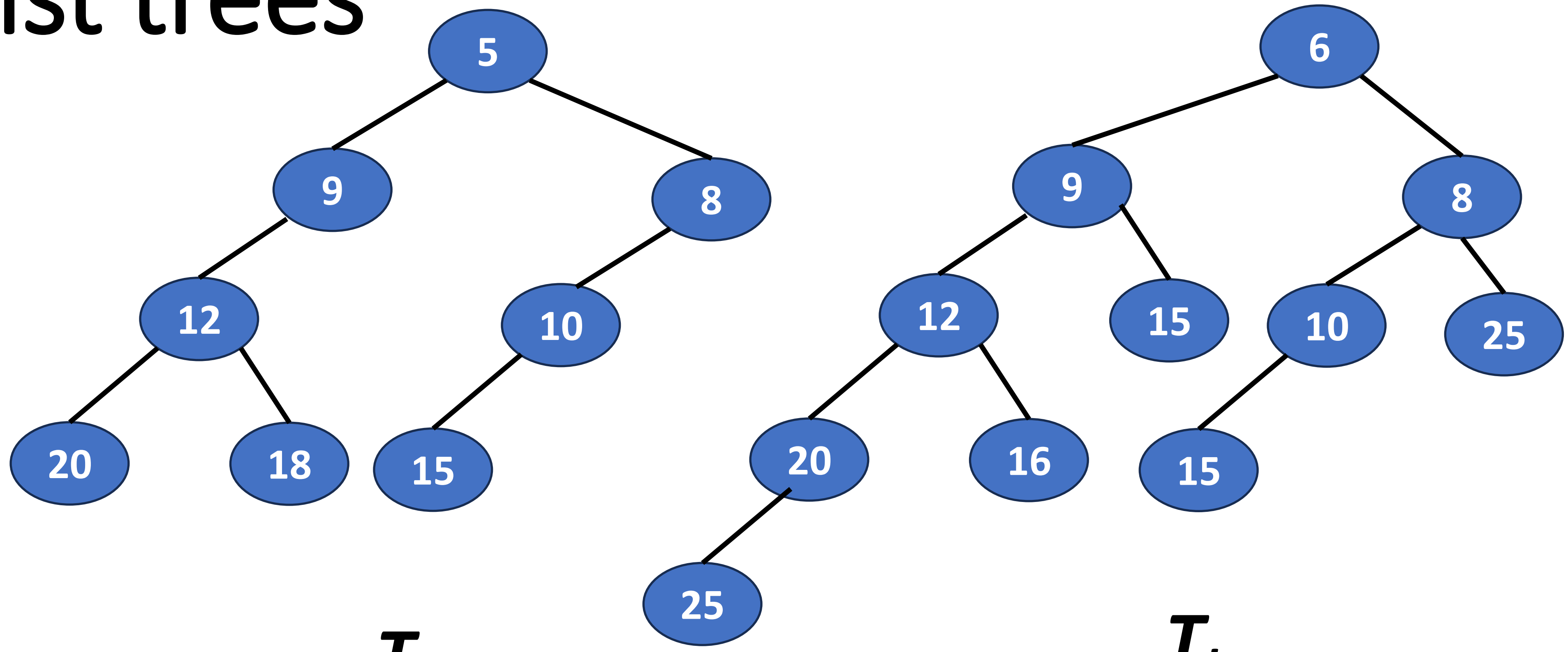
Leftist trees



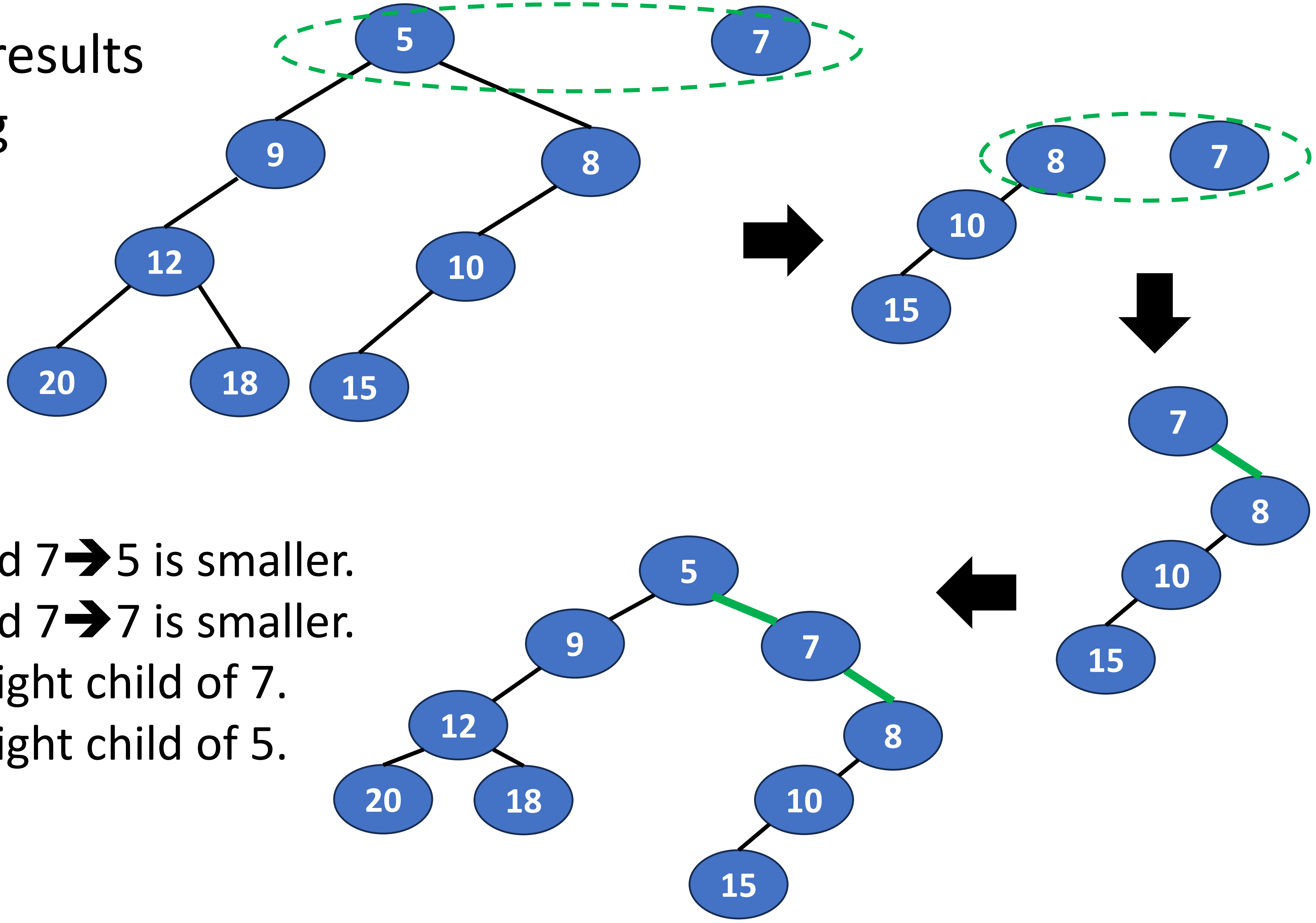
T_a ***T_b***

Given two <u>height biased leftist trees</u> T_a and T_b.

- Q1: Write out the results after executing insert(7, T_a).
- Q2: Write out the results after executing deleteMin(T_b) once.

Please use array representation to show the answers.

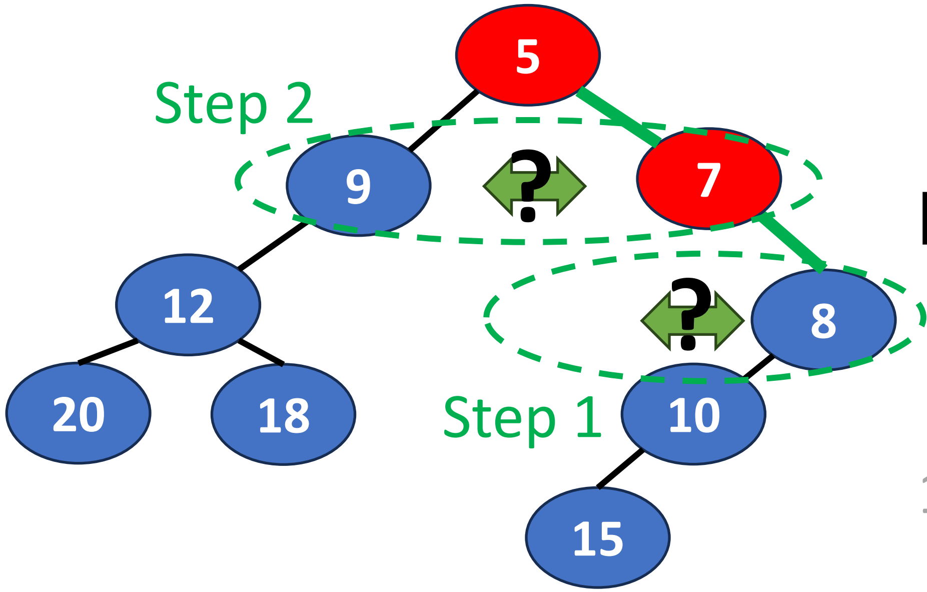
Write out the results after executing insert(7, T_a).



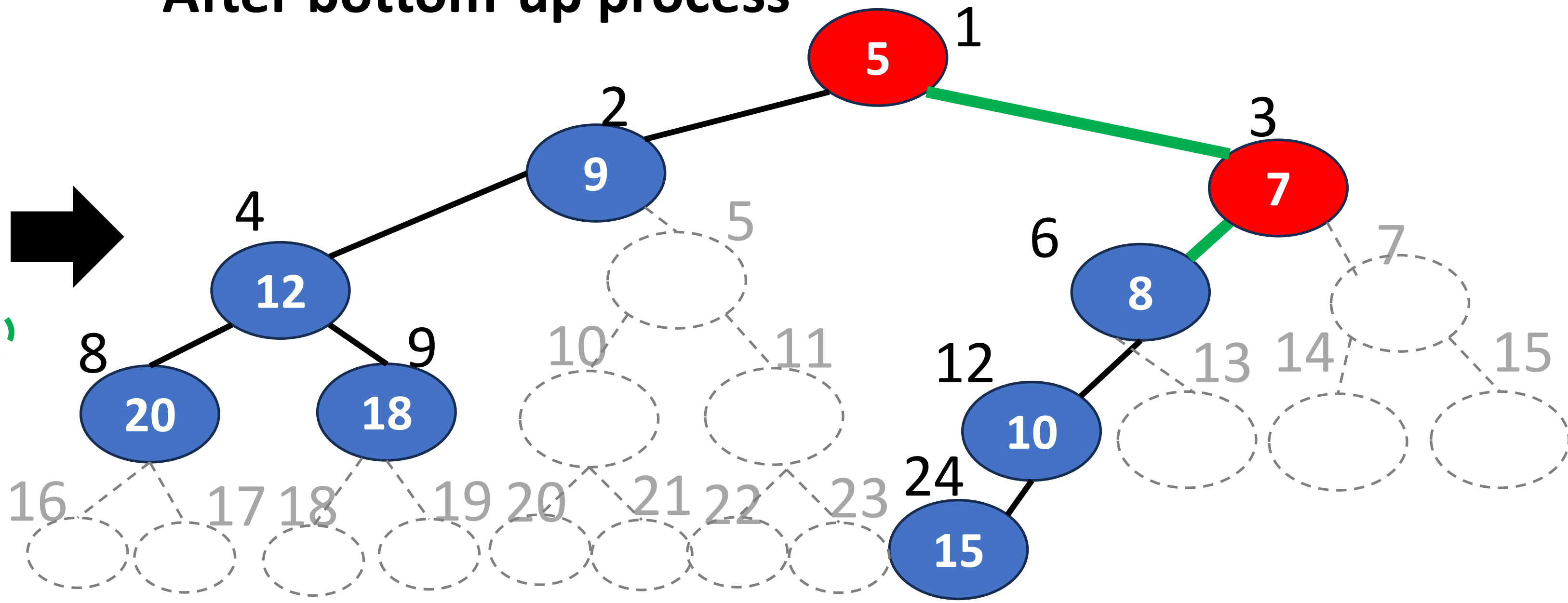
Top-down phase

1. Compare 5 and 7➔5 is smaller.
2. Compare 8 and 7➔7 is smaller.
3. Add 8 as the right child of 7.
4. Add 7 as the right child of 5.

After top-down process



After bottom-up process



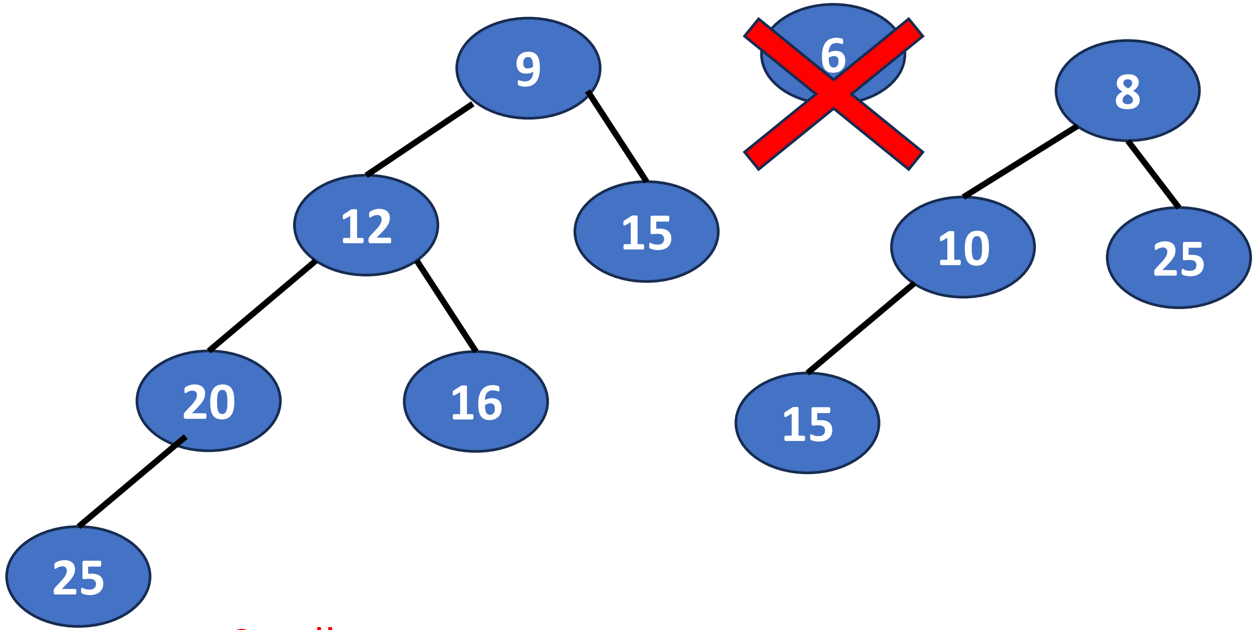
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	…	[24]
-	5	9	7	12	-	8	-	20	18	-	-	10	-	…	15

Please review Ch 5.2.3 for array representation of the binary trees.

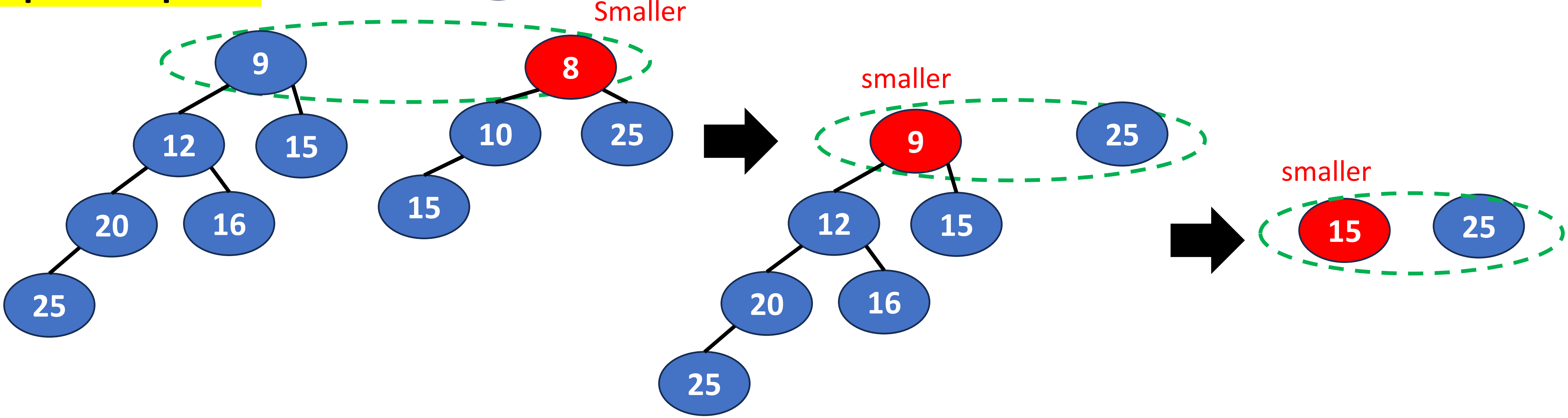
Bottom-up phase

1. Compare shortest(left child of 7) and shortest(right child of 7)
   ➔Left child has a smaller value of shortest().
   ➔ Interchange left and right subtrees of 7.
2. Compare shortest(left child of 5) and shortest(right child of 5)
   ➔Equal ➔ No interchange

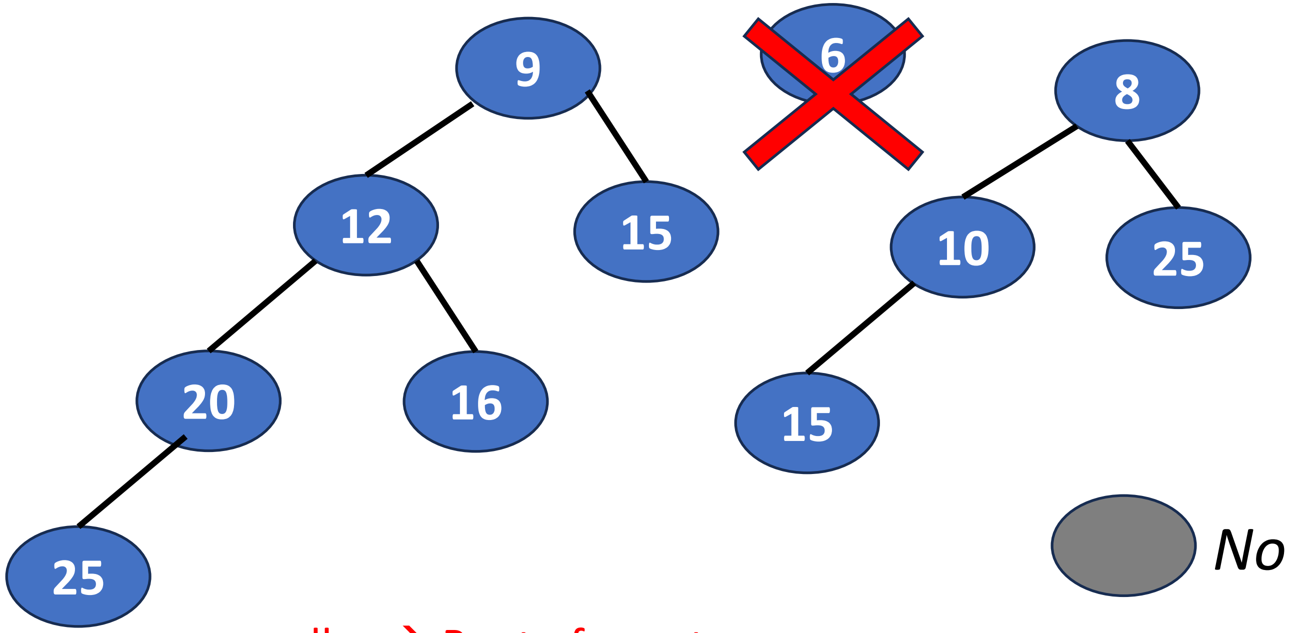
Write out the results after executing deleteMin(T_b) once.



Top-down phase

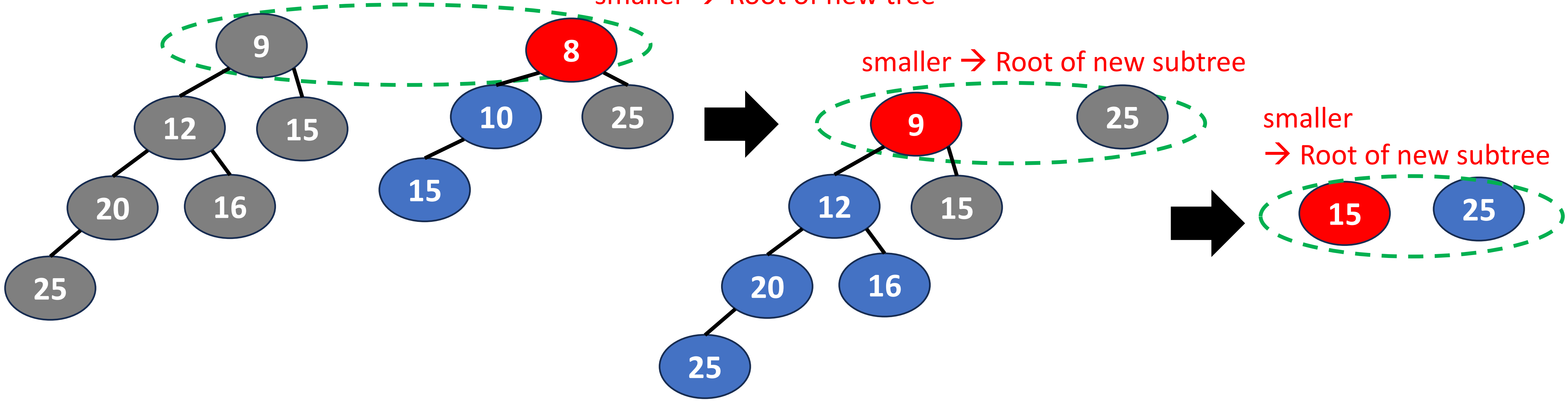


Write out the results after executing deleteMin(T_b) once.



● *Nodes to be melded*

Top-down phase

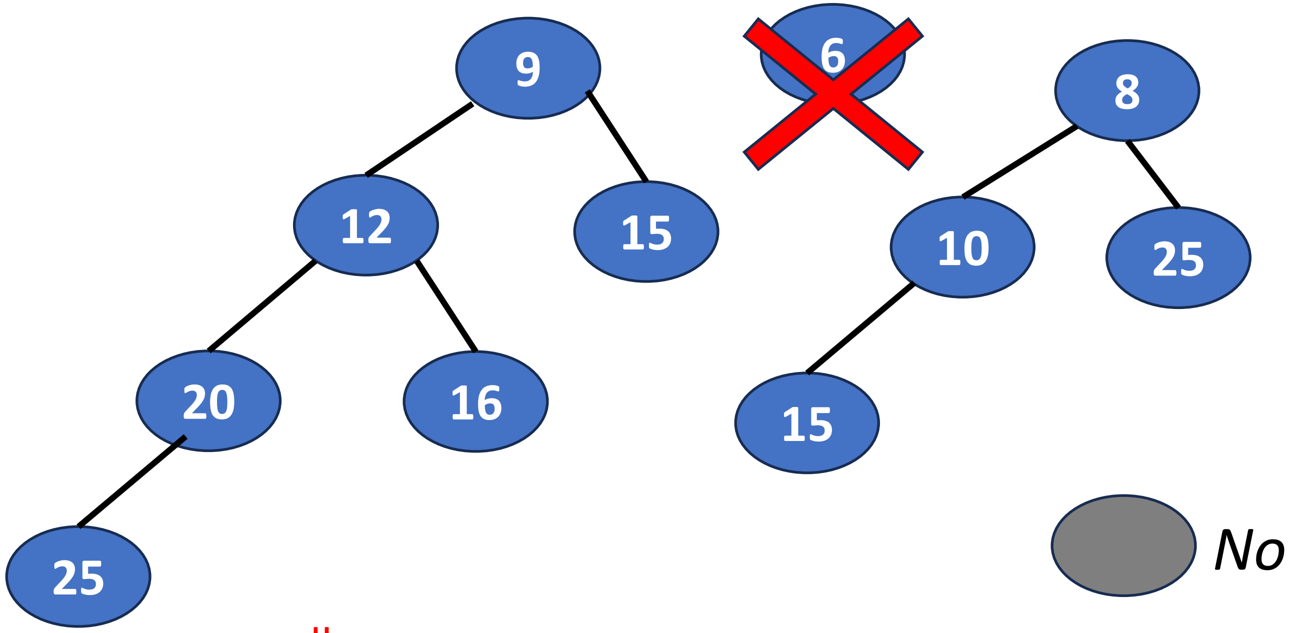


smaller → Root of new tree

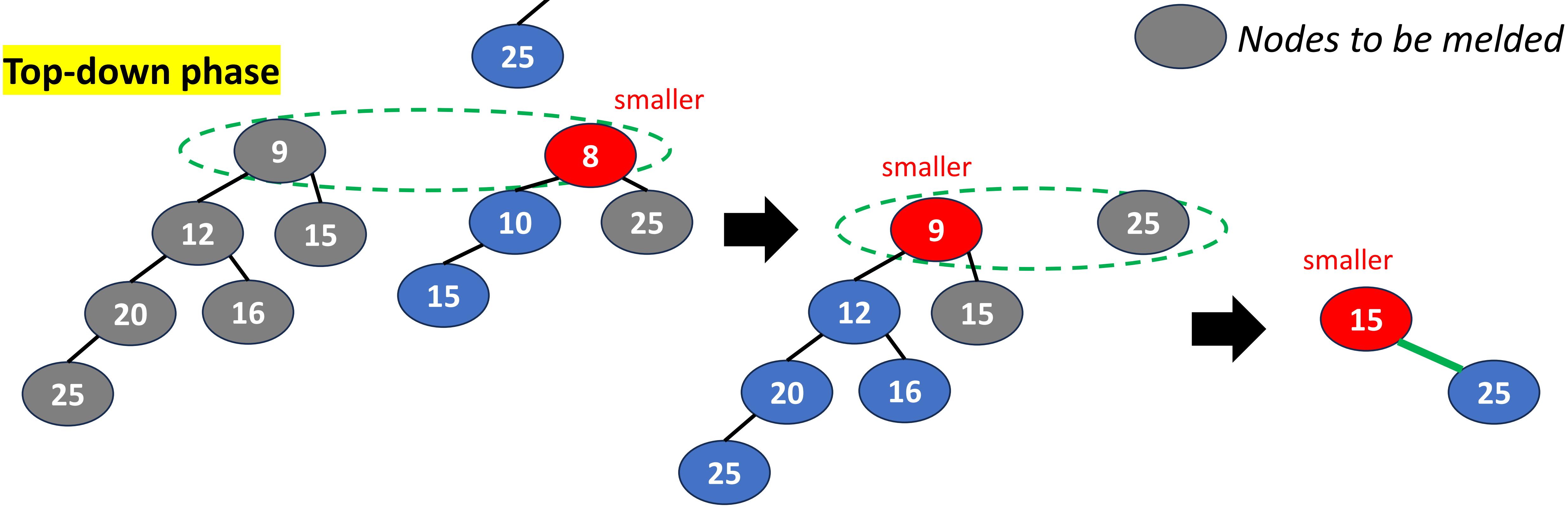
smaller → Root of new subtree

smaller → Root of new subtree

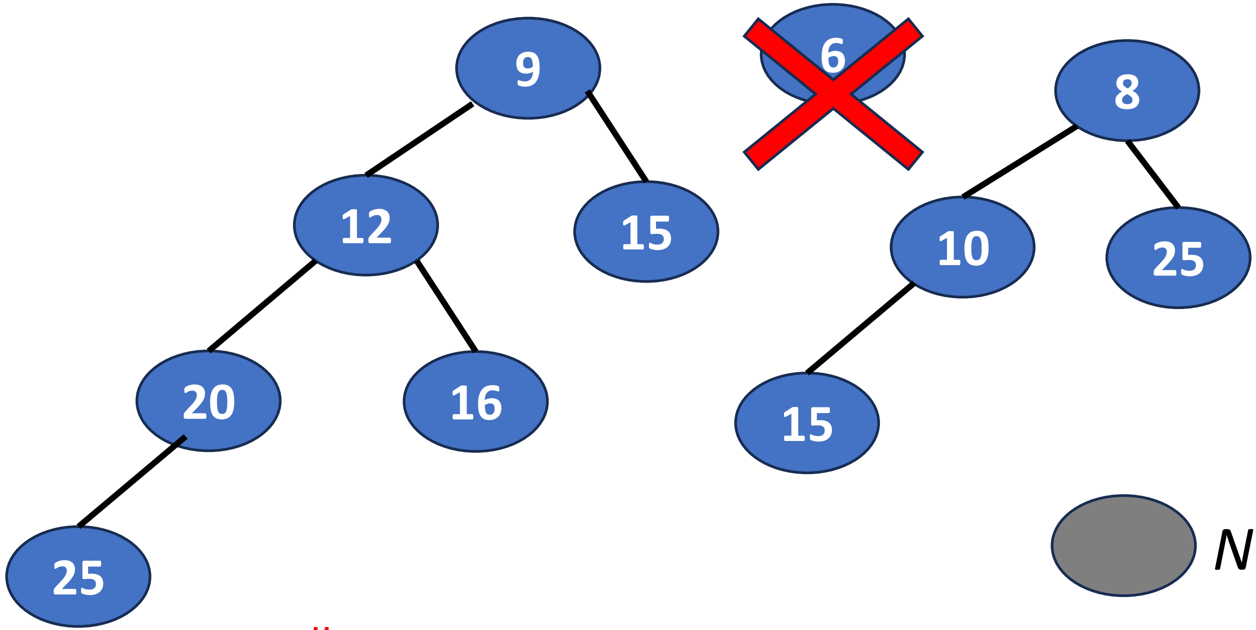
Write out the results after executing deleteMin(T_b) once.



Top-down phase

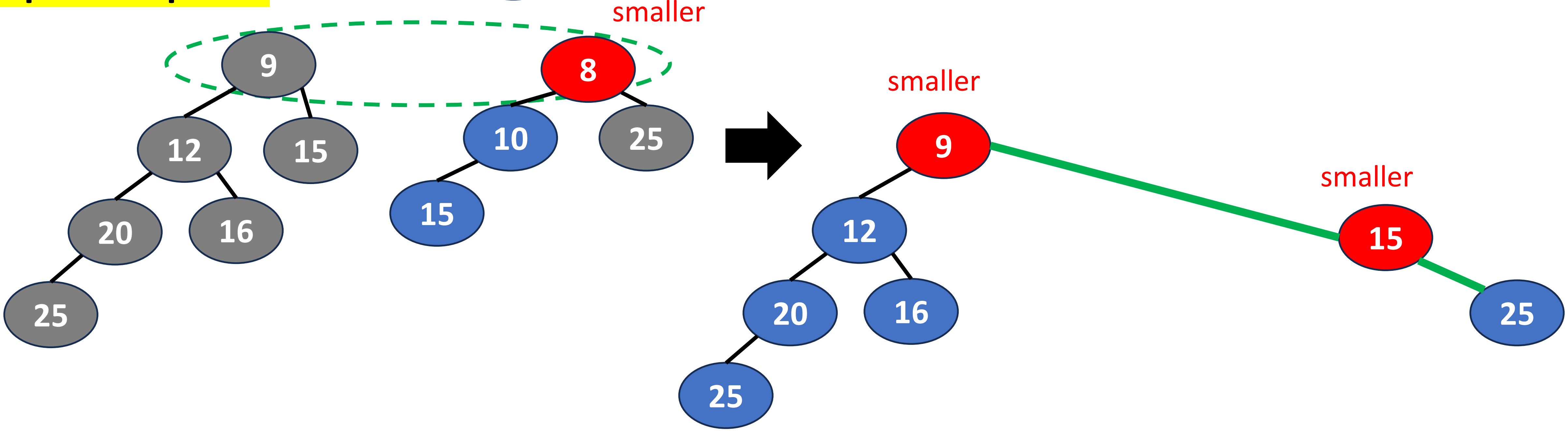


Write out the results after executing deleteMin(T_b) once.

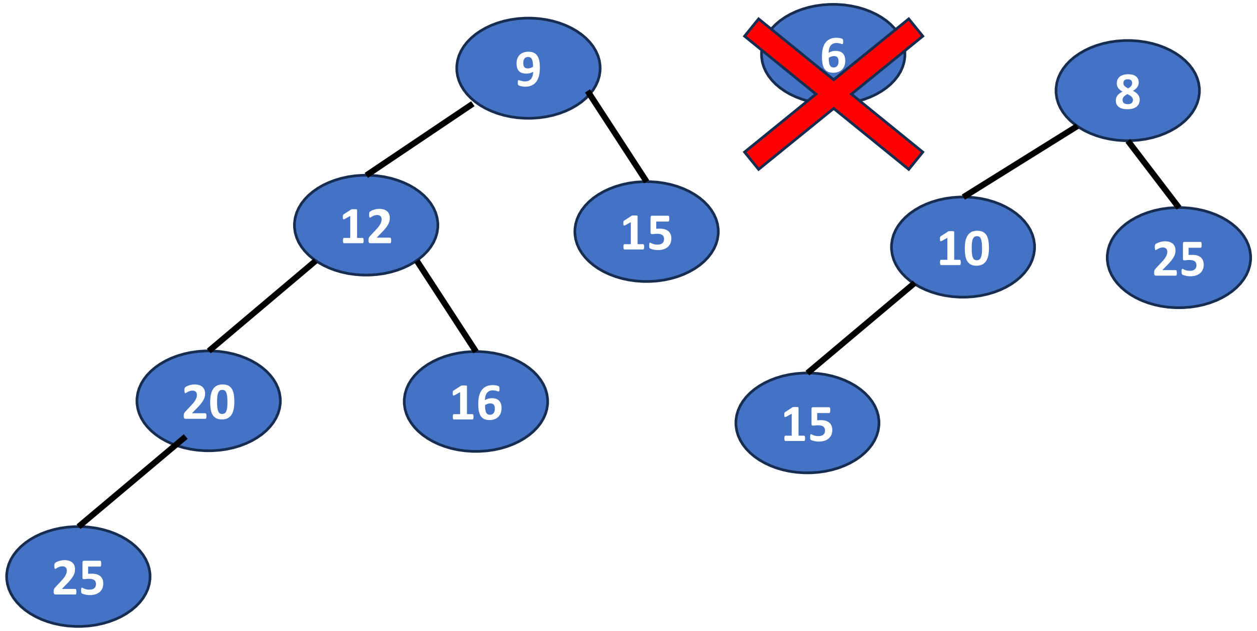


● *Nodes to be melded*

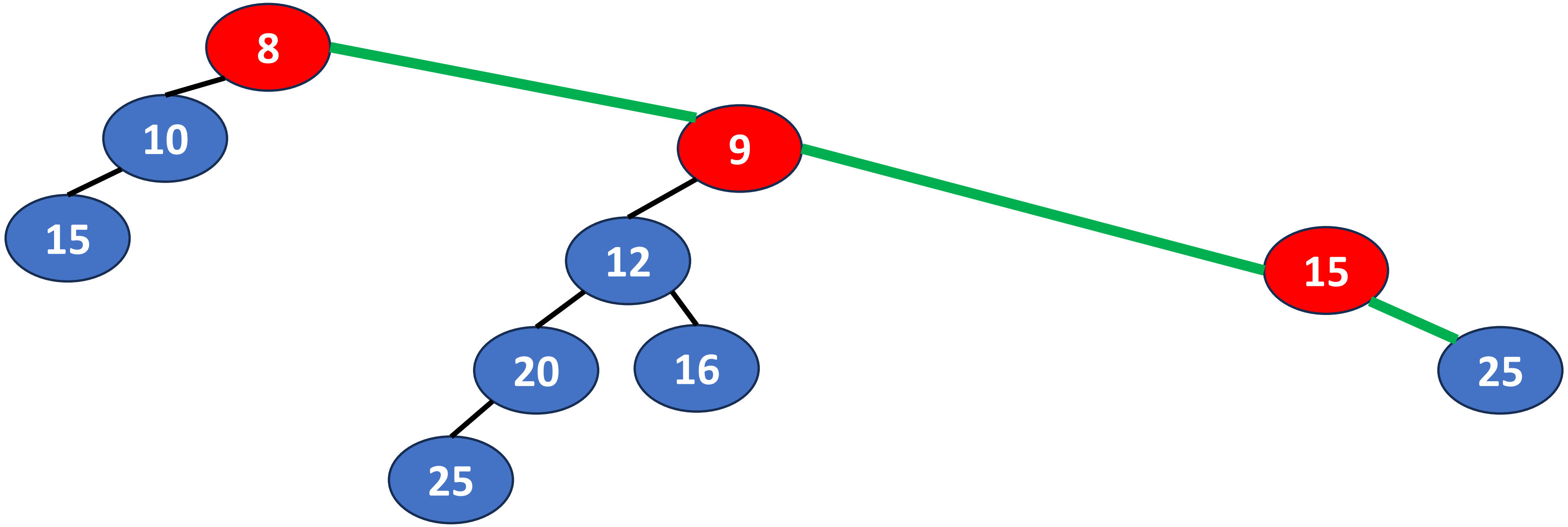
Top-down phase



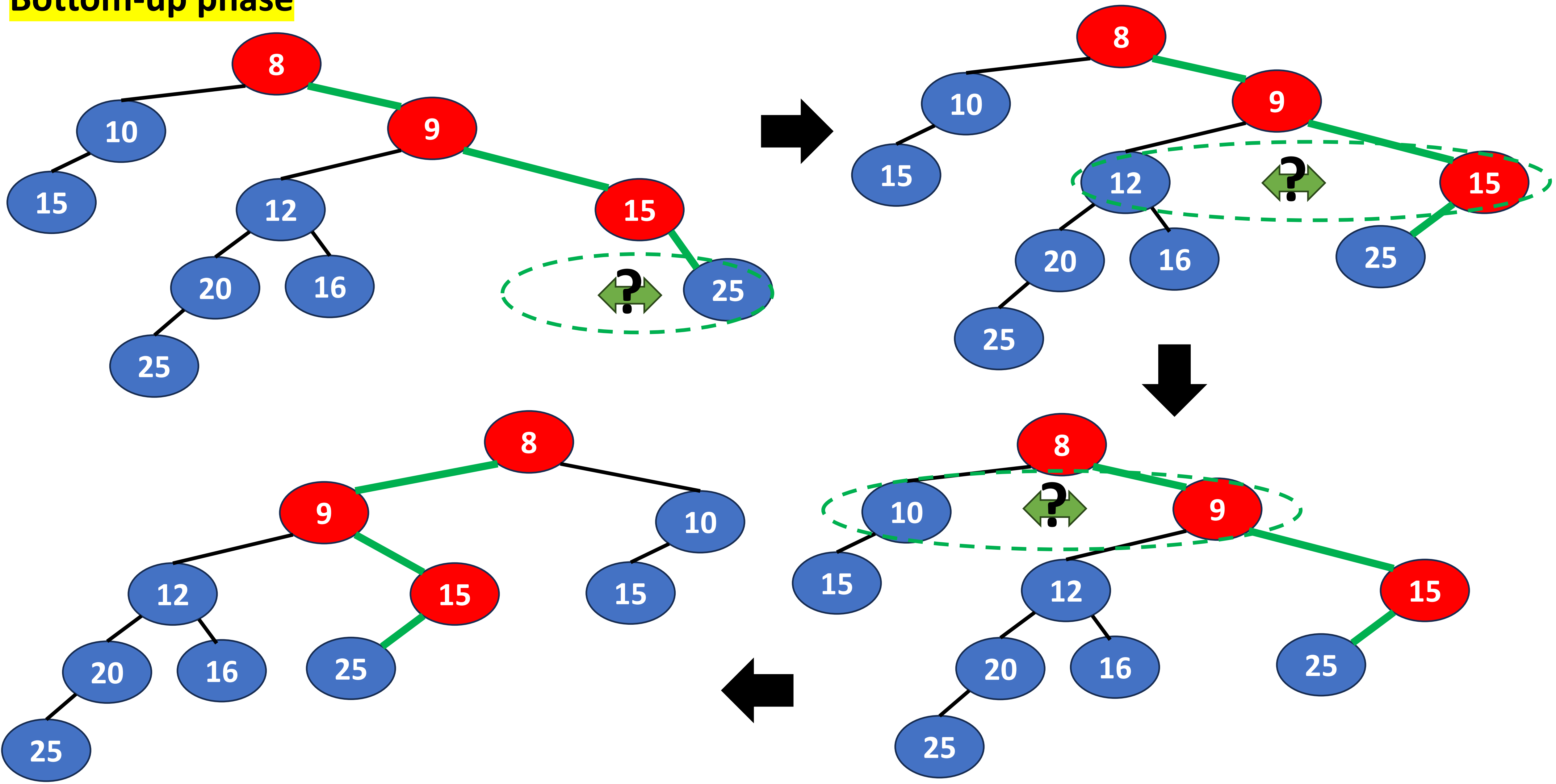
Write out the results after executing deleteMin(T_b) once.



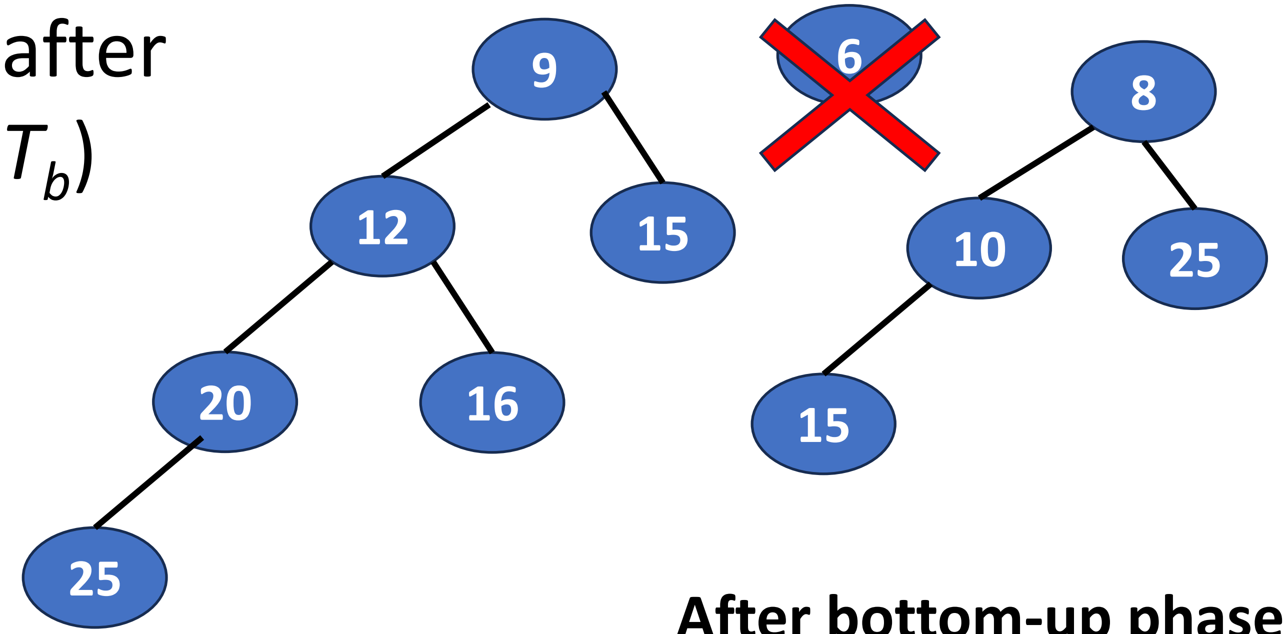
Top-down phase



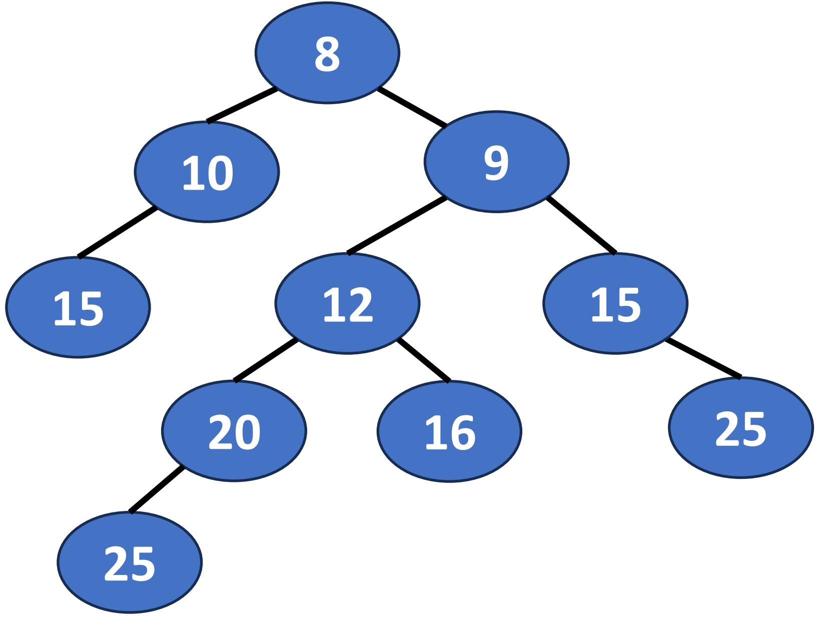
Bottom-up phase



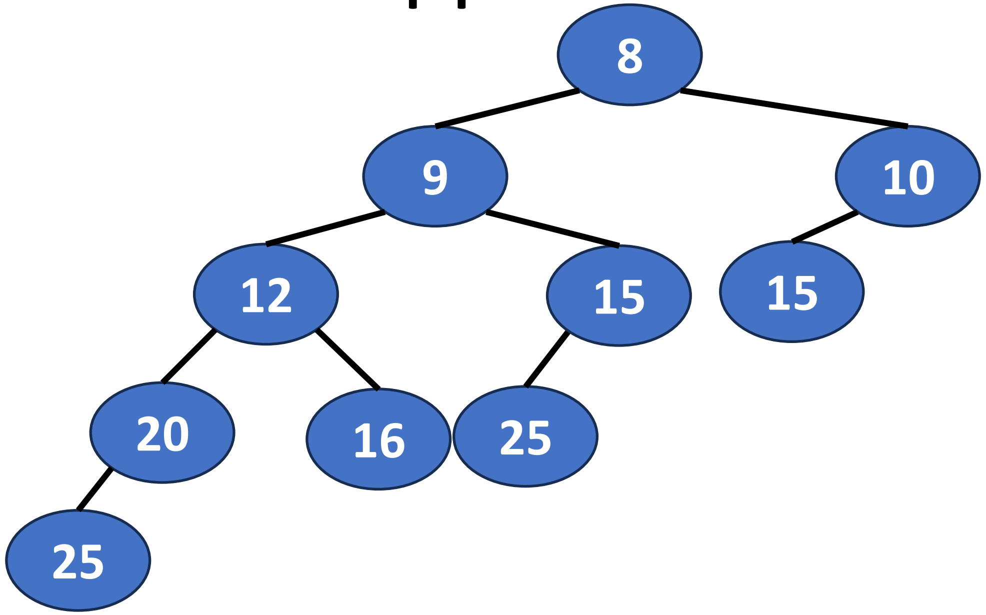
Write out the results after executing deleteMin(T_b) once.



After top-down phase

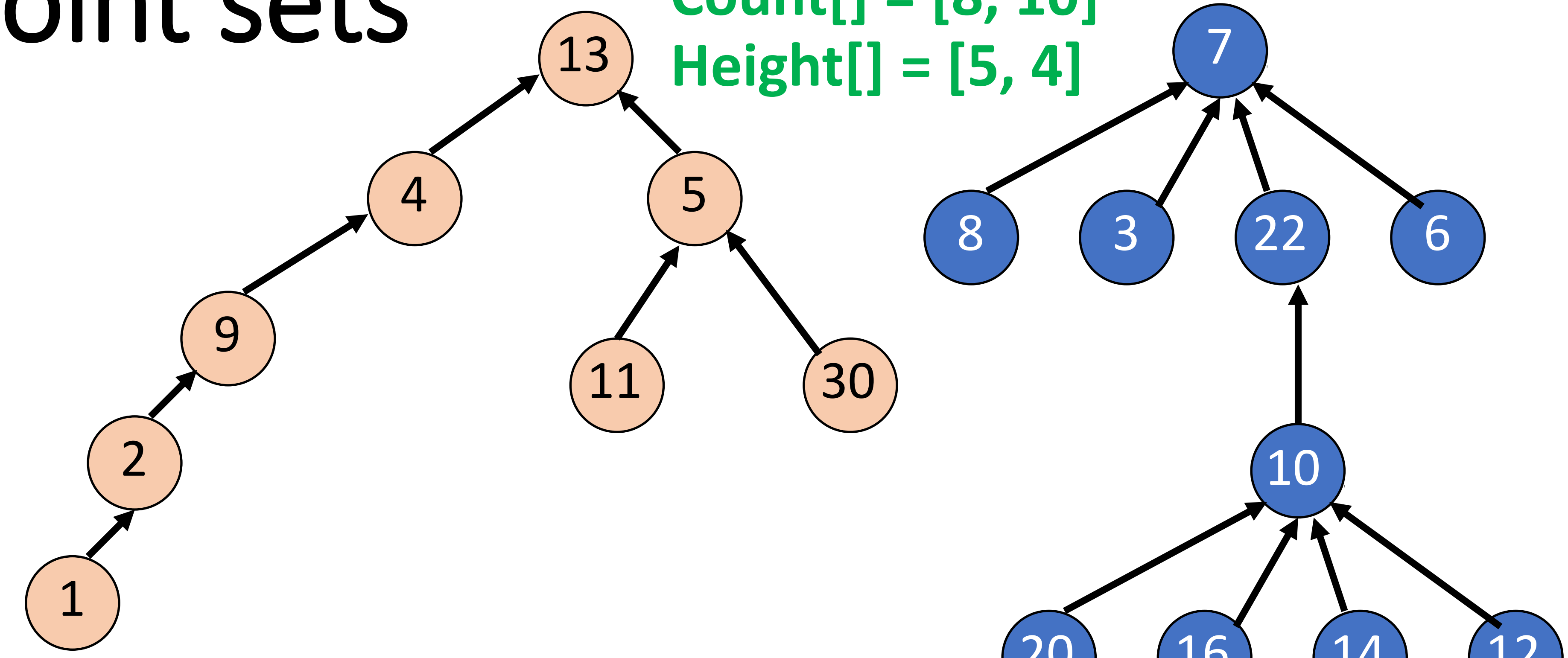


After bottom-up phase



[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]
-	8	9	10	12	15	15	-	20	16	25	-	-	-	-	-	25

Disjoint sets



Count[] = [8, 10]
Height[] = [5, 4]

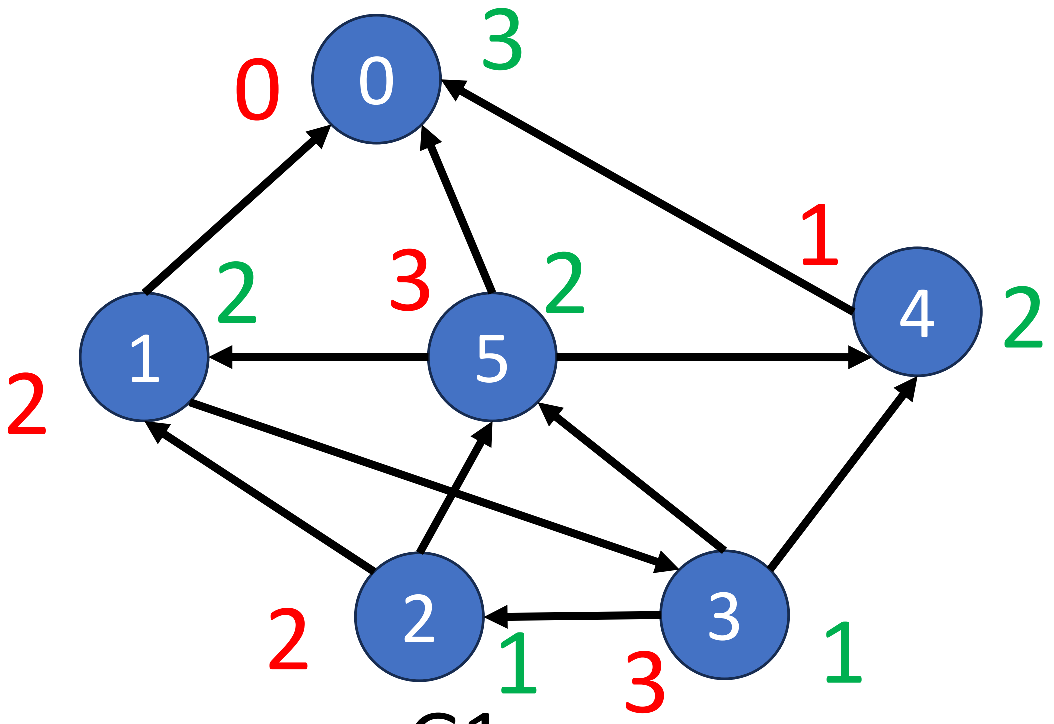
- union(7, 13)

Which tree should become a subtree of the other?

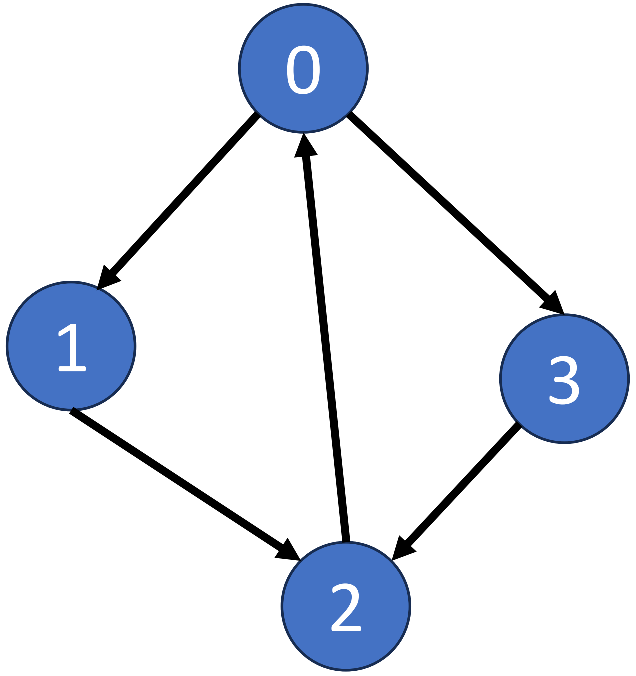
Q4: Write out the answer based on **weight** rule. Left tree

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

Graph (1)



G1



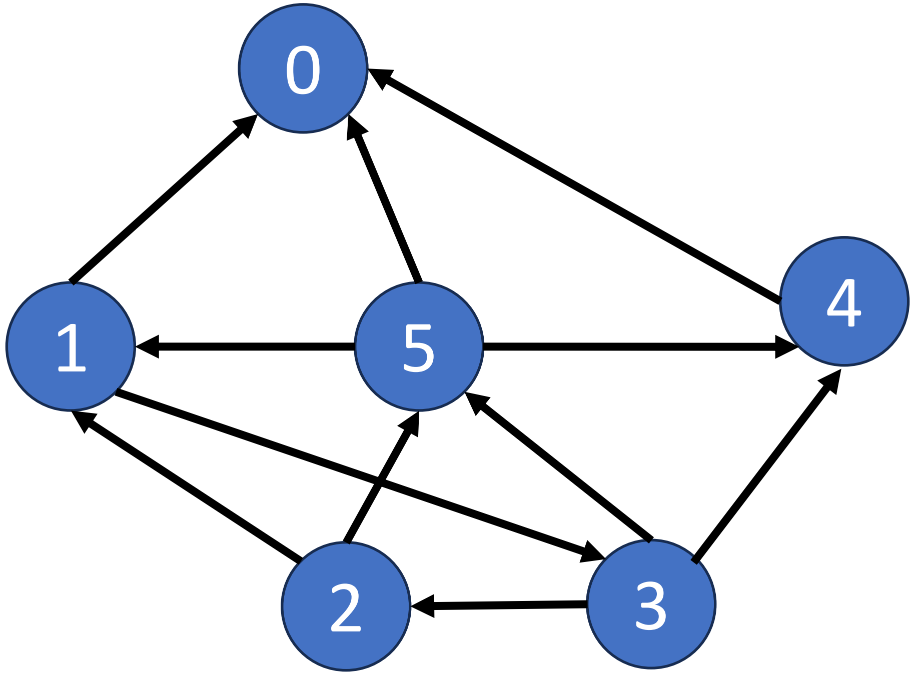
G2

0→1: 0,1
1→0: 1,2,0
0→2: 0,1,2
2→0: 2,0
0→3: 0,3
3→0: 3,2,0
1→2: 1,2
2→1: 2,0,1
1→3: 1,2,0,3
3→1: 3,2,0,1
2→3: 2,0,3
3→2: 3,2

For any pair of nodes <u, v>, there is a directed path from u to v and also from v to u.

- Q6: Write out the in-degree of vertices 0, 1, …, 5 in G1.
- Q7: Write out the out-degree of vertices 0, 1, …, 5 in G1.
- Q8: Is the directed graph G2 strongly connected? Why? YES

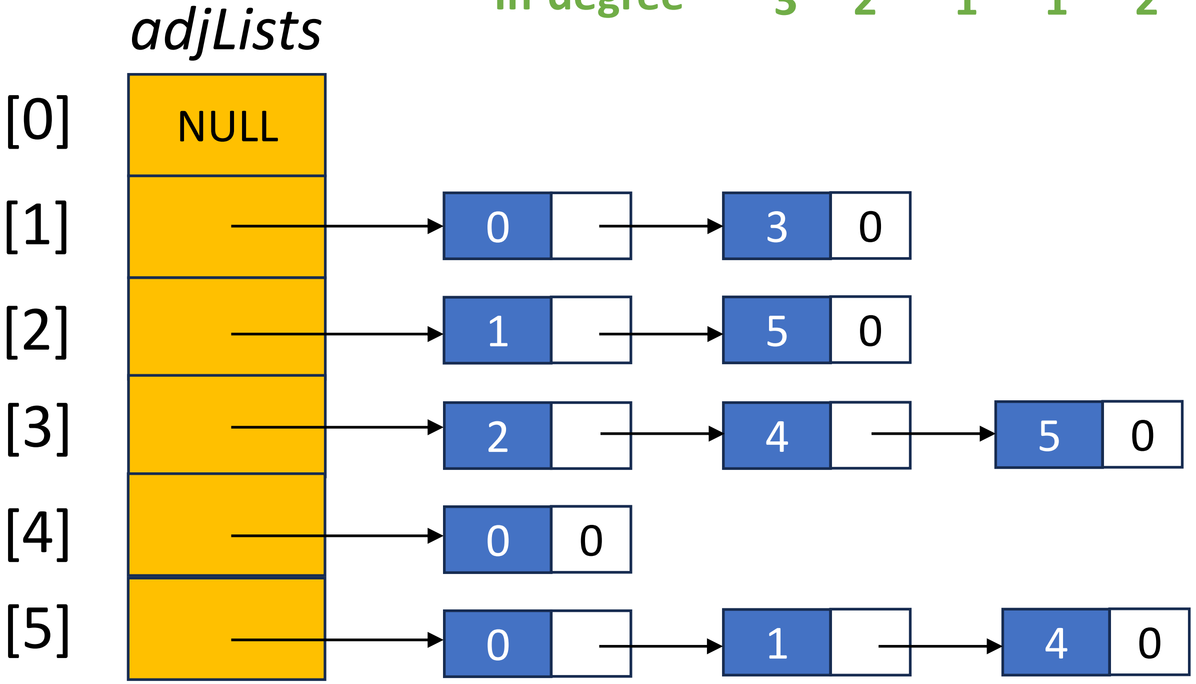
Graph (2)



- Write out the adjacency matrix.

	0	1	2	3	4	5	Out-degree
0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	2
2	0	1	0	0	0	1	2
3	0	0	1	0	1	1	3
4	1	0	0	0	0	0	1
5	1	1	0	0	1	0	3
In-degree	3	2	1	1	2	2	

- Write out the linked adjacency-list representation.



adjLists

[0] NULL

[1] → 0 → 3 0

[2] → 1 → 5 0

[3] → 2 → 4 → 5 0

[4] → 0 0

[5] → 0 → 1 → 4 0