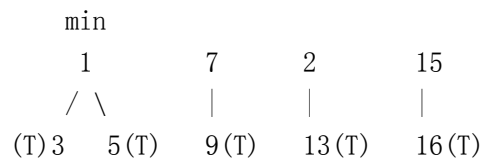
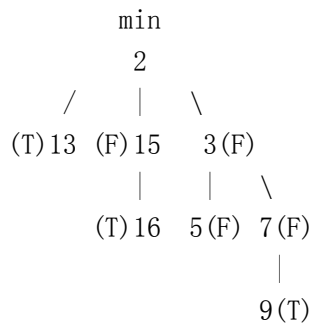
1. F-heap (ChildCut: T(true), F(false))

(a)



(b)

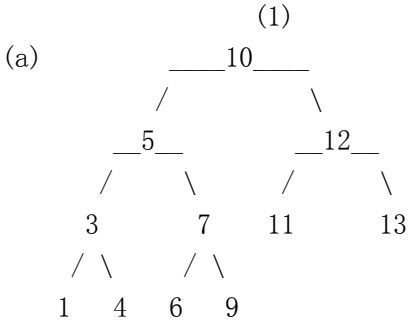


2.

```
Find-kth(ibst:tree, k:integer)
{
    if (ibst == nulll) error
    else if (ibst.leftSize == k-1) return (ibst)
    else if (ibst.leftSize < k-1)
        return (Find-kth(ibst.rightchild, k-ibstSize))
    else return (Find-kth(ibst.leftchild,k))
}
```

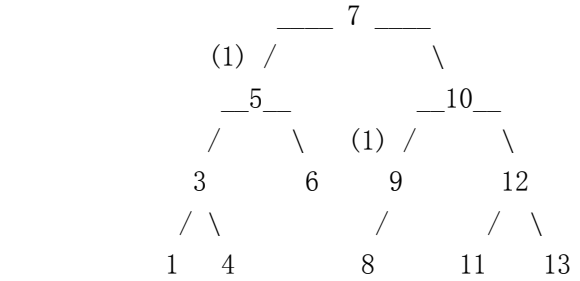
3. AVL tree (balance factor 0 is not shown).

(a)

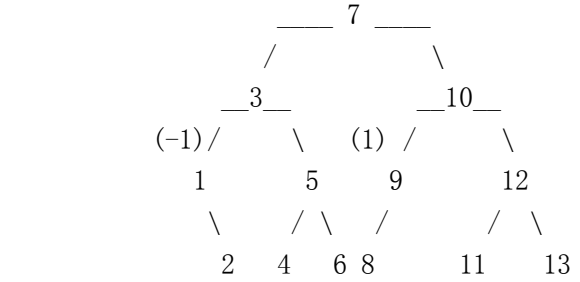


(b)

o insert 8 : LR rotation

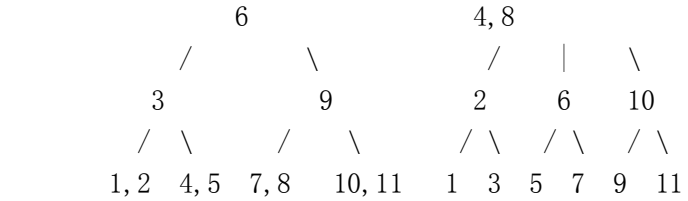


o insert 2 : LL rotation

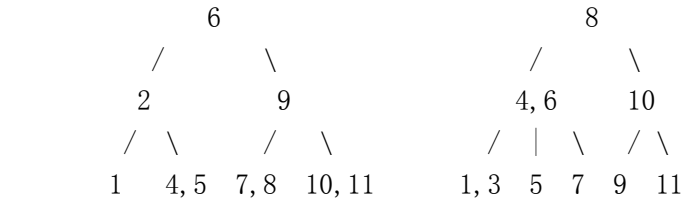


4.

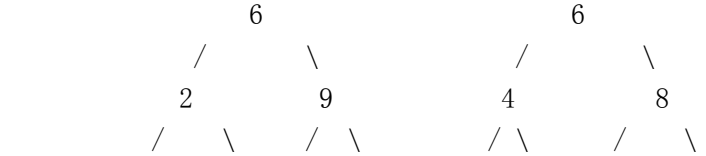
Original Tree (two version)



After first deletion:



After second deletion:

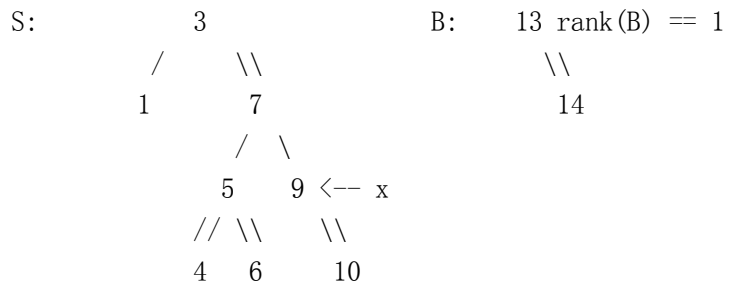


1 4,5 7,8 10 1,3 5 7 9,10

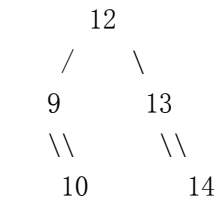
5. red-black tree

(a) Join(S,10,B)

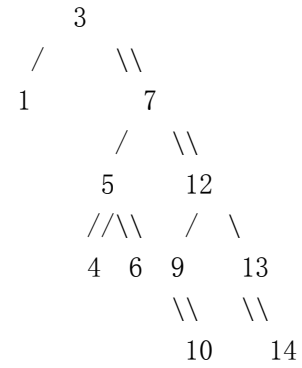
i) follow the right-child pointer until rank(B) == rank(x), where rank(B) == 1, x is a node pointer of tree S.



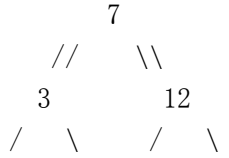
ii) combine subtree x, 10, and tree B

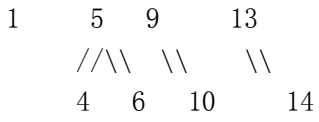


iii) connect the combined tree to node 5 through red pointer.



iv) perform RR rotation to remove consecutive red pointers.





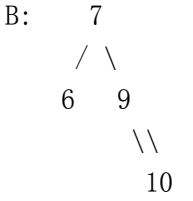
(b) Split(5,S,x,B) for tree S.

i) find node 5 and copy node value to x.
Since node 5 has no child,
tree S is NULL (left subtree) and
tree B is NULL (right subtree).

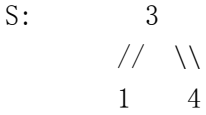
ii) perform Join(S, 4, NULL).
perform Join(B, 6, NULL).

S: 4
B: 6

iii) Join(B, 7, right-subtree of node 7).



iv) Join(left-subtree of node 3, 3, S).



So, the result of split operation: