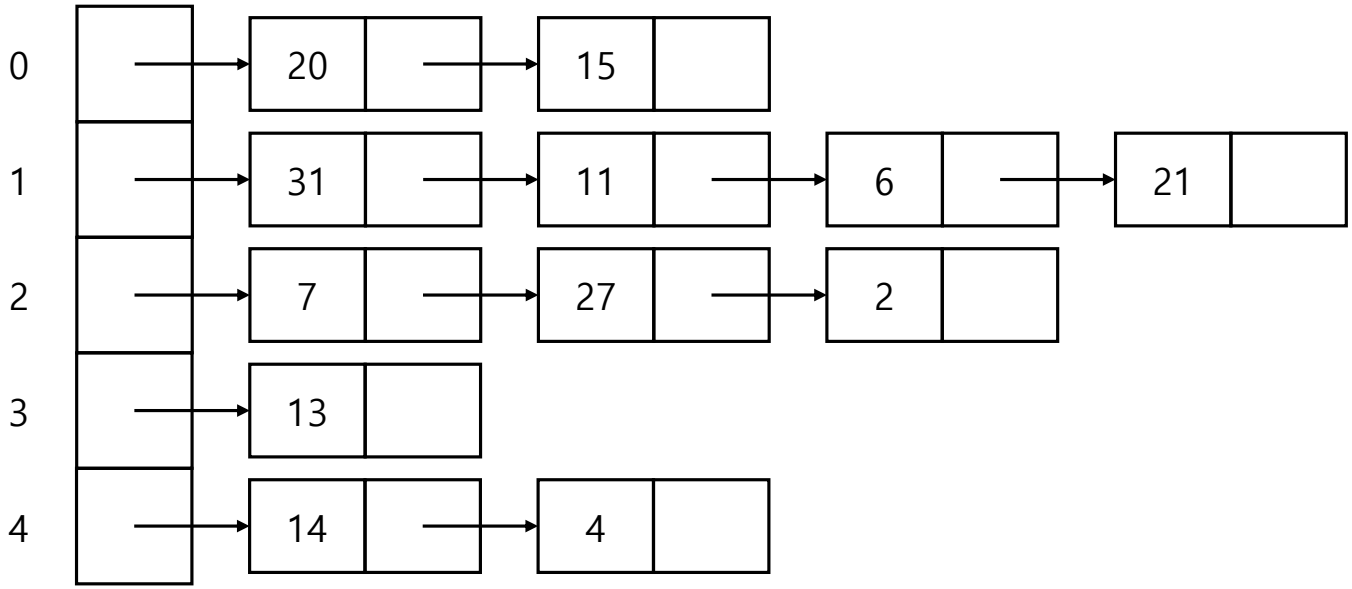
Algorithm 2022 Fall Problem Solving Solution

Report 2

1. Consider inserting the keys 31, 11, 20, 6, 21, 15, 7, 27, 14, 13, 4, 2 into a hash table of length m = 5 using separate chaining where h(k) = k mod m. Draw the resulting hash tables after inserting all these keys.



2. Consider inserting the keys 12, 9, 8, 17, 2, 10, 31, 1, 18 into a hash table of length **m =11** using open addressing with the auxiliary hash function h′(k)=k mod m. Draw the resulting hash tables after inserting all these keys.

a) using linear probing with h(k, i) = (h′(k) + i) mod m

h(12,0) = (h′(12)+0) mod 11 = (1+0) mod 11 = 1

h(9,0) = (h′(9)+0) mod 11 = (9+0) mod 11 = 9

h(8,0) = (h′(8)+0) mod 11 = (8+0) mod 11 = 8

h(17,0) = (h′(17)+0) mod 11 = (6+0) mod 11 = 6

h(2,0) = (h′(2)+0) mod 11 = (2+0) mod 11 = 2

h(10,0) = (h′(10)+0) mod 11 = (10+0) mod 11 = 10

h(31,0) = (h′(31)+0) mod 11 = (9+0) mod 11 = 9 $(collision)$

h(31,1) = (h′(31)+1) mod 11 = (9+1) mod 11 = 10 $(collision)$

h(31,2) = (h′(31)+2) mod 11 = (9+2) mod 11 = 0

h(1,0) = (h′(1)+0) mod 11 = (1+0) mod 11 = 1 $(collision)$

h(1,1) = (h′(1)+1) mod 11 = (1+1) mod 11 = 2 $(collision)$

h(1,2) = (h′(1)+2) mod 11 = (1+2) mod 11 = 3

h(18,0) = (h′(18)+0) mod 11 = (7+0) mod 11 = 7

0	31
1	12
2	2
3	1
4	
5	
6	17
7	18
8	8
9	9
10	10

2. Consider inserting the keys 12, 9, 8, 17, 2, 10, 31, 1, 18 into a hash table of length **m =11** using open addressing with the auxiliary hash function h′(k)=k mod m. Draw the resulting hash tables after inserting all these keys.

b) using quadratic probing with (k, i) = (h′(k) + c_1 + $c_2 i^2$) mod m, where c_1=1 and c_2=3

h(12,0) = (h'(12) + 1 + 0) mod 11) = (1 + 1 +0) mod 11 = 2

h(9,0) = (h'(9) + 1 + 0) mod 11) = (9 + 1 +0) mod 11 = 10

h(8,0) = (h'(8) + 1 + 0) mod 11) = (8 + 1 +0) mod 11 = 9

h(17,0) = (h'(17) + 1 + 0) mod 11) = (6 + 1 +0) mod 11 = 7

h(2,0) = (h'(2) + 1 + 0) mod 11) = (2 + 1 +0) mod 11 = 3

h(10,0) = (h'(10) + 1 + 0) mod 11) = (10 + 1 +0) mod 11 = 0

h(31,0) = (h'(31) + 1 + 0) mod 11) = (9 + 1 +0) mod 11 = 10 ($collision$)

h(31,1) = (h'(31) + 1 + 3) mod 11) = (9 + 1 +3) mod 11 = 2 ($collision$)

h(31,2) = (h'(31) + 1 + 12) mod 11) = (9 + 1 +12) mod 11 = 0 ($collision$)

h(31,3) = (h'(31) + 1 + 27) mod 11) = (9 + 1 +27) mod 11 = 4

h(1,0) = (h'(1) + 1 + 0) mod 11) = (1 + 1 +0) mod 11 = 2 ($collision$)

h(1,1) = (h'(1) + 1 + 3) mod 11) = (1 + 1 +3) mod 11 = 5

h(18,0) = (h'(18) + 1 + 0) mod 11) = (7 + 1 +0) mod 11 = 8

0	10
1	
2	12
3	2
4	31
5	1
6	
7	17
8	18
9	8
10	9

2. Consider inserting the keys 12, 9, 8, 17, 2, 10, 31, 1, 18 into a hash table of length **m =11** using open addressing with the auxiliary hash function h′(k)=k mod m. Draw the resulting hash tables after inserting all these keys.

c) using double probing with h(k, i) = (h′(k) + i*h_2(k)) mod m,

where h_2(k)=1+(k mod (m-1))

h(12,0)=(h′ (12)+0∗h′(12)) mod 11=(1+0) mod 11=1

h(9,0)=(h′ (9)+0∗h′(9)) mod 11=(9+0) mod 11=9

h(8,0)=(h′ (8)+0∗h′(8)) mod 11=(8+0) mod 11=8

h(17,0)=(h′ (17)+0∗h′(17)) mod 11=(6+0) mod 11=6

h(2,0)=(h′ (2)+0∗h′(2)) mod 11=(2+0) mod 11=2

h(10,0)=(h′ (10)+0∗h′(10)) mod 11=(10+0) mod 11=10

h(31,0)=(h′ (31)+0∗h′(31)) mod 11=(9+0) mod 11=9 $(collision)$

h(31,1)=(h′ (31)+1∗h′(31)) mod 11=(9+2) mod 11=0

h(1,0)=(h′ (1)+0∗h′(1)) mod 11=(1+0) mod 11=1 $(collision)$

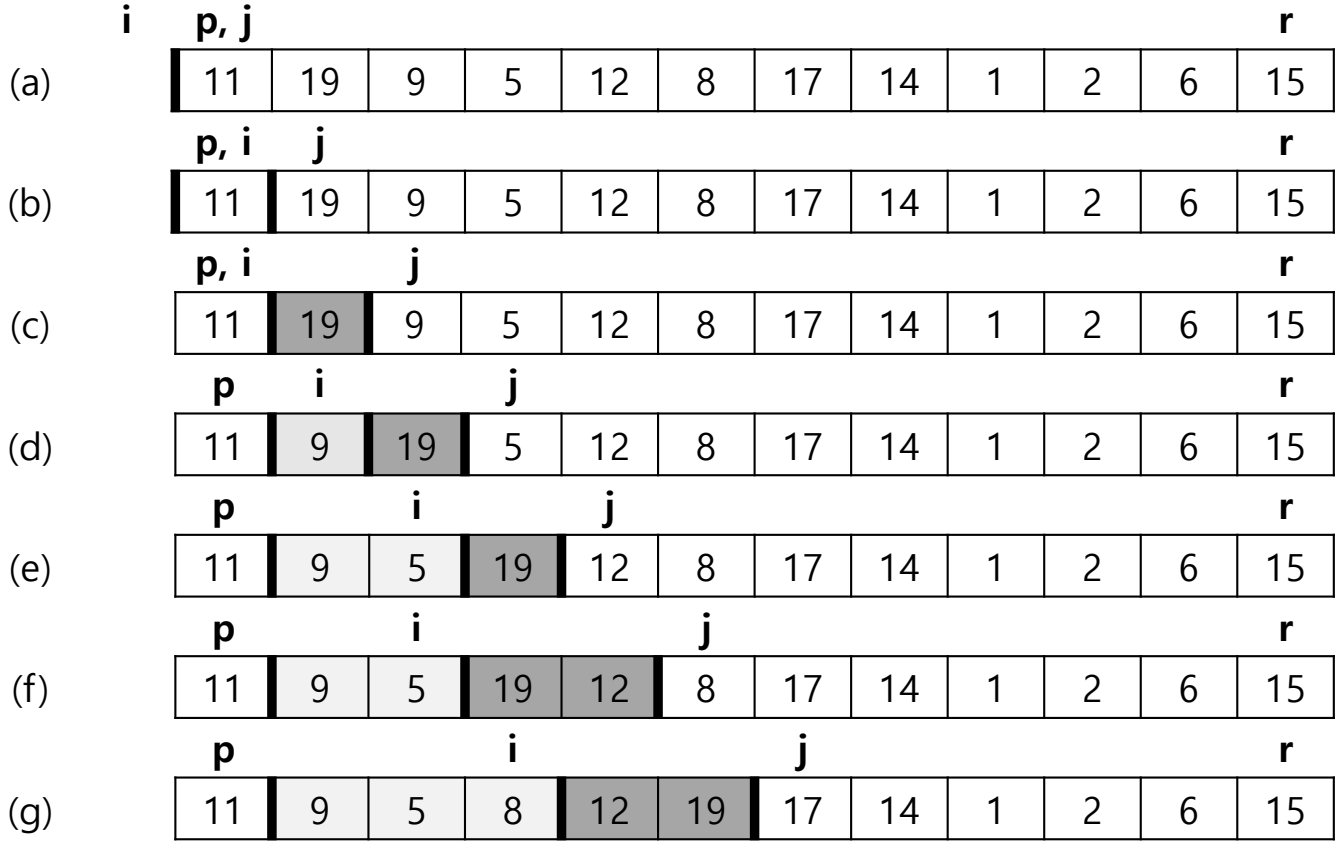
h(1,1)=(h′ (1)+1∗h′(1)) mod 11=(1+2) mod 11=3

h(18,0)=(h′ (18)+0∗h′(18)) mod 11=(7+0) mod 11=7

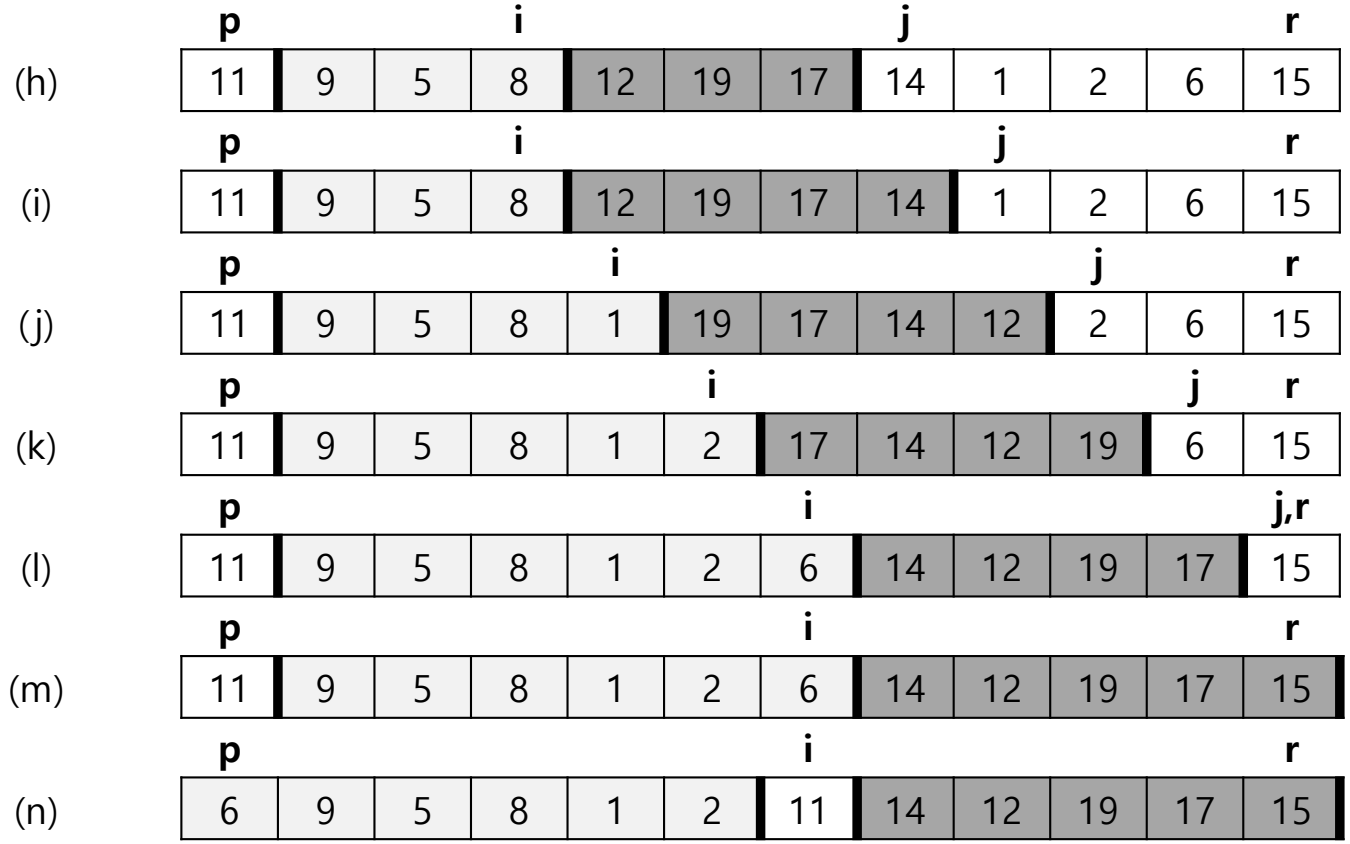
0	31
1	12
2	2
3	1
4	
5	
6	17
7	18
8	8
9	9
10	10

3. Using Figure 7.1 as a model, illustrate the operation of PARTITION in quicksort on the arrays shown below. The pivot is the first element in A

a) A = <11, 19, 9, 5, 12, 8, 17, 14, 1, 2, 6, 15>

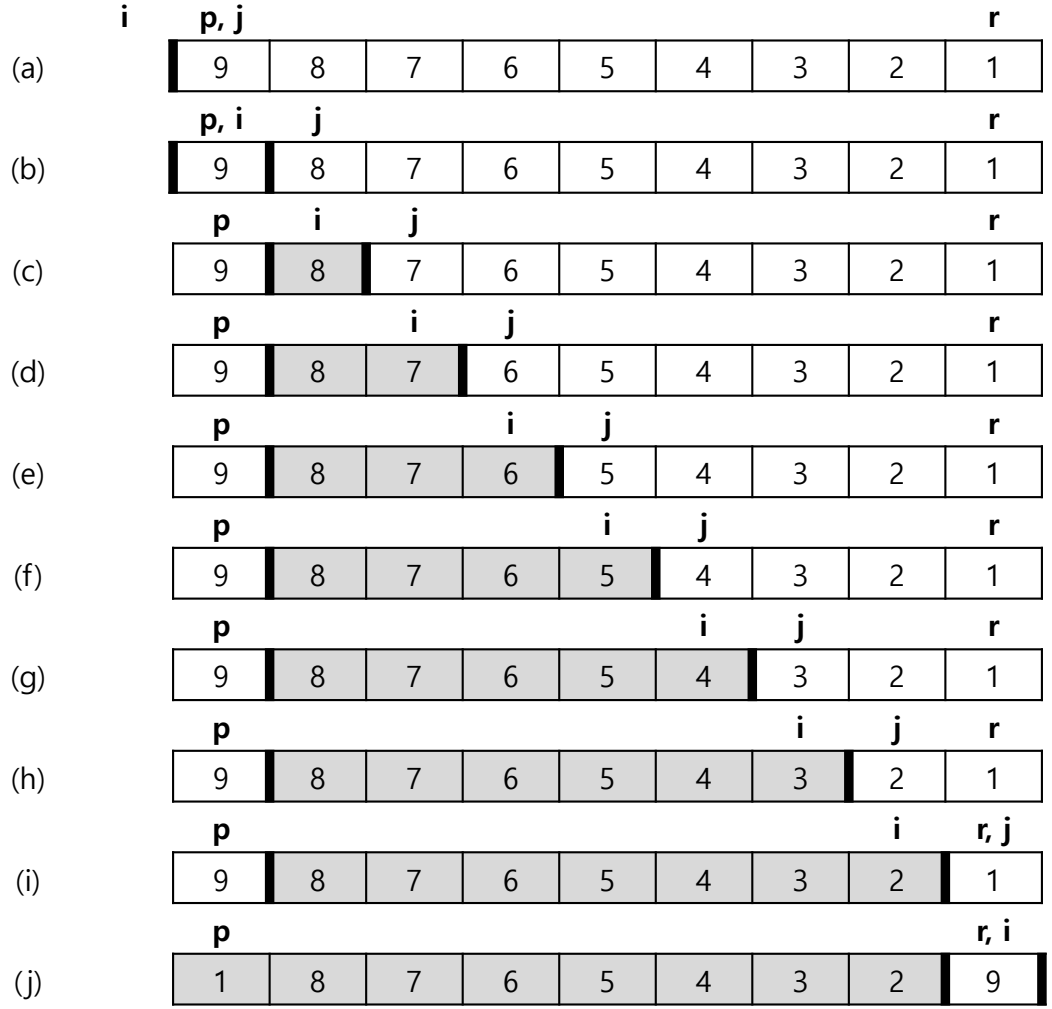


3. Using Figure 7.1 as a model, illustrate the operation of PARTITION in quicksort on the arrays shown below. The pivot is the first element in A



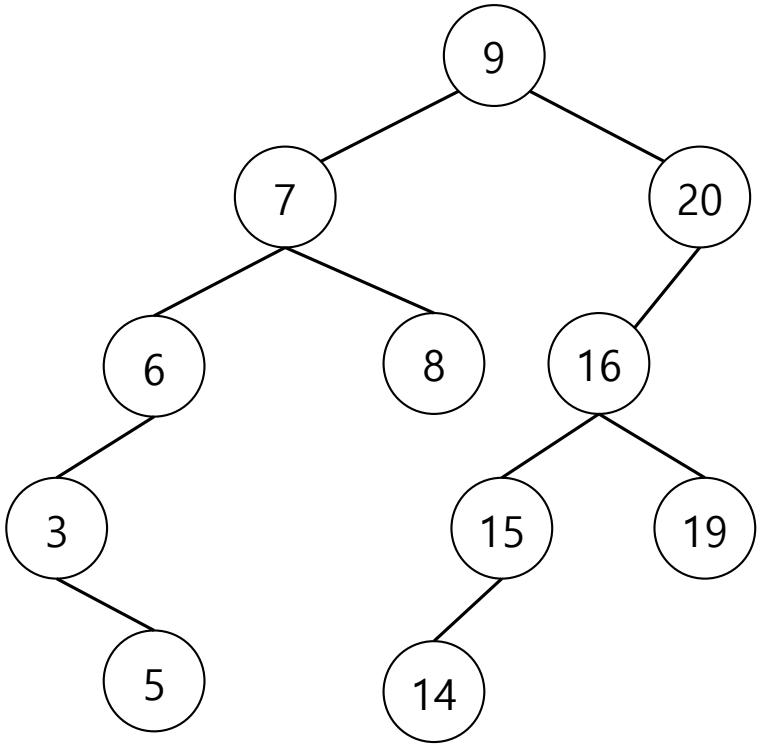
3. Using Figure 7.1 as a model, illustrate the operation of PARTITION in quicksort on the arrays shown below. The pivot is the first element in A

b) A = <9,8,7,6,5,4,3,2,1>



4. Answer the following questions for the keys: 9, 20, 7, 6, 16, 3, 19, 8, 15, 14, 5

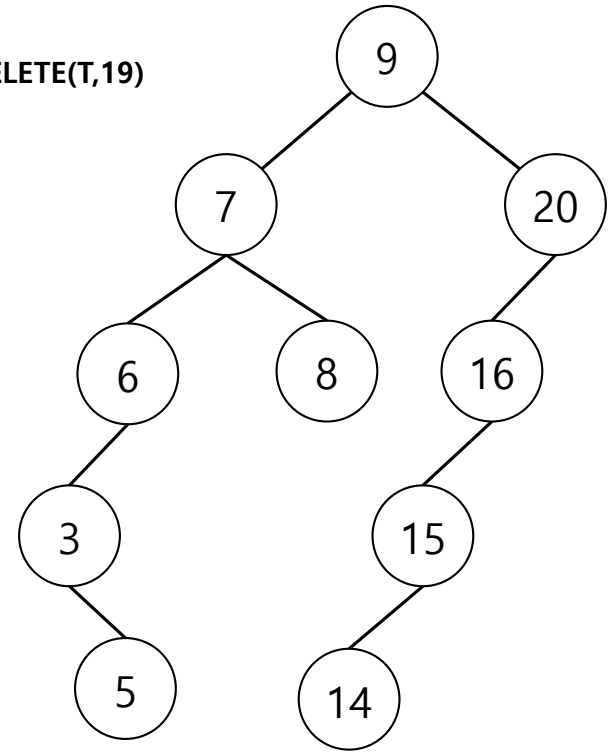
a) Draw the final structure of binary search tree T when above keys are inserted from left to right.



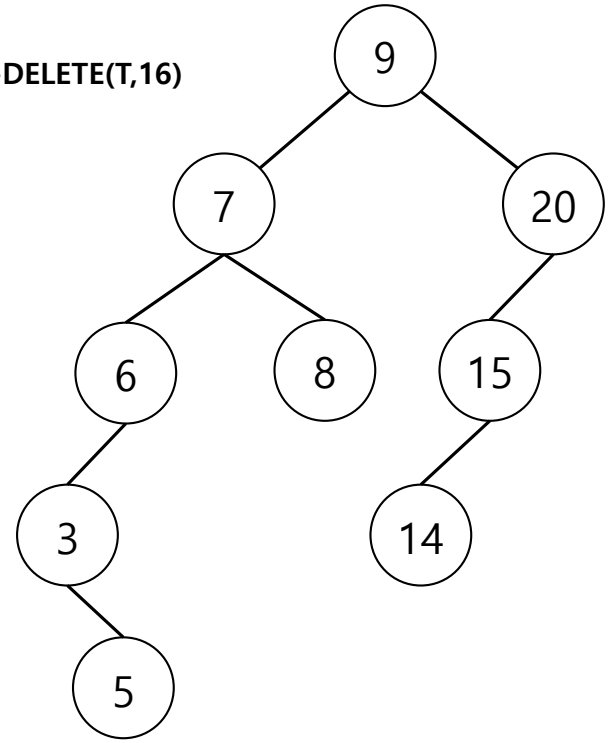
4. Answer the following questions for the keys: 9, 20, 7, 6, 16, 3, 19, 8, 15, 14, 5

b) Draw the tree that results after successively executing the following functions. TREE-DELETE(T,19), TREE-DELETE(T,16), TREE-DELETE(T,7), TREE-DELETE(T,9). Draw the tree each time TREE-DELETE is executed.

TREE-DELETE(T,19)



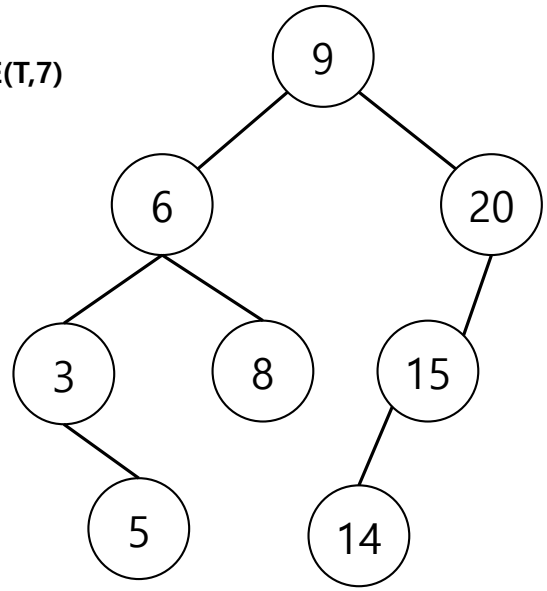
TREE-DELETE(T,16)



4. Answer the following questions for the keys: 9, 20, 7, 6, 16, 3, 19, 8, 15, 14, 5

b) Draw the tree that results after successively executing the following functions. TREE-DELETE(T,19), TREE-DELETE(T,16), TREE-DELETE(T,7), TREE-DELETE(T,9). Draw the tree each time TREE-DELETE is executed.

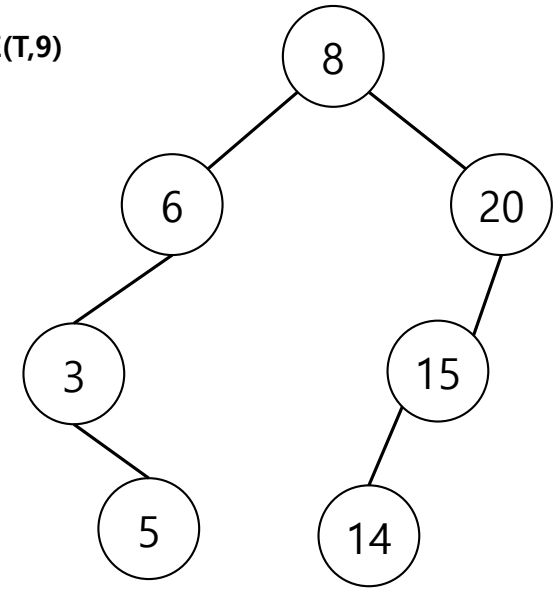
TREE-DELETE(T,7)



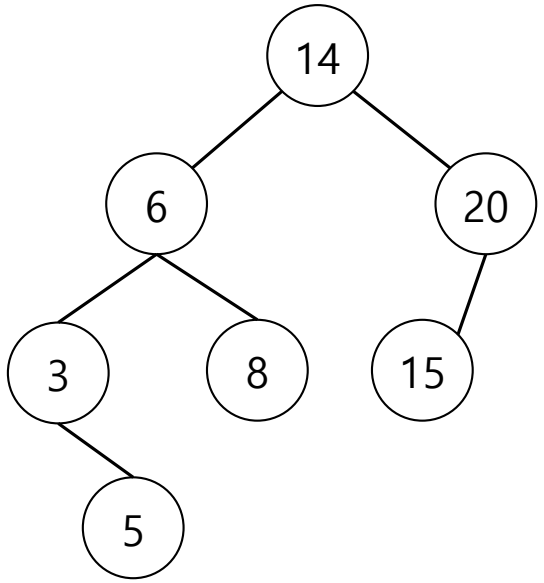
4. Answer the following questions for the keys: 9, 20, 7, 6, 16, 3, 19, 8, 15, 14, 5

b) Draw the tree that results after successively executing the following functions. TREE-DELETE(T,19), TREE-DELETE(T,16), TREE-DELETE(T,7), TREE-DELETE(T,9). Draw the tree each time TREE-DELETE is executed.

TREE-DELETE(T,9)



OR



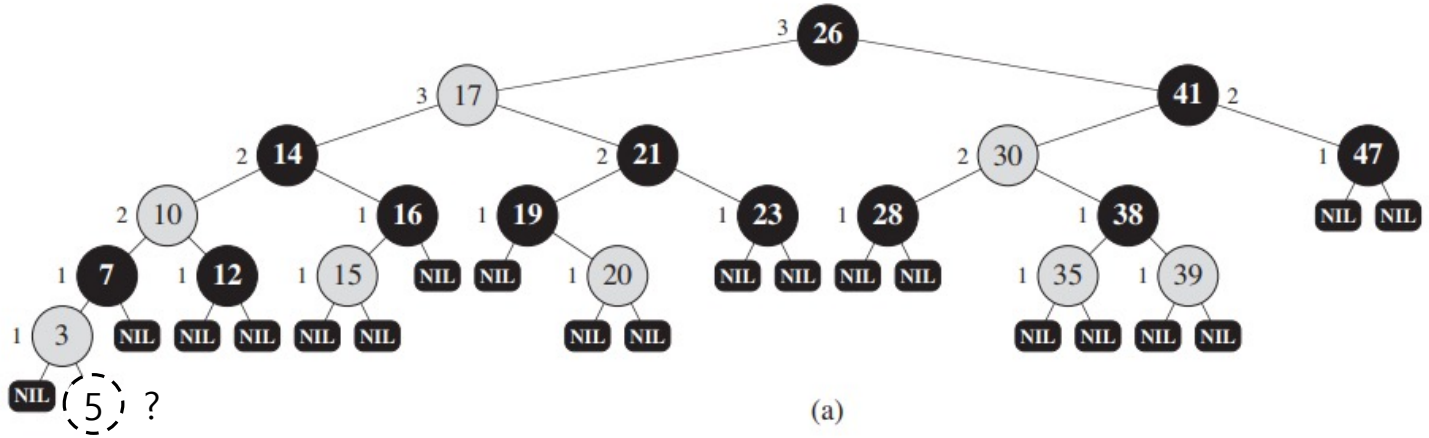
5. Write pseudo code for TREE-PREDECESSOR procedure

TREE-PREDECESSOR (x)

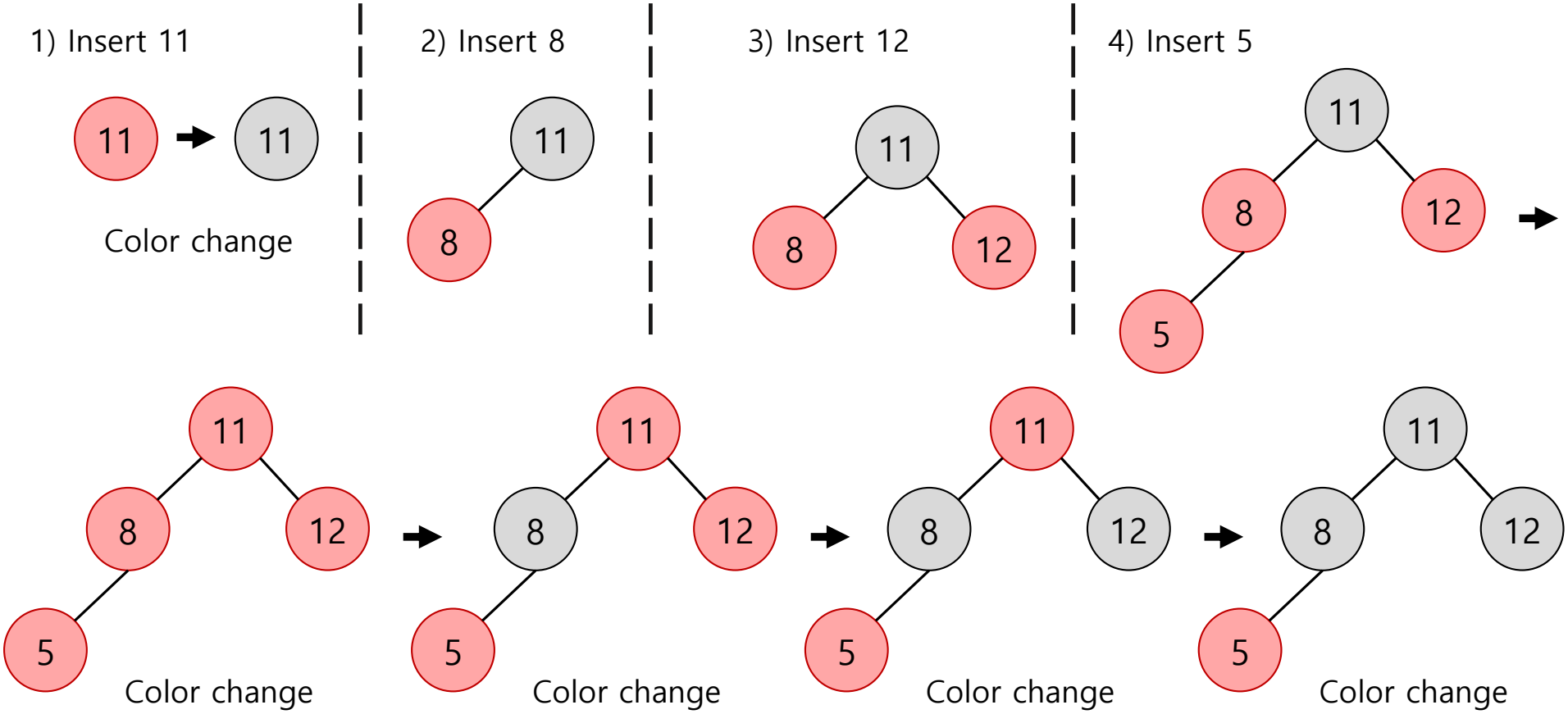
```
1.  if x.left ≠ NIL
2.      return TREE_MAXIMUM( x.left )
3.  y = x.parent
4.  while y ≠ NIL and x == y.left
5.      x = y
6.      y = y.parent
7.  return y
```

6. Draw the red-black tree that results after TREE-INSERT is called on the tree in Figure 13.1(c) with key 5. If the inserted node is colored red, is the resulting tree a red-black tree? What if it is colored black? Answer without TREE-INSERT-FIXUP execution.

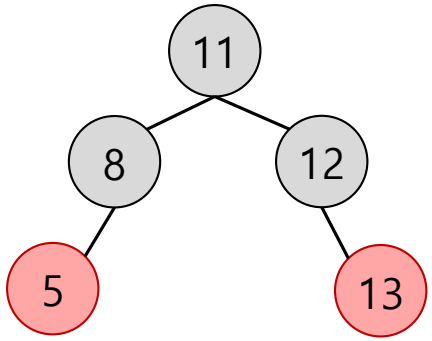
- Node 5 is **red**: double red problem occurs (violate property 4)
- Node 5 is **black**: The number of black nodes on a path from the root to a leaf does not match (violate property 5)



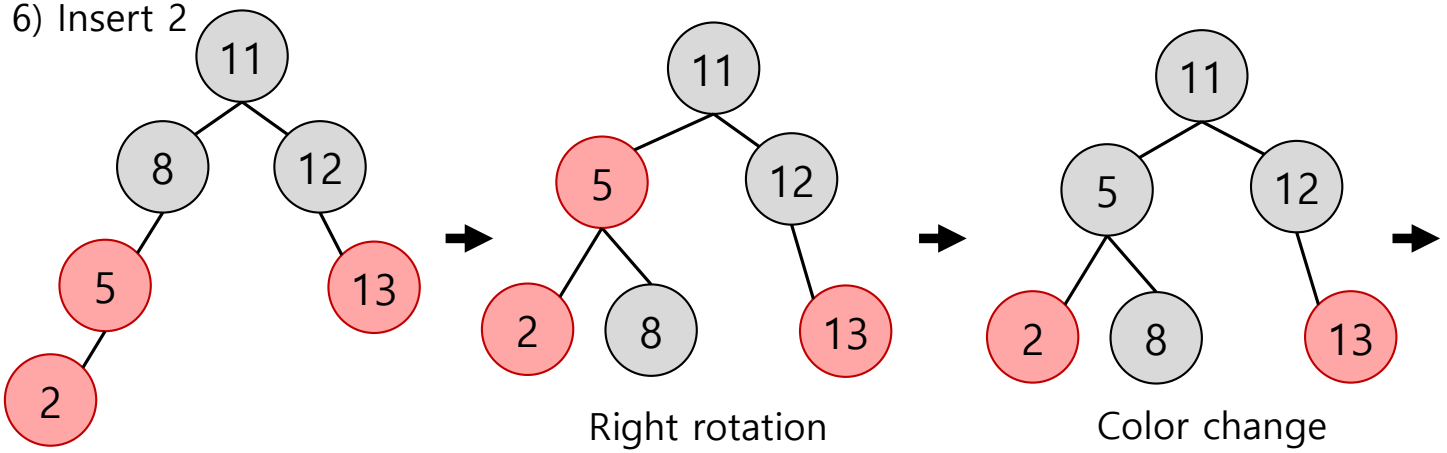
7. Draw the red-black trees that result from successive insertions of the keys in the order 11, 8, 12, 5, 13, 2, 10, 17, 26, 9 into an initially empty red-black tree. For each insertion, count the number of color changes, left rotations, and right rotations. Also, count the sum of each of these three operations, respectively.



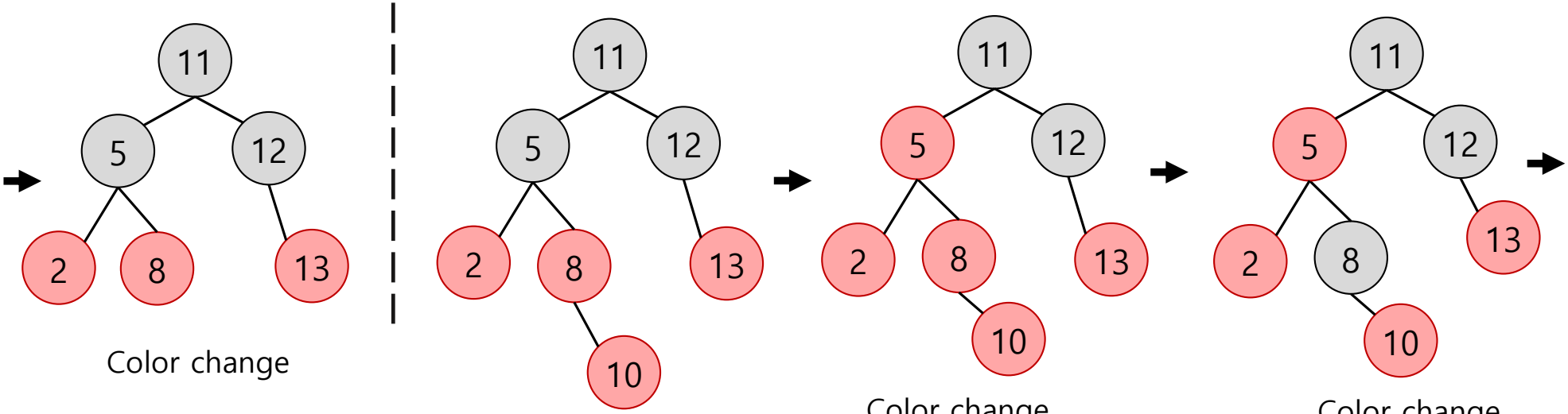
5) Insert 13

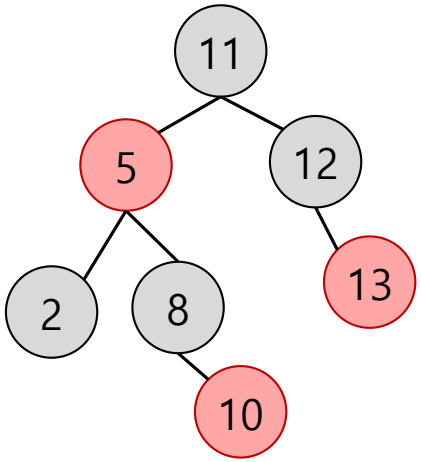


6) Insert 2



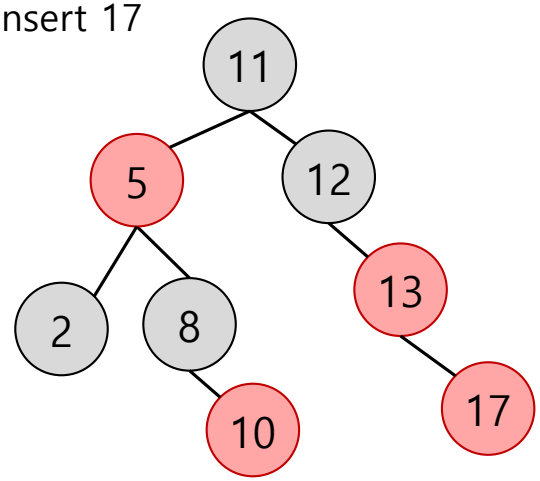
7) Insert 10

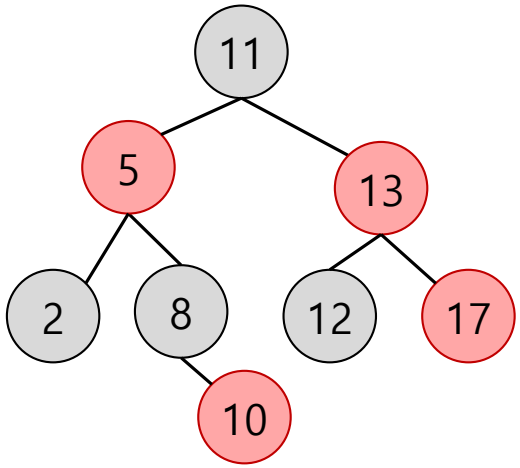




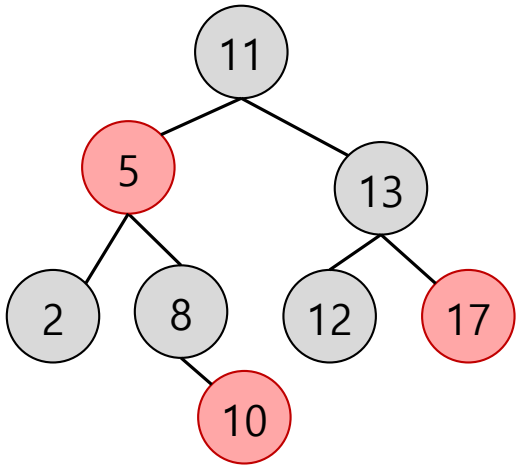
Color change

8) Insert 17

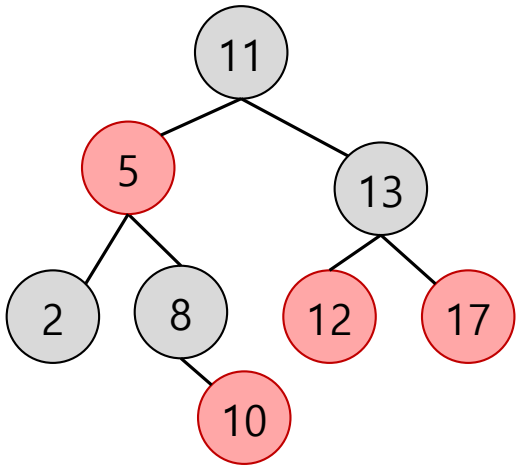




Left rotation

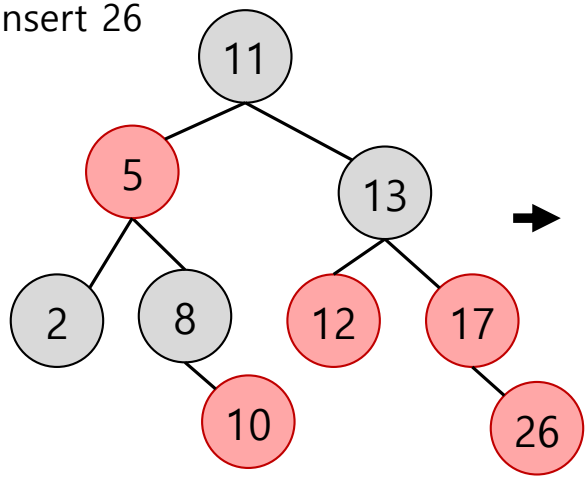


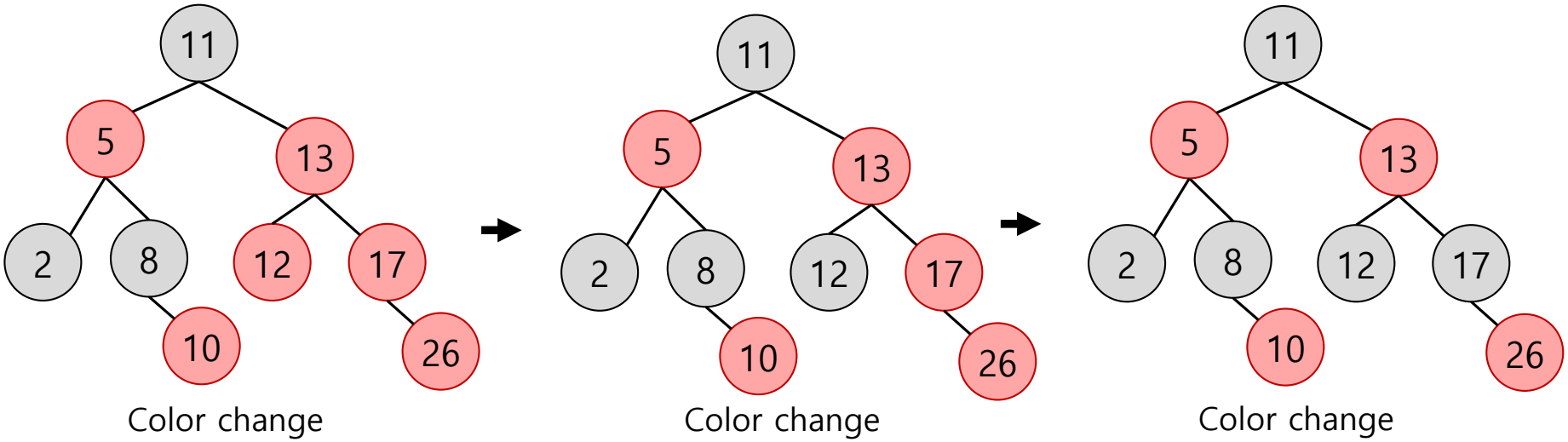
Color change



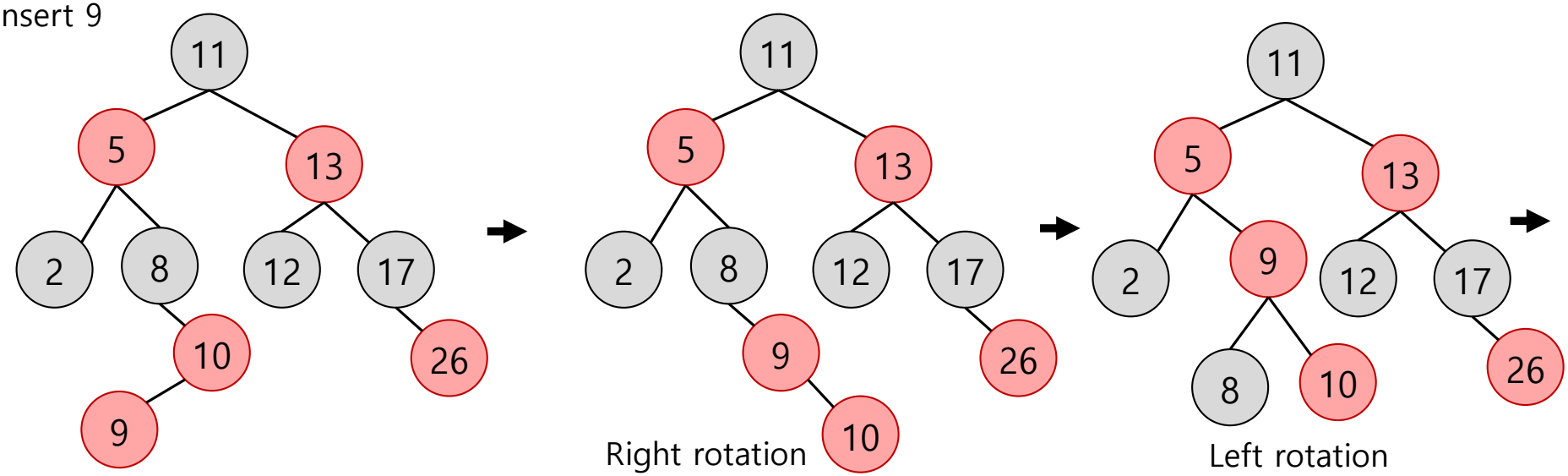
Color change

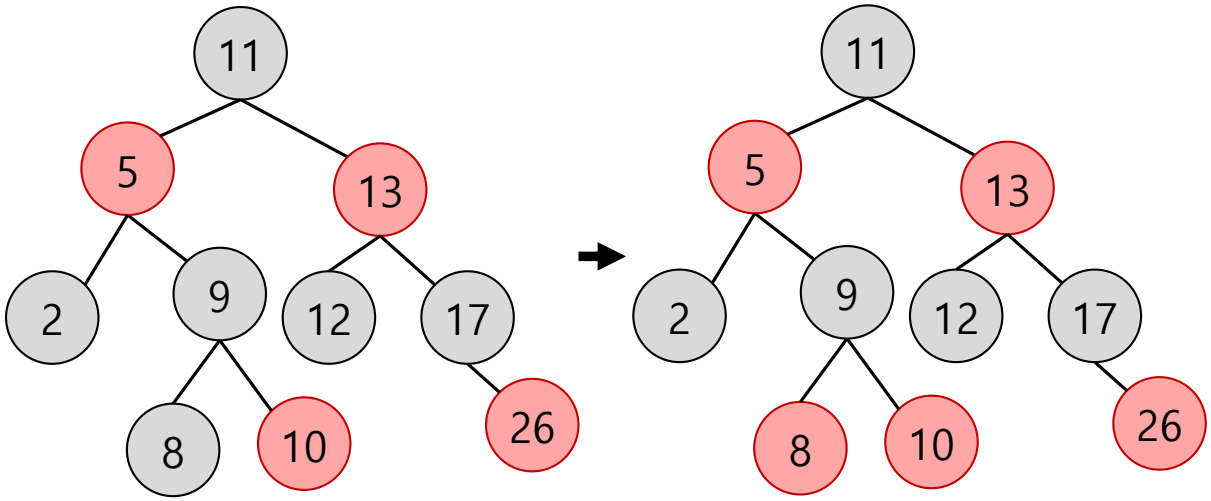
9) Insert 26




11
5
13
2
8
12
17
10
26
Color change
11
5
13
2
8
12
17
10
26
Color change
11
5
13
2
8
12
17
10
26
Color change

10) Insert 9

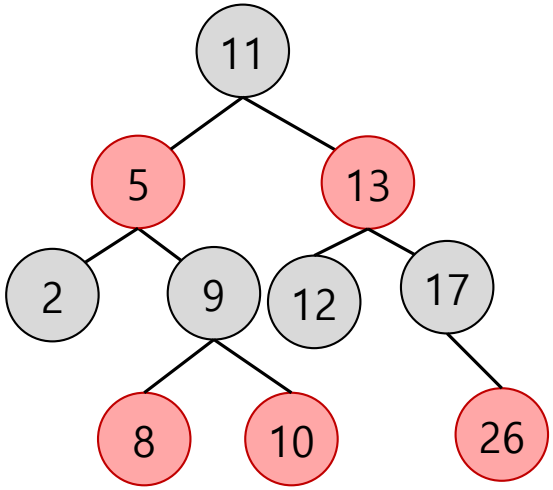

11
5
13
2
8
12
17
10
26
9
11
5
13
2
8
12
17
9
26
10
Right rotation
11
5
13
2
9
12
17
8
10
26
Left rotation



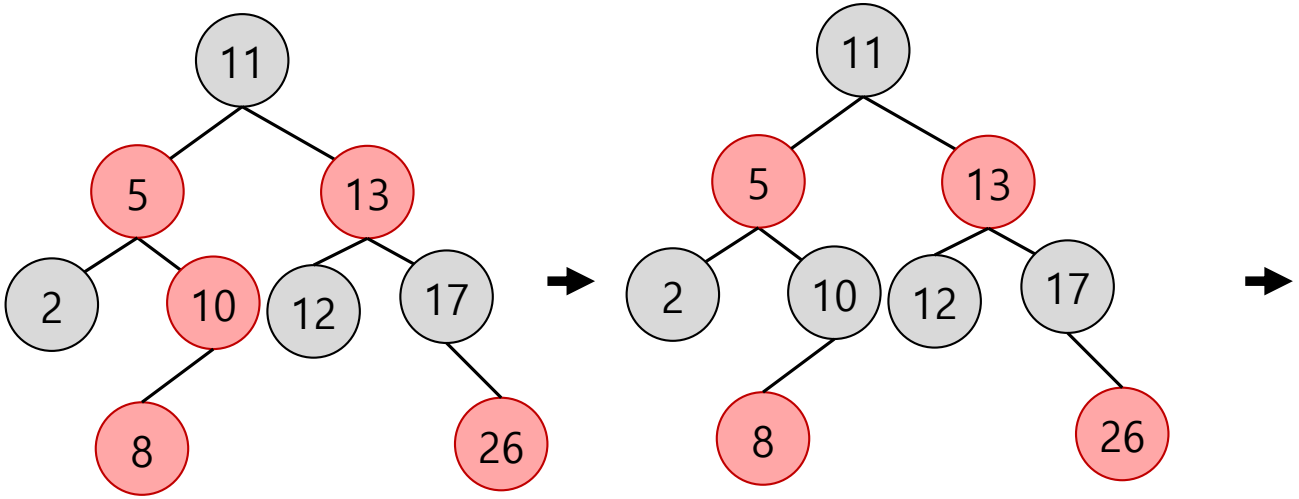
Color change: 17
Right rotation: 2
Left rotation: 2

8. Draw the red-black trees that result from the successive deletions of the keys in the order 9, 12, 10, 8, 11, 5, 17 on the tree generated in exercise 7. For each deletion, count the number of color changes, left rotations, and right rotations. Also, count the sum of each of these three operations, respectively.

0) initial

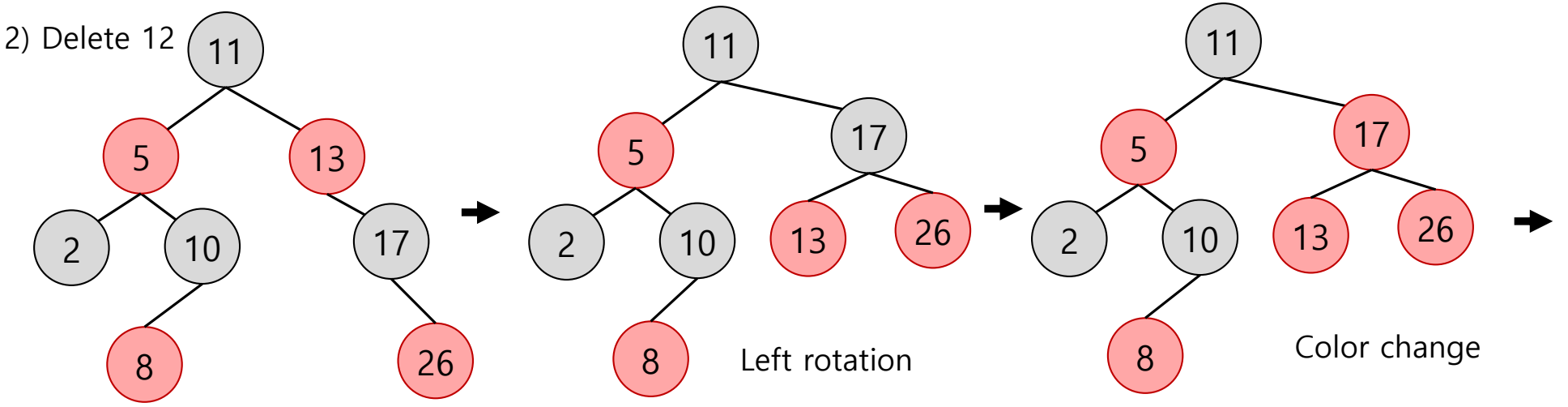


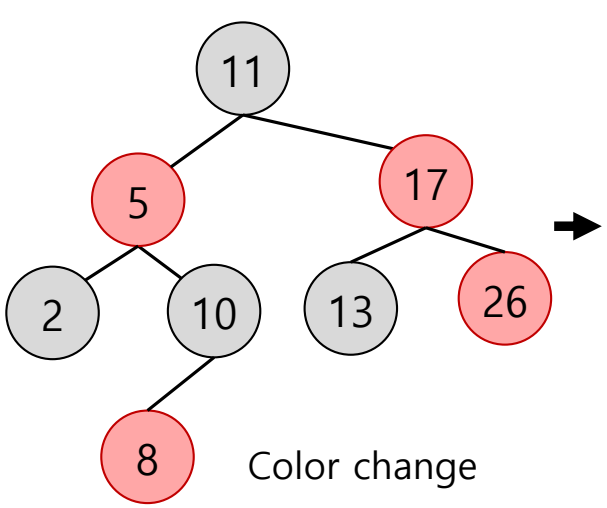
1) Delete 9

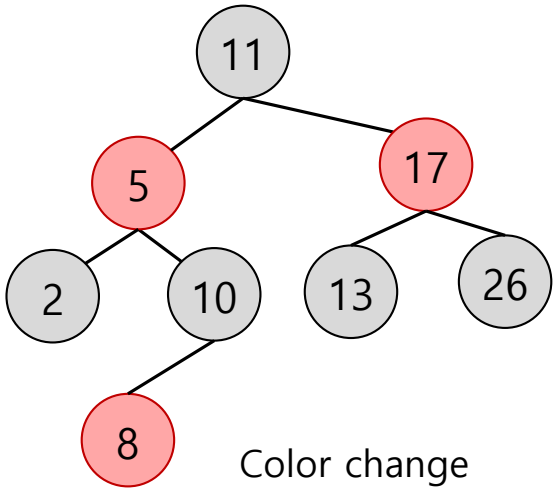


Color change

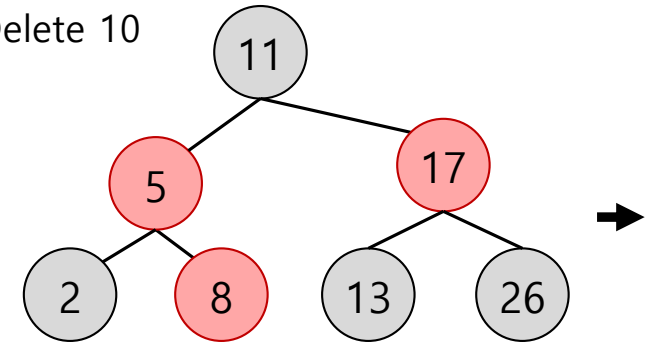
2) Delete 12

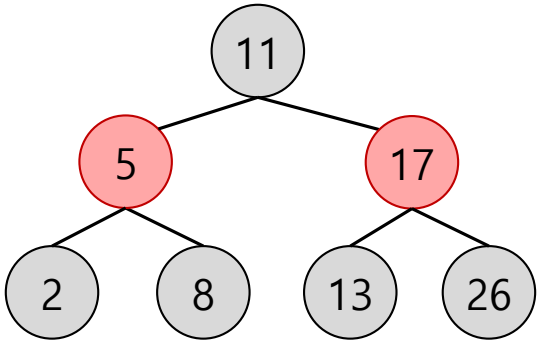






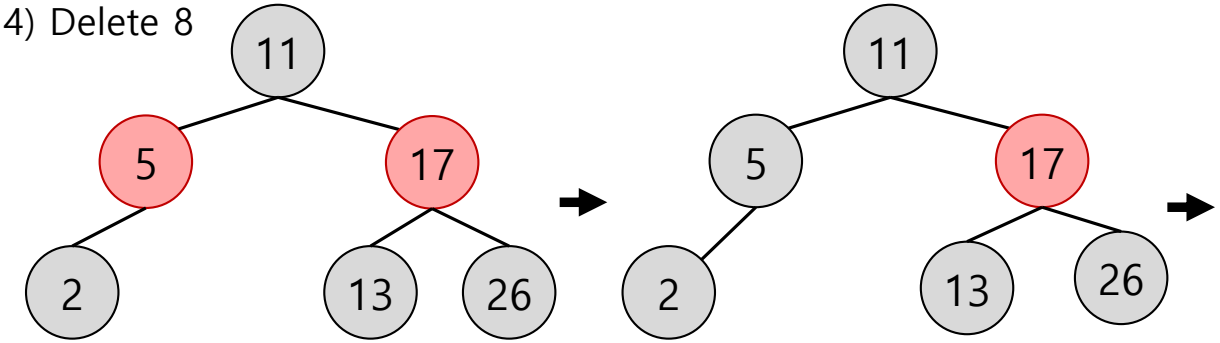
3) Delete 10



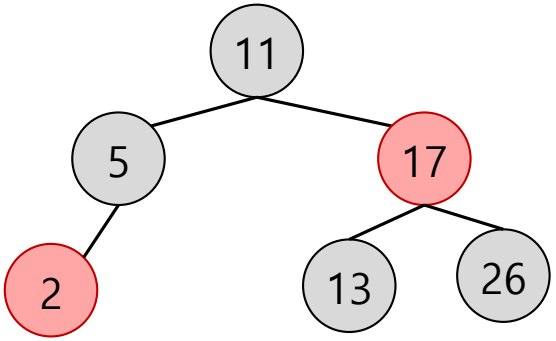


Color change

4) Delete 8

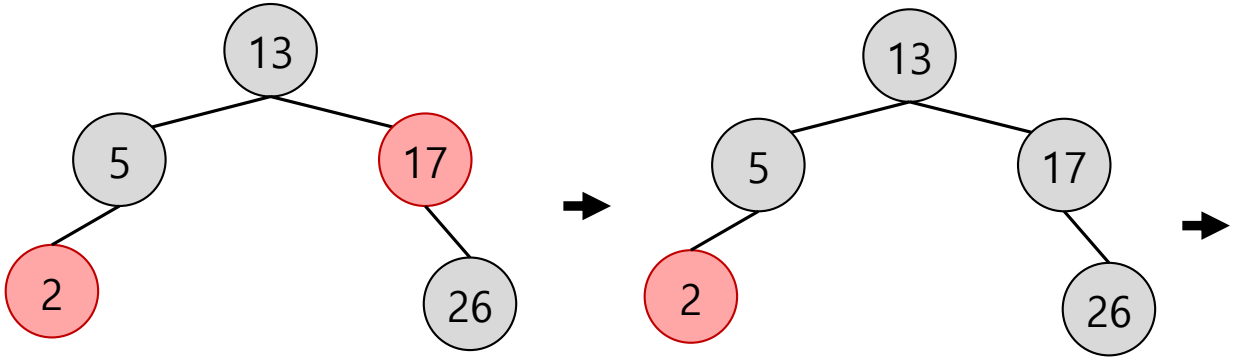


Color change

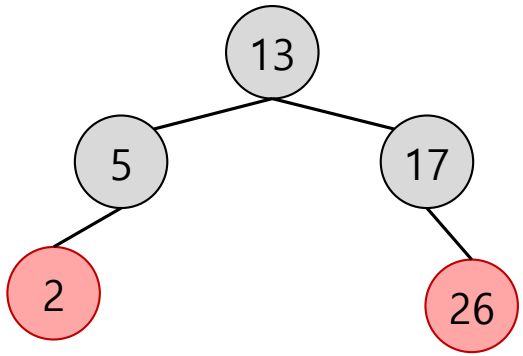


Color change

5) Delete 11

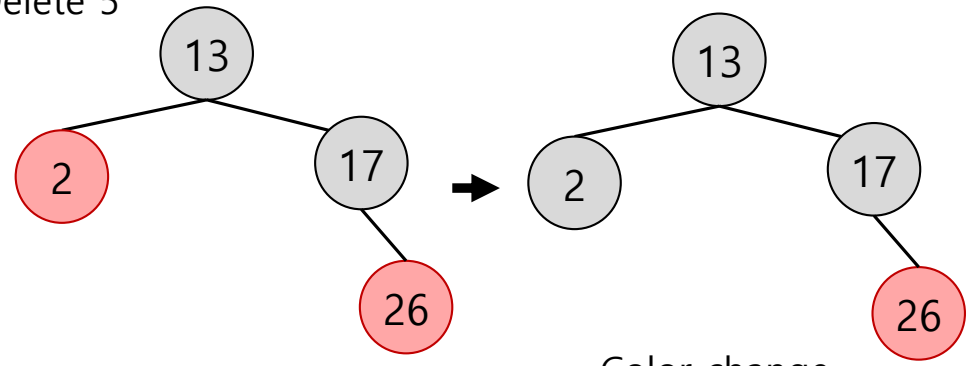


Color change



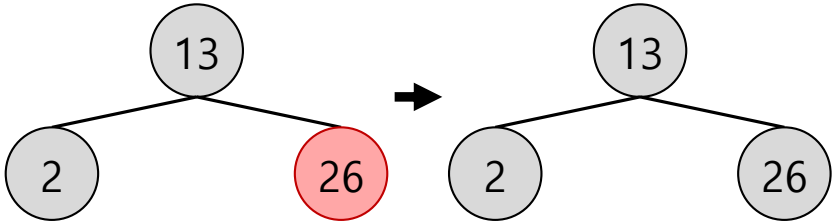
Color change

6) Delete 5



Color change

7) Delete 17



Color change

Color change: 11
Right rotation: 0
Left rotation: 1

9. Suppose that a node x is inserted into a red-black tree with RB-INSERT and then it is immediately deleted with RB-DELETE. Is the resulting red-black tree the same as the initial red-black tree? Justify your answer.

- No, the red-black tree will not necessarily be the same. In RB-tree, when a node is inserted, color change or rotation occurs depending on the tree shape and the tree is changed, so the tree shape from which the inserted node is deleted is different from the initial tree shape.