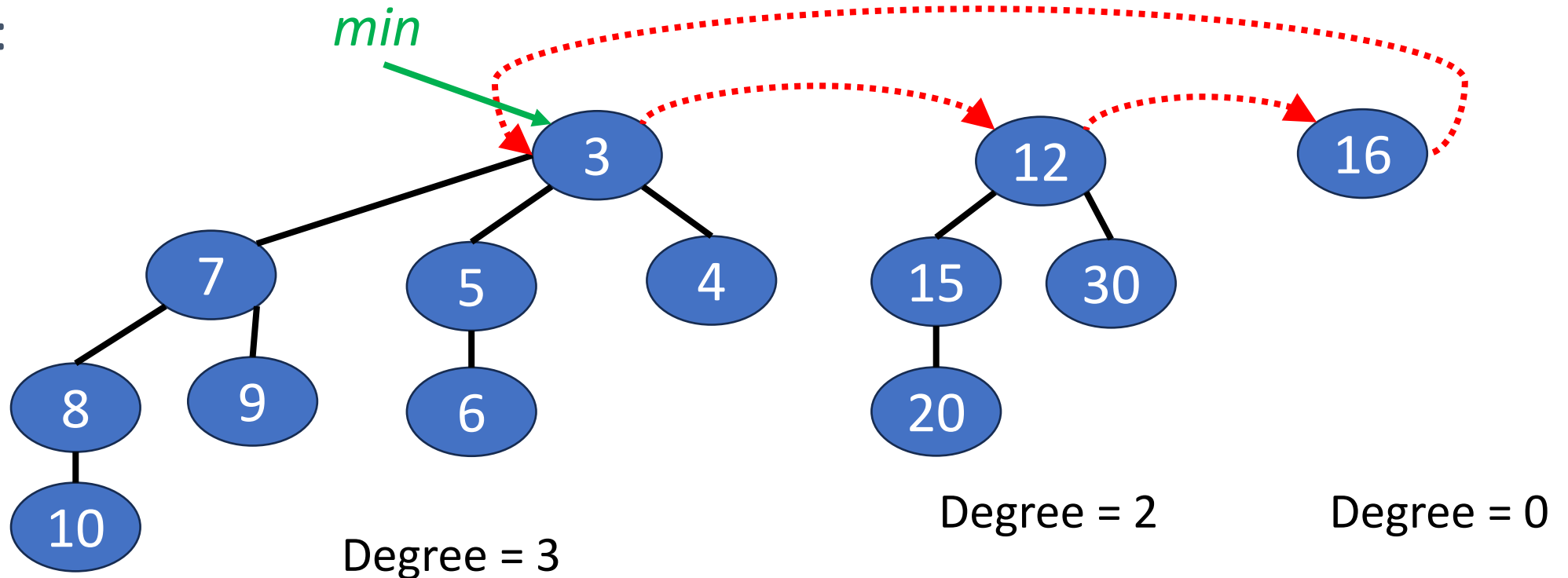
Operation: Delete-min

See programs 9.3 and 9.4 in textbook for more details.

3) Form **min-tree root list** and update **min pointer**.

- Link roots of the remaining min-trees together using linked circular list.
- Set min to point to the root with minimum key.

Example:



Importance of circular lists in B-heaps

- Reinsert sub-trees due to removal of min
 - Deleting *min* node results in several sub-min trees.
 - With the circular list, those sub-min trees can be visited.
- Find *min* node
 - Searching the top-level circular list.
- Meld two binomial heaps
 - Melding the top-level circular lists of the two B-heaps.

Complexity of Delete-min

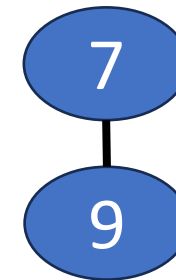
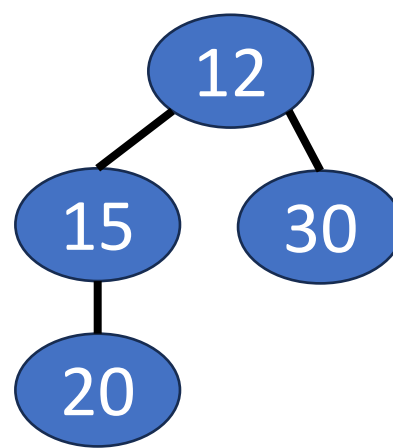
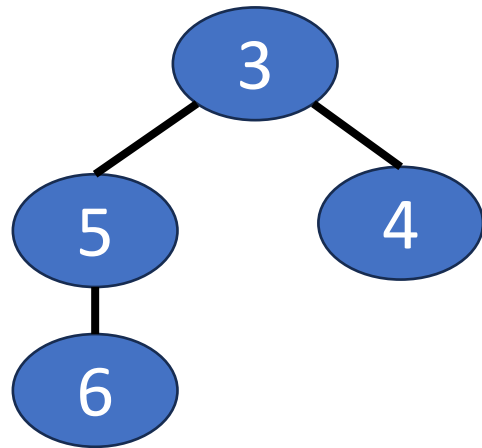
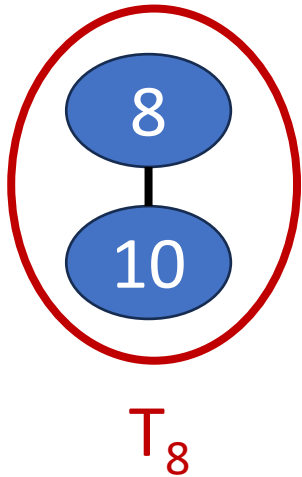
- 1) Remove the node pointed by min.
 - $O(1)$
- 2) Min-tree joining.
 - $O(s)$, where s is the number of min trees.
 - $s = O(n)$, since there may be several one-node subtrees.
- 3) Form min-tree root list and update min pointer.
 - The final number of min trees $\leq s$.

Overall complexity of delete-min is $O(n)$.

Pairwise combine

- Can be implemented using an array *tree* to keep track of trees by degree.

Example:



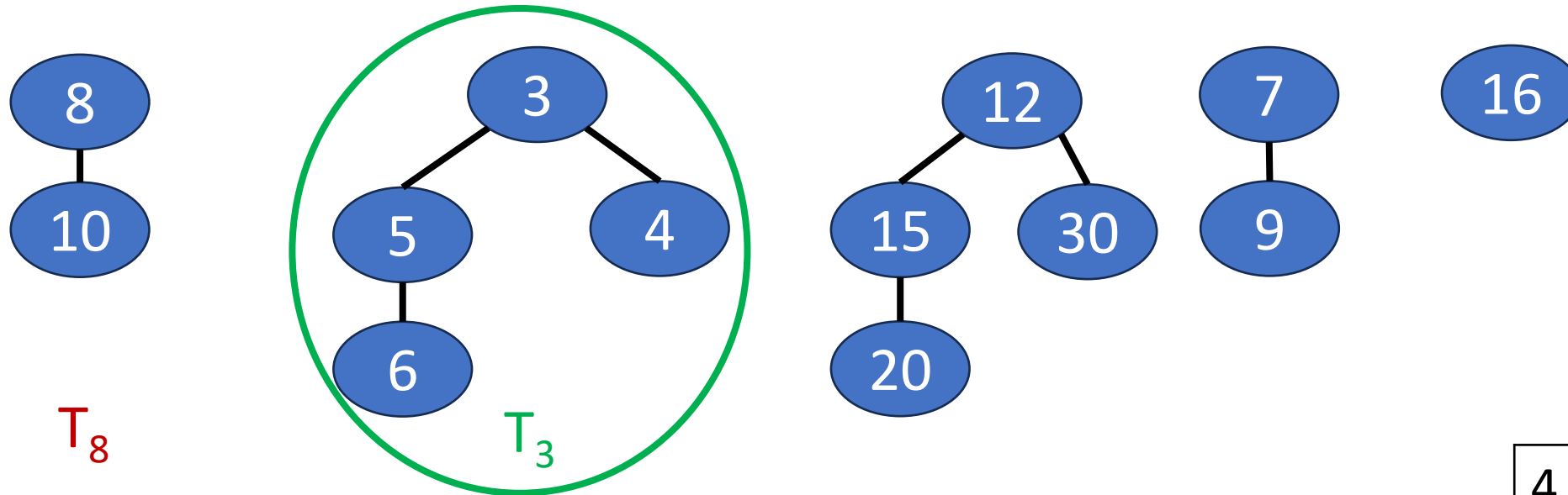
tree

4	
3	
2	
1	T_8
0	

Visiting the min trees following the top-level circular list.
Recording the degree of visited min trees.

Pairwise combine

Example:



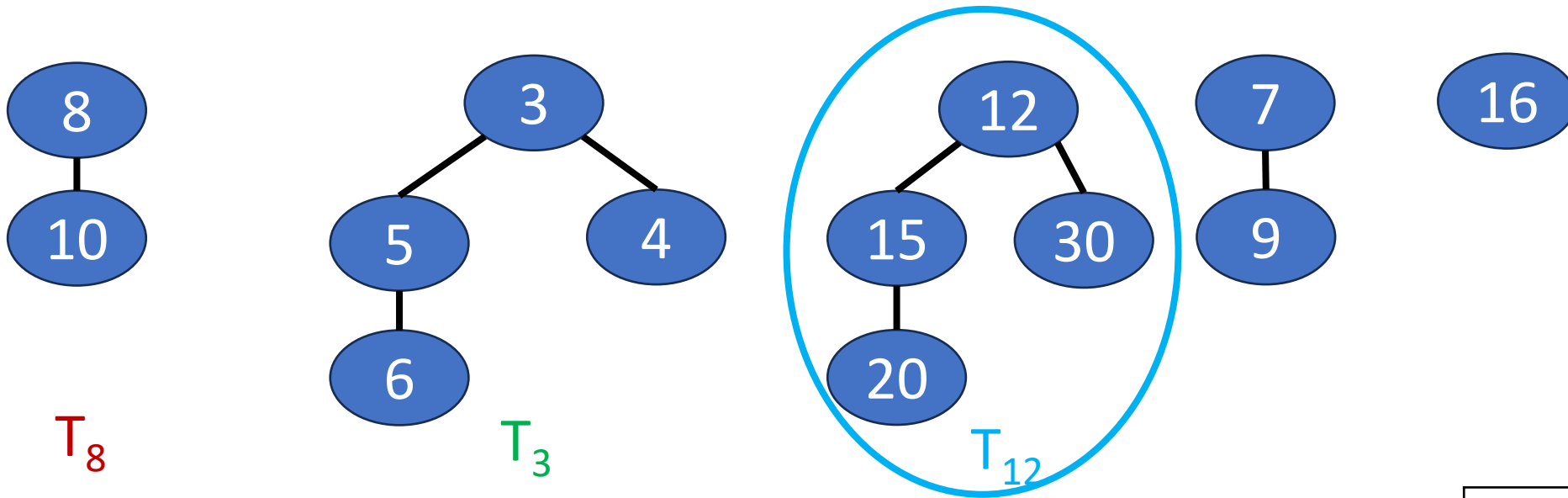
Visiting the next min tree.
Recording the degree of visited min trees.

tree

4	
3	
2	T_3
1	T_8
0	

Pairwise combine

Example:



Visiting the next min tree.

Recording the degree of visited min trees.

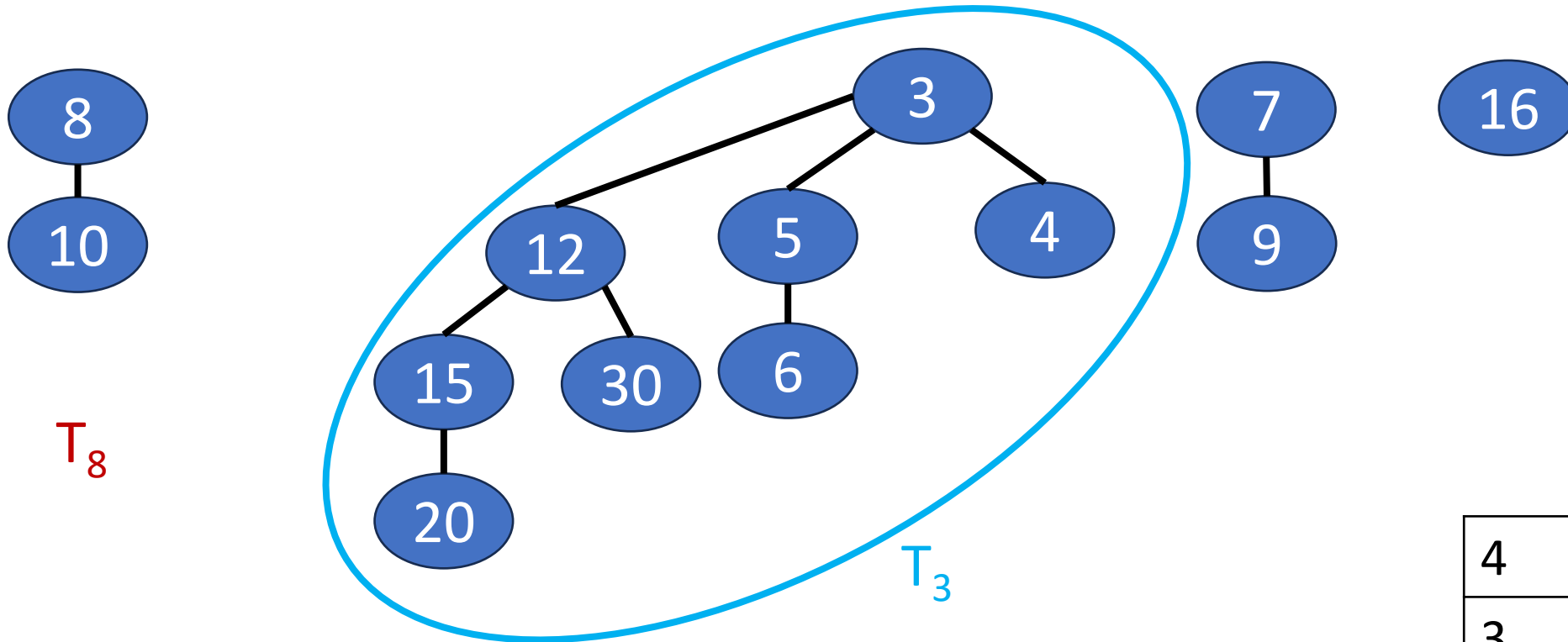
Finding another min tree T_3 having same degree. → Meld them.

tree

4	
3	
2	T_3
1	T_8
0	

Pairwise combine

Example:



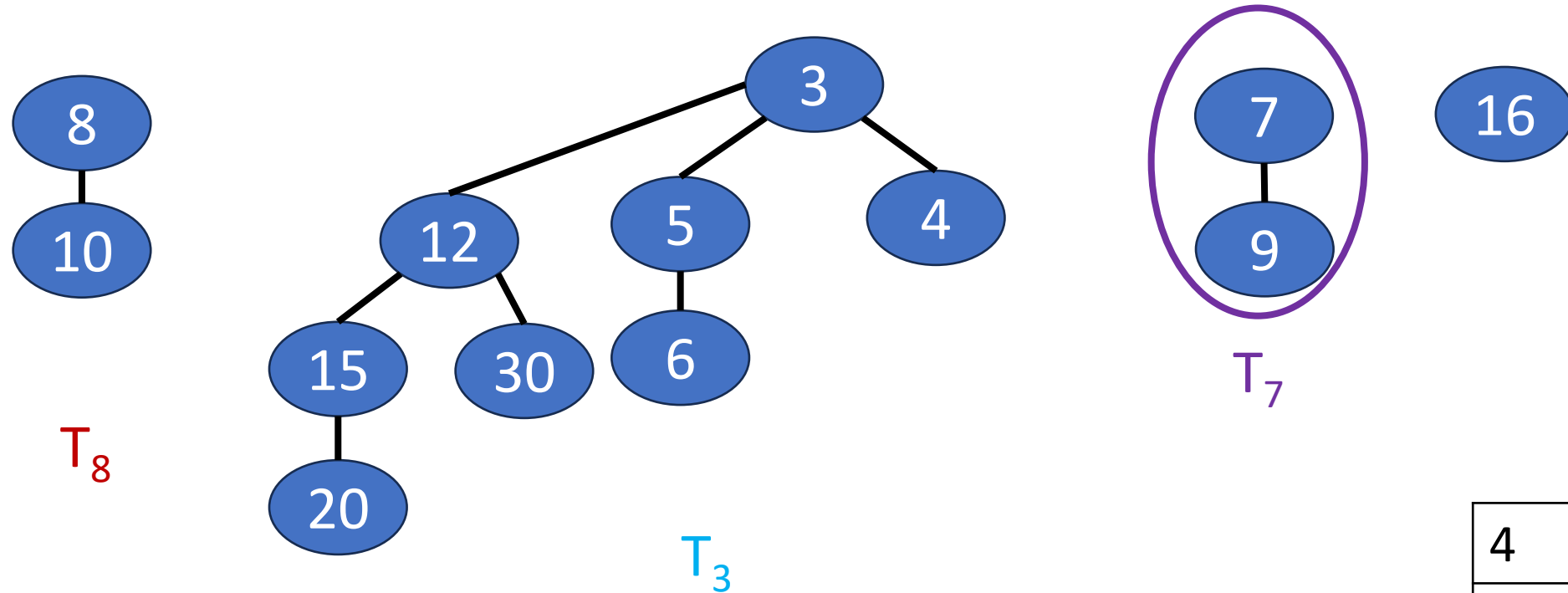
tree

4	
3	T_3
2	
1	T_8
0	

Recording the degree of new min-tree.

Pairwise combine

Example:



Visiting the next min tree.

Recording the degree of visited min trees.

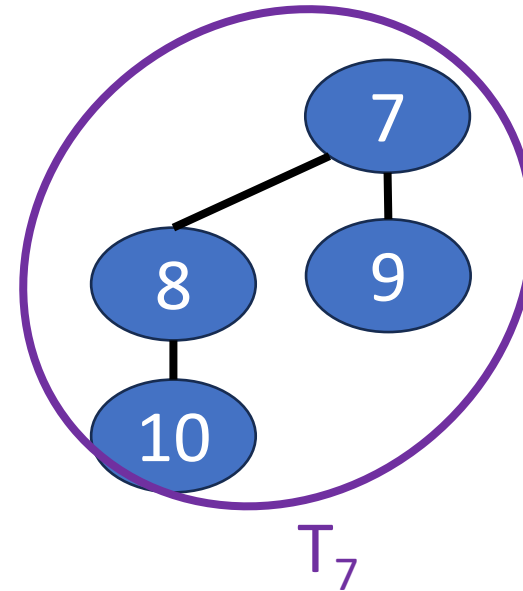
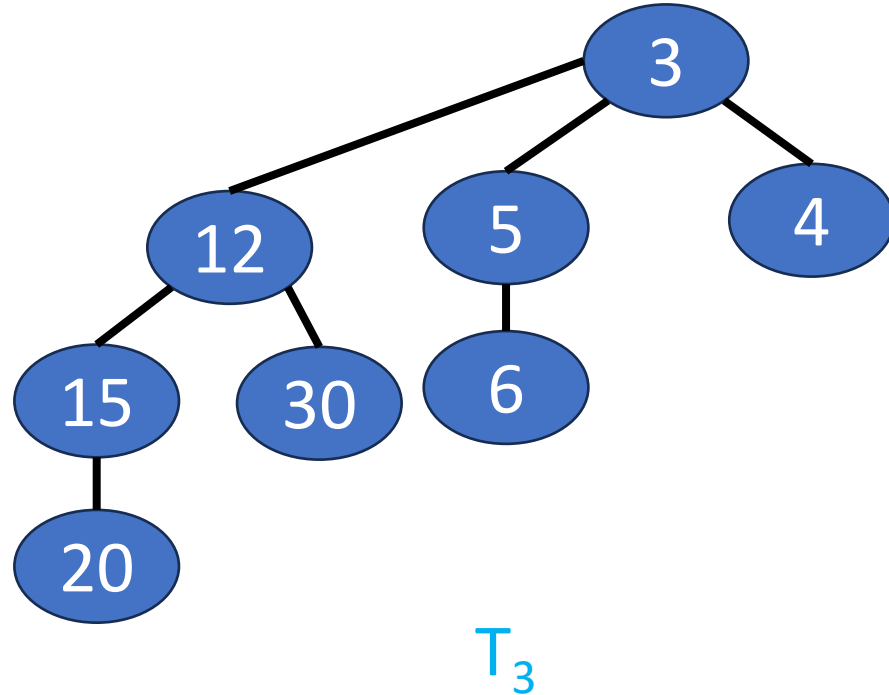
Finding another min tree T_8 having same degree. → Meld them.

tree

4	
3	T_3
2	
1	T_8
0	

Pairwise combine

Example:



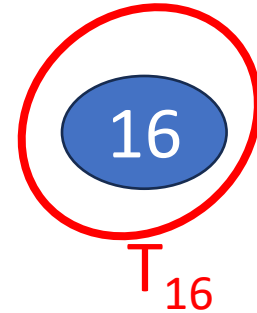
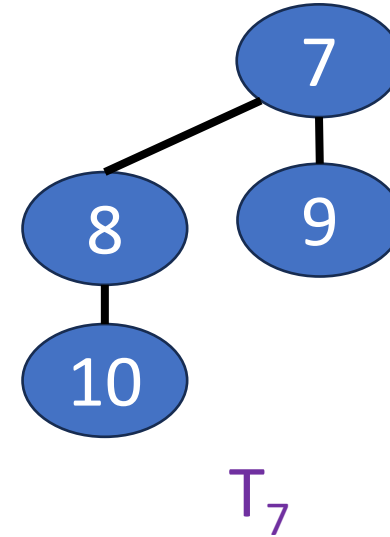
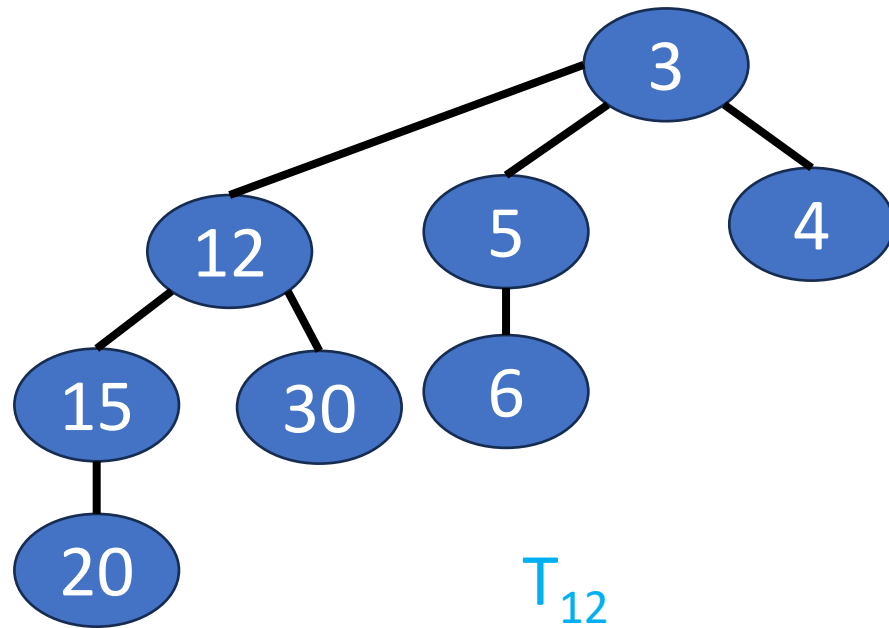
tree

4	
3	T_3
2	T_7
1	
0	

Recording the degree of new min-tree.

Pairwise combine

Example:



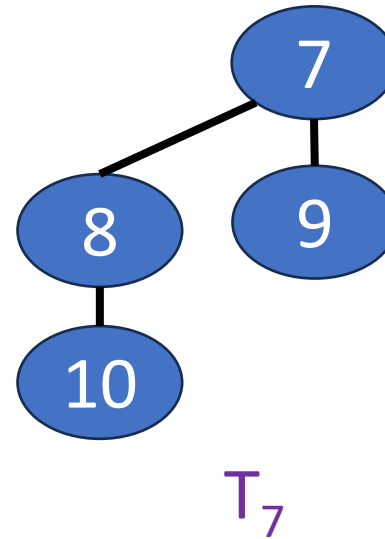
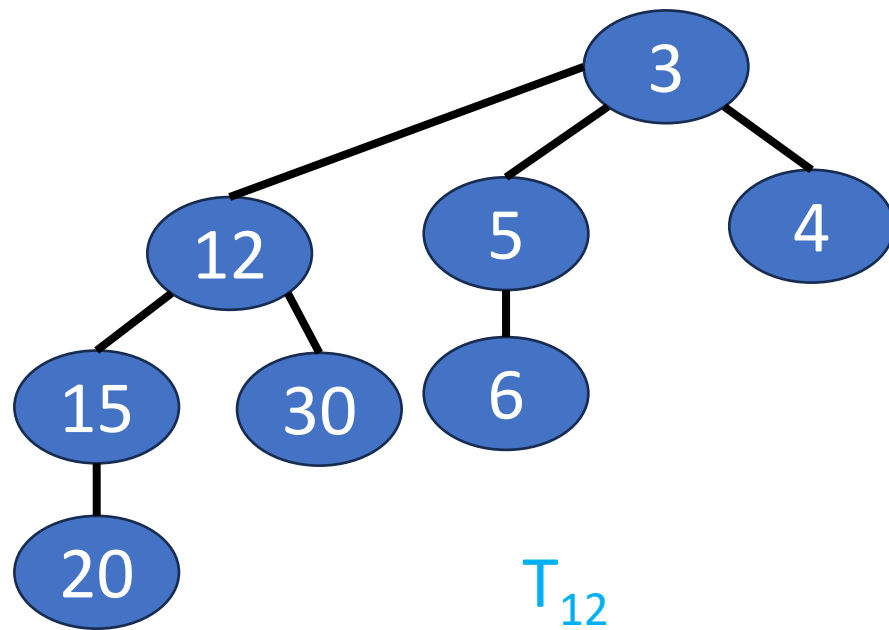
Visiting the next min tree.
Recording the degree of visited min trees.

tree

4	
3	T_{12}
2	T_7
1	
0	T_{16}

Pairwise combine

Example:



tree

4	
3	T_{12}
2	T_7
1	
0	T_{16}

Create circular list of remaining trees in the *tree* table.

Complexity of Delete-min

2) Min-tree joining.

- Create and initialize tree table.
 - $O(\text{MaxDegree})$.
 - Done once only.
- Examine s min trees and pairwise combine.
 - $O(s)$, where s is the number of min trees.

Number of joins is at most $s-1$ as each join reduces number of min-trees by 1.

3) Form min-tree root list and update min pointer.

- Collect remaining trees from tree table, reset table entries to **null**, and set min pointer.
 - $O(\text{MaxDegree})$.

Overall complexity of delete-min is $O(\text{MaxDegree} + s)$.

Exercise

- Q8: Into an empty B-heap, insert elements with priorities **20, 10, 5, 18, 6, 12, 14, 4, and 22** (in this order). Each insertion operation includes **min-tree joining (pairwise combine)**. Please write the roots and degrees of min trees in the final B-heap.
- Q9: Delete the min element from the final B-heap of Q8. Please write the roots and degrees of min trees in the resulting B-heap.

Please reply your answers of Q8
via the following link:



<https://forms.gle/rzSHma4tLUURKEzw5>

Group members: 2~4 people

Exercise

- Q9: Delete the min element from the final B-heap of Q8. Please write the roots and degrees of min trees in the resulting B-heap.

Please reply your answers of Q9
via the following link:



<https://forms.gle/rzSHma4tLUURKEzw5>

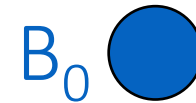
Group members: 2~4 people

Binomial trees

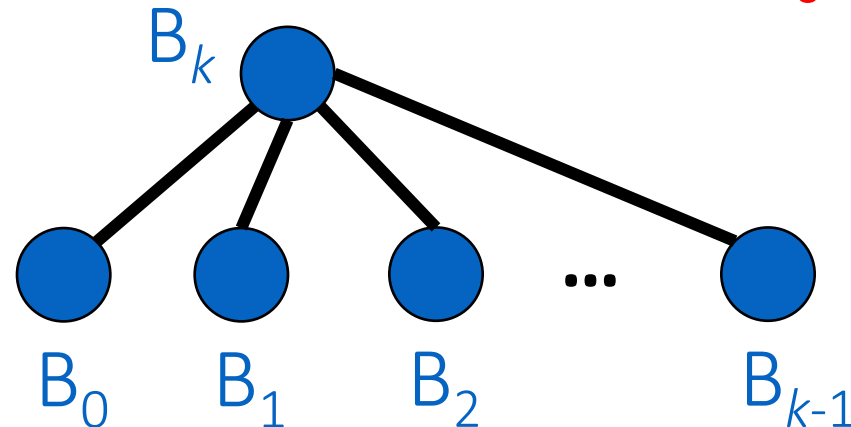
- If performing inserts, melds, and delete-mins only, the min trees in B-heap are **binomial trees**.

- B_k is degree- k binomial tree.

- $k=0$, the tree has one node.



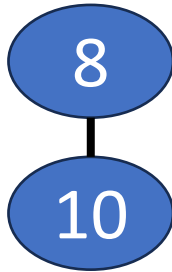
- $k>0$, root with degree k and subtrees are B_0, B_1, \dots, B_{k-1} .



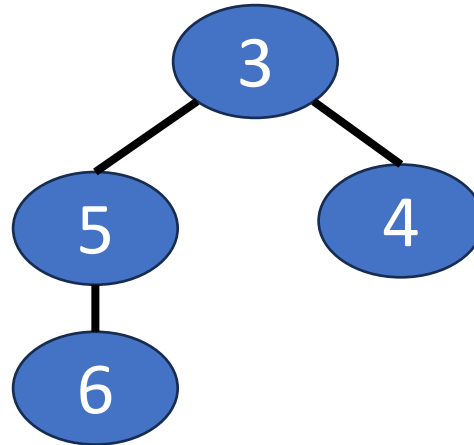
Example of binomial trees



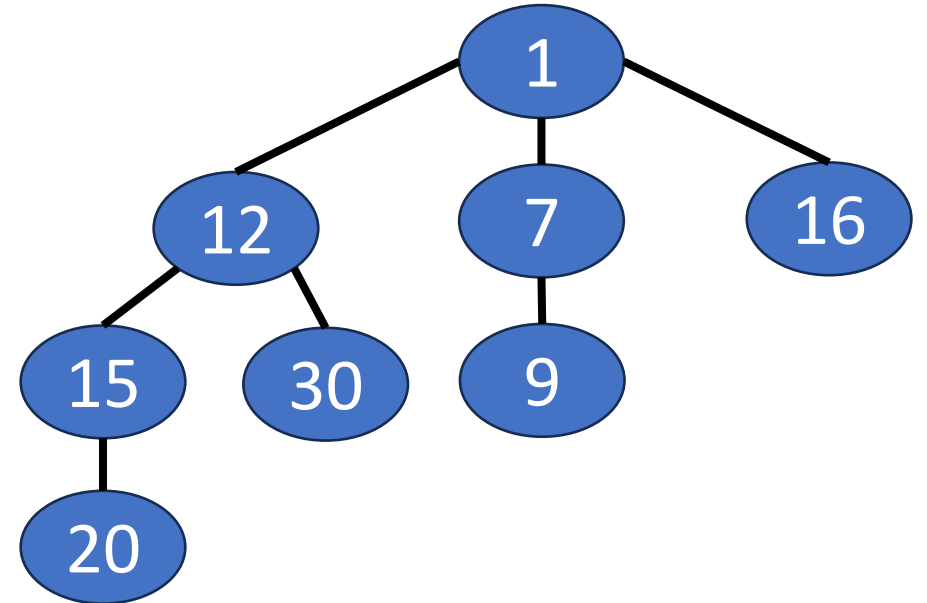
B_0



B_1



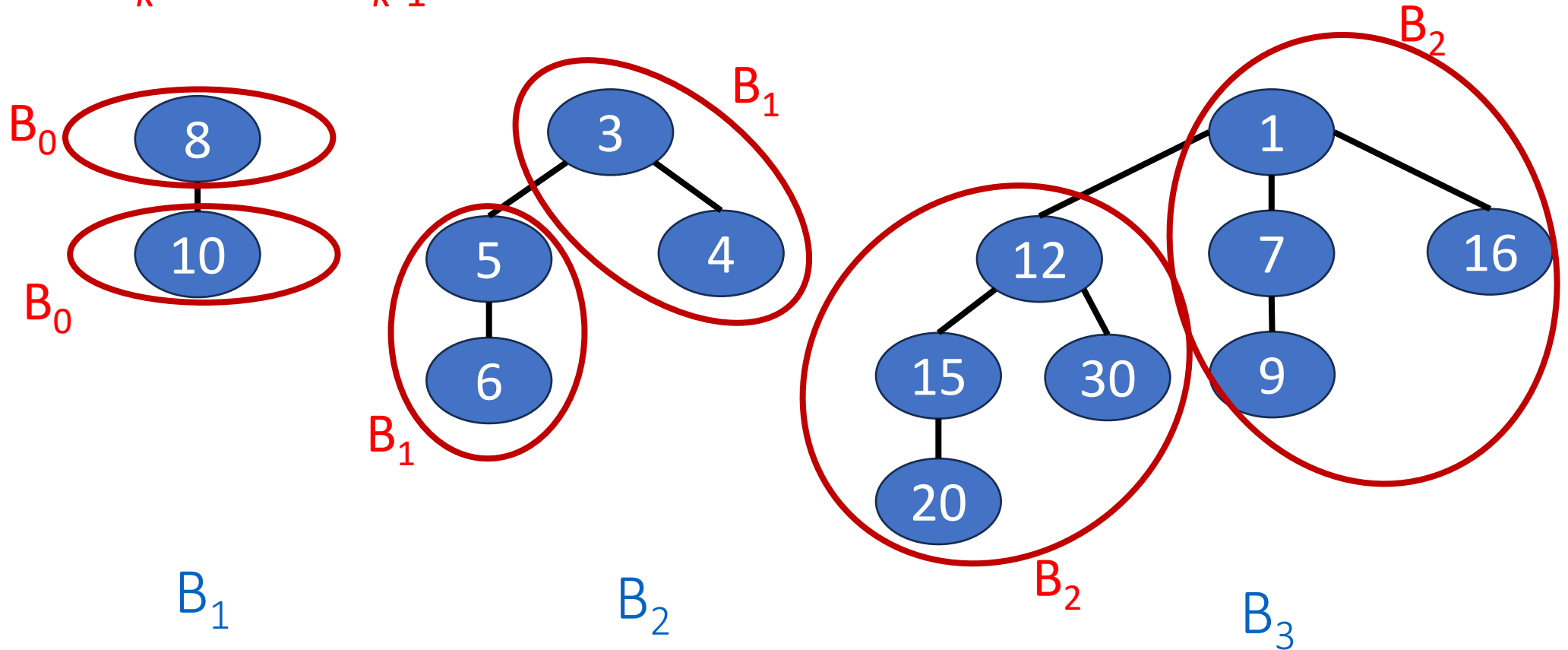
B_2



B_3

Equivalent definition of binomial trees

- B_k is degree- k binomial tree.
 - $k > 0$, B_k is two B_{k-1} .



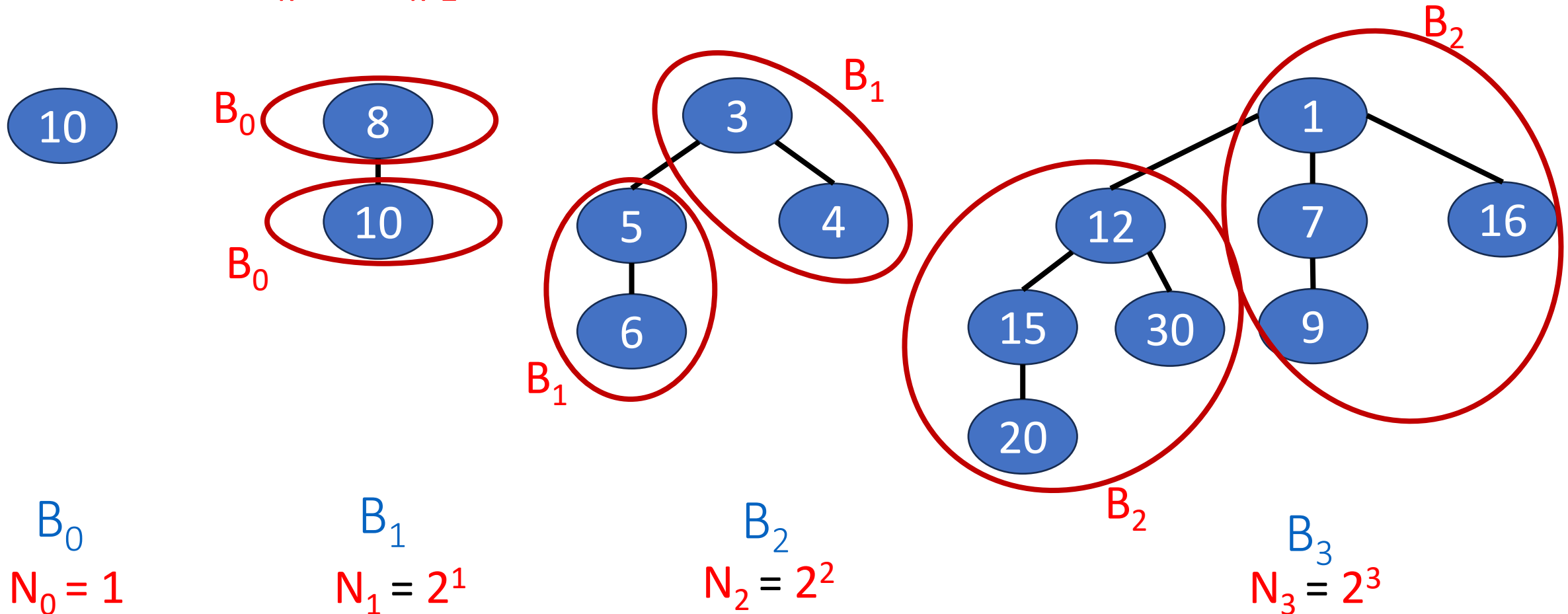
Number of nodes in binomial trees

- N_k is number of nodes in degree- k binomial tree.

- $N_0 = 1$

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

In B-heap, all trees are binomial trees.
Thus MaxDegree of a min tree is $O(\log n)$.



Performance analysis

See Theorem 9.1 in textbook for more details.

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

- Actual cost of a delete-min: $O(\text{MaxDegree} + s) = O(\log n + s)$.
- If a sequence of insert, meld, and delete-min operations is performed on empty B-heaps, we can transfer the cost of delete-min s to prior insertions.
- Amortized cost of insertion becomes $O(1+1) = O(1)$ and that of delete-min becomes $O(\text{MaxDegree}) = O(\log n)$.

Summary

- Binomial heaps
 - Operations: Insert, Meld, Delete-min, Delete-min using tree table.
- Binomial trees
 - Definition
 - Number of nodes
- Performance analysis
- Amortized cost