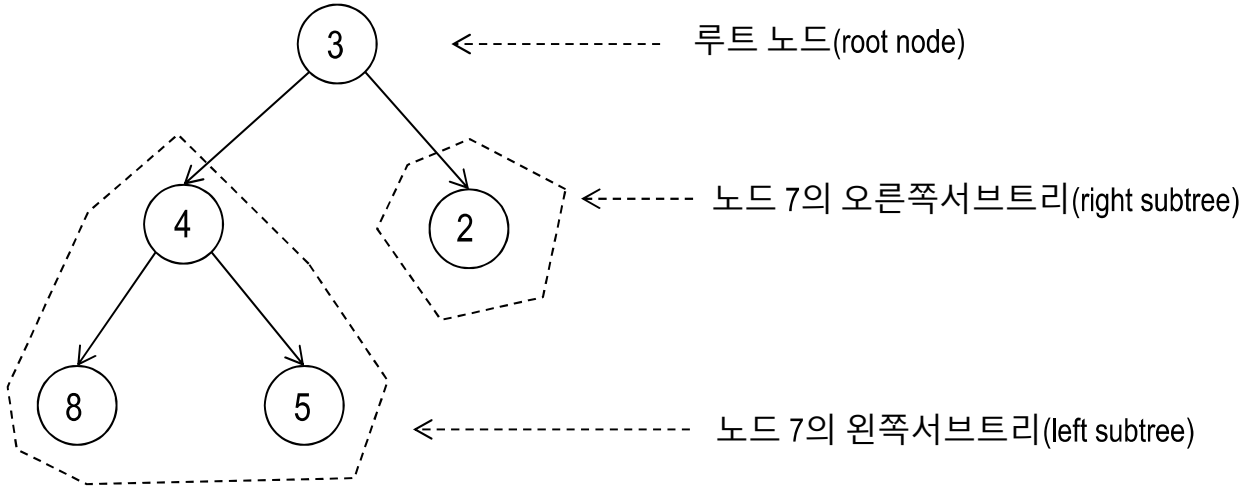
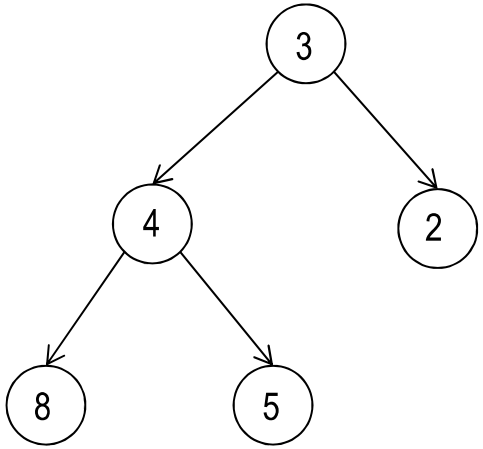
균형탐색트리

이진트리 (binary tree)

이진트리는 0개 이상 노드(node)들의 모음이며,
각 노드는 다른 노드로의 참조를 0개 이상 최대 2개까지 가지며,
모든 참조는 유일하고(사이클이 없음),
루트로의 참조는 없다.



이진트리 (binary tree): 링크 기반 표현법



- 중위순회(in-order traversal)

1. 현재 노드의 왼쪽 서브트리 중위순회
2. 현재 노드 방문
3. 현재 노드의 오른쪽 서브트리 중위순회

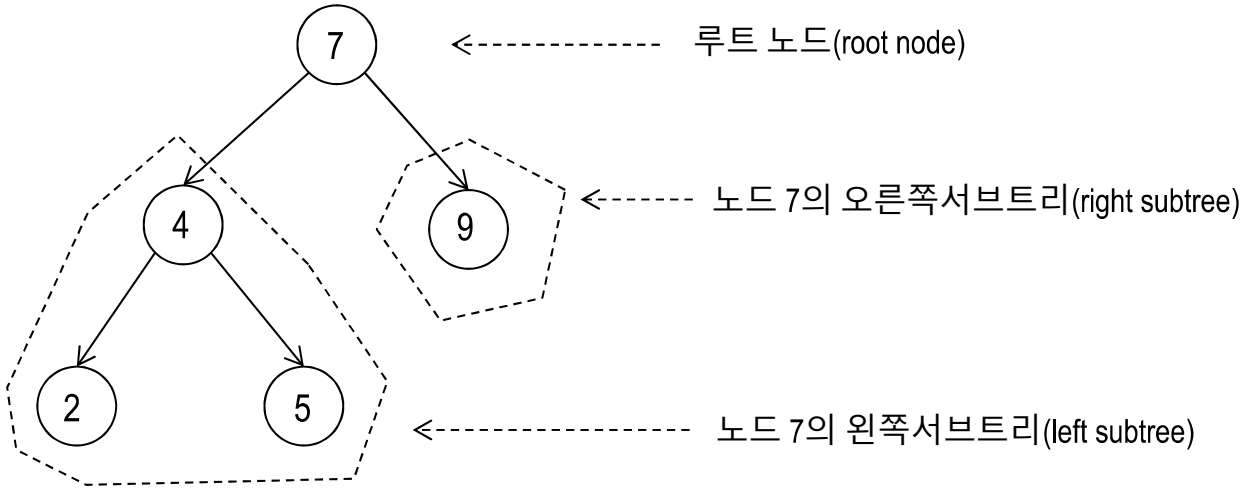
```
class TreeNode {
        int    key;
        TreeNode        left, right;
        public TreeNode(int key) {  this.key=key;  }
}
public class Test {
        public static void main(String[] args) {
                TreeNode        root;

                root=new TreeNode(3);
                root.left=new TreeNode(4);
                root.right=new TreeNode(2);
                root.left.left=new TreeNode(8);
                root.left.right=new TreeNode(5);

                inorder(root);
        }
        private static void inorder(TreeNode node) {
                if(node==null) return;
                inorder(node.left);
                System.out.print(node.key+" ");
                inorder(node.right);
        }
}
```

이진탐색트리 (binary search tree)

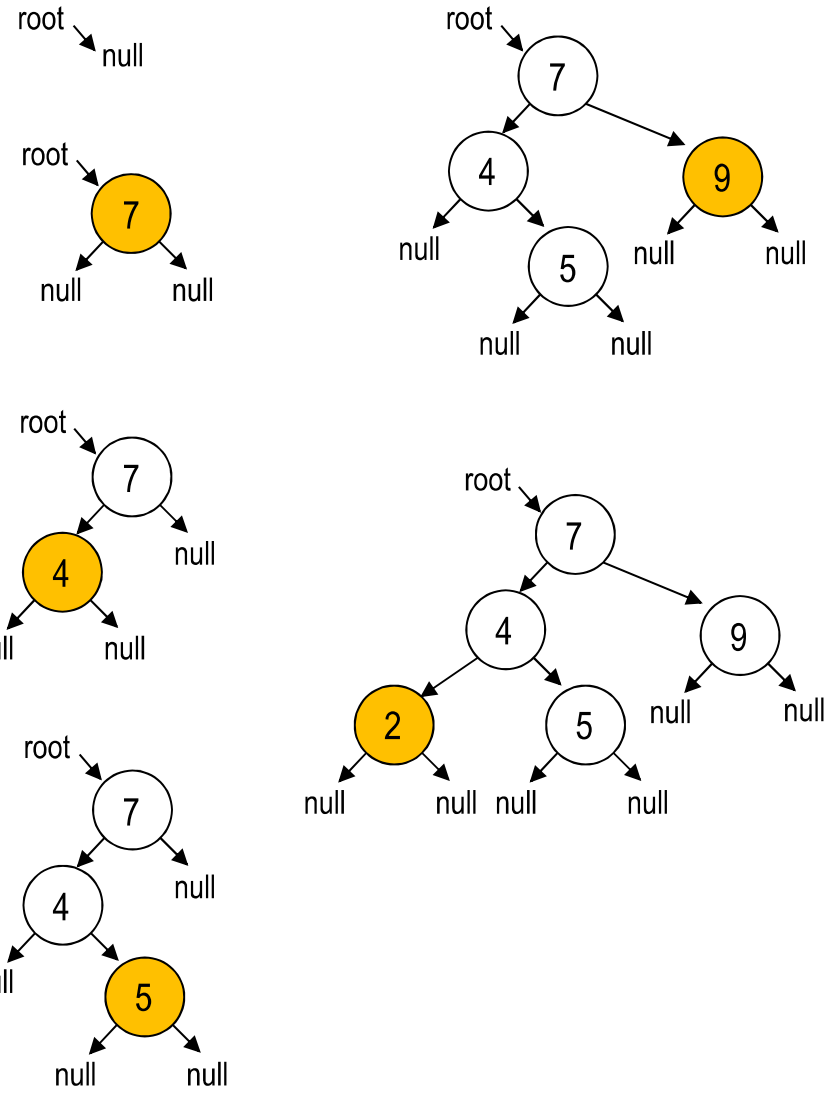
- 이진탐색트리는 이진트리이며,
- 이진탐색트리 내 각 노드의 키(key) 값은 그 노드의 왼쪽서브트리 내 모든 노드의 키 값보다 크고, 오른쪽서브트리 내 모든 노드의 키 값보다 작다.
- 이진탐색트리에 대한 중위순회는 이진탐색트리 내 모든 키 값들에 대한 오름차순 순회 생성



- 2, 4, 5 < 7 < 9
- 2 < 4 < 5
- ь ㅇ ι Æ : 2 4 5 7 9

이진탐색트리 (binary search tree): 삽입

Reference: https://algs4.cs.princeton.edu/32bst/BST.java.html, GPLv3



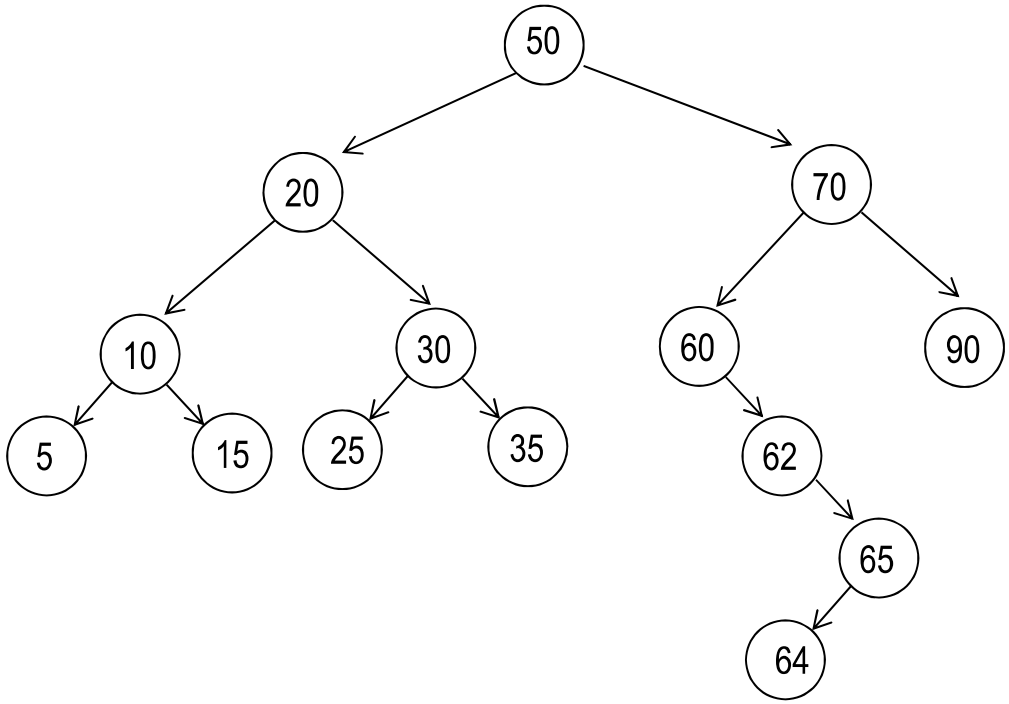

```
class TreeNode {
        int     key;
        TreeNode  left, right;
        public TreeNode(int key) {  this.key=key;  }
}

class BinarySearchTree {
        TreeNode  root;
        public void add(int key) {   root=add(root, key);  }
        private TreeNode add(TreeNode node, int key) {
                if(node==null) return new TreeNode(key);
                if(node.key<key) node.right=add(node.right, key);
                else if(node.key>key) node.left=add(node.left, key);
                return node;
        }
}

public class Test {
        public static void main(String[] args) {
                BinarySearchTree      tree=new BinarySearchTree();
                int     n[]={7, 4, 5, 9, 2};
                for (int i = 0; i < n.length; i++) tree.add(n[i]);
        }
}
```

이진탐색트리

이진탐색트리 삽입 순서: 50, 20, 70, 10, 30, 5, 15, 25, 60, 90, 62, 65, 64, 35

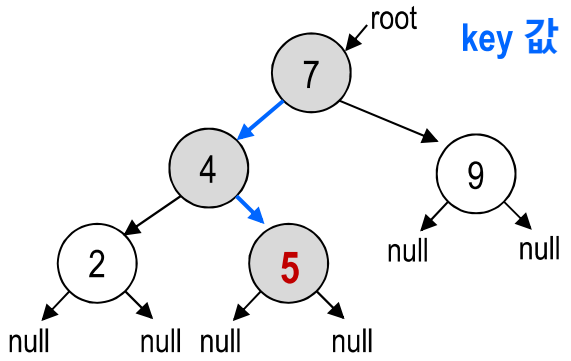


이진탐색트리의 inorder 순회(키 정렬 목록 생성, O(n)):
5 10 15 20 25 30 35 50 60 62 64 65 70 90

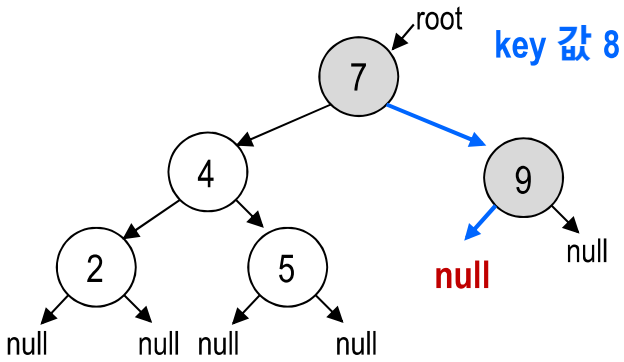
이진탐색트리 (binary search tree): 탐색


```
class TreeNode {
        int     key;
        TreeNode  left, right;
        public TreeNode(int key) {  this.key=key;  }
}

class BinarySearchTree {
        TreeNode  root;
        public void add(int key) {  root=add(root, key);  }
        private TreeNode add(TreeNode node, int key) {
                if(node==null) return new TreeNode(key);
                if(node.key<key) node.right=add(node.right, key);
                else if(node.key>key) node.left=add(node.left, key);
                return node;
        }
        public TreeNode search(int key) {
                TreeNode node=root;
                while(node!=null){
                        if(node.key==key) return node;
                        if(node.key<key) node=node.right;
                        else node=node.left;
                }
                return node;
        }
}
```



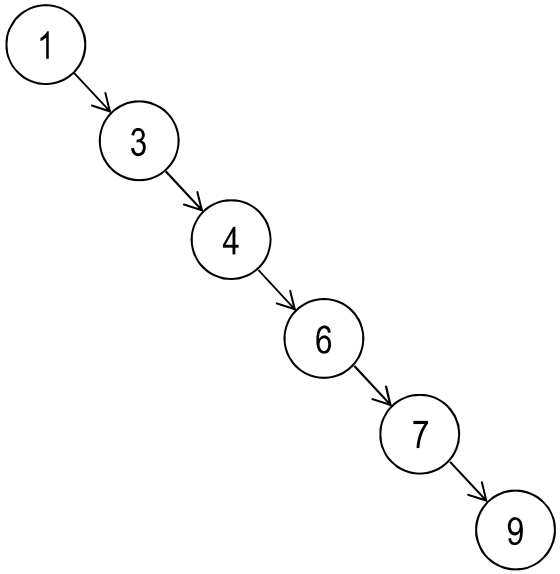

```
public class Test {
        public static void main(String[] args) {
                BinarySearchTree       tree=new BinarySearchTree();
                int     n[]={7,4,5,9,2};
                for (int i = 0; i < n.length; i++) tree.add(n[i]);
                System.out.println(tree.search(5));
                System.out.println(tree.search(8));
        }
}
```



불균형 이진탐색트리의 문제

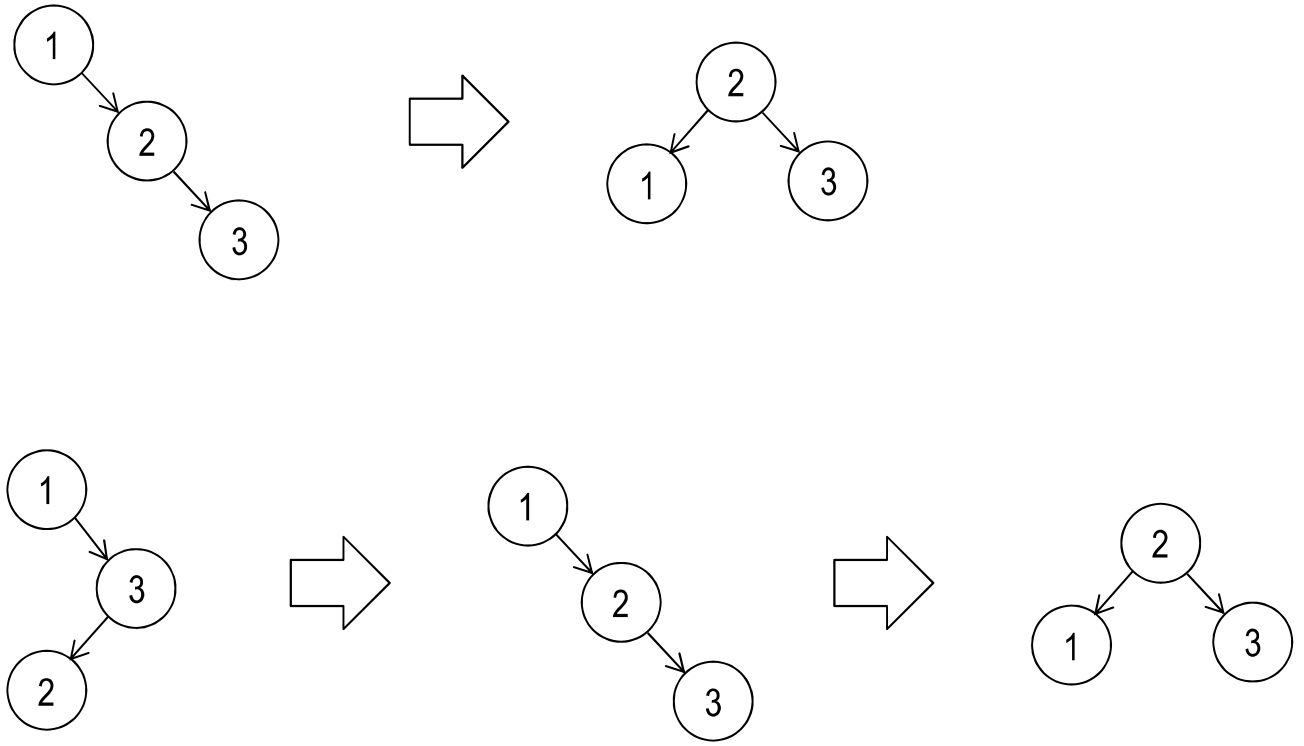
- (unbalanced) 이진탐색트리
  - 삽입(insert), 삭제(delete), 탐색(search) 시간복잡도
    - 평균 → $O(\log n)$
    - 최악의 경우 → $O(n)$

이진탐색트리 삽입 순서: 1, 3, 4, 6, 7, 9

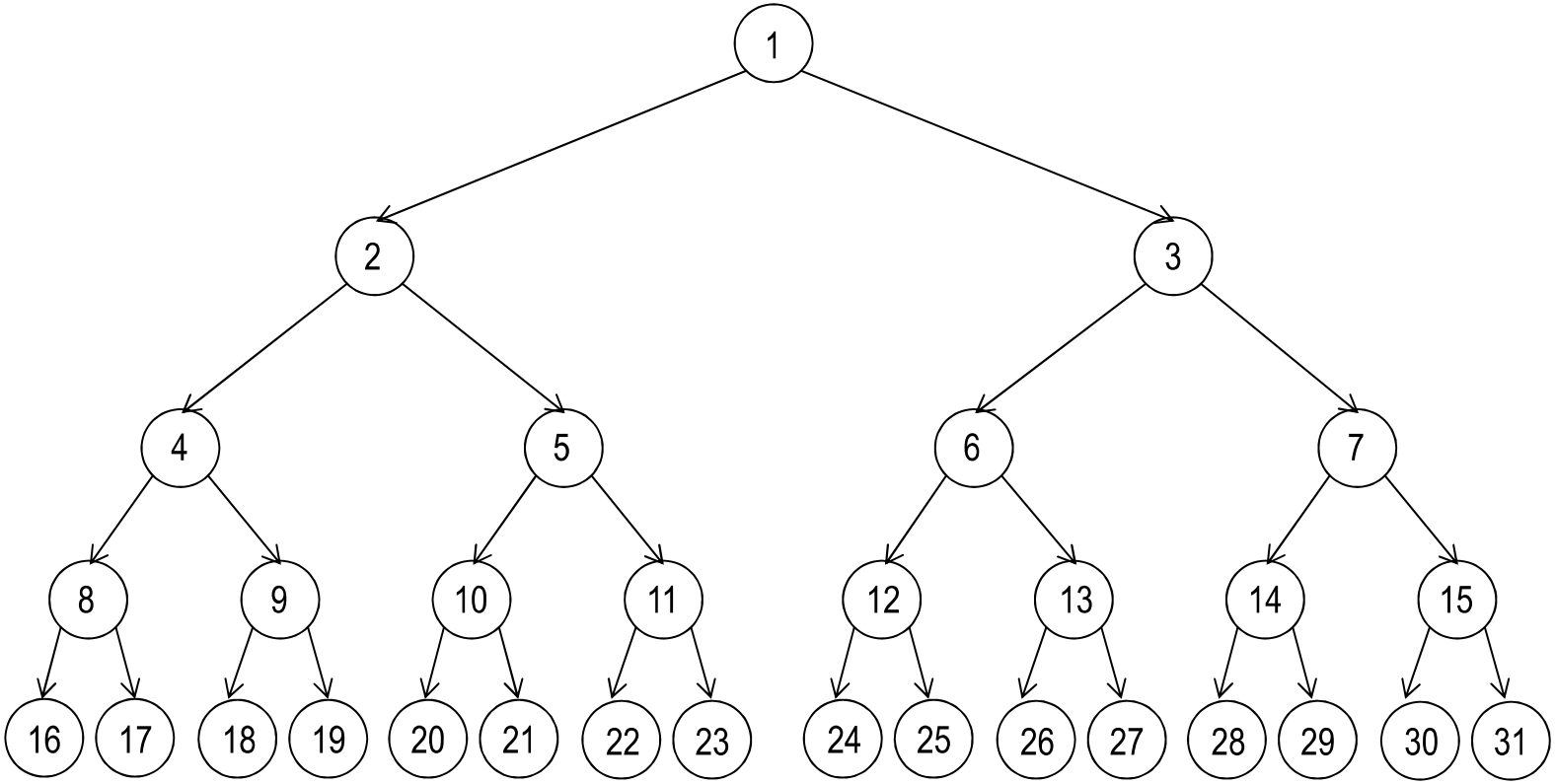


균형이진탐색트리

- 균형이진탐색트리 (balanced binary search tree)



완전이진트리 높이



n개 노드로 이루어진 완전이진트리의 높이는 $\lfloor \log_2 n \rfloor$

n=7일 때, 트리 높이 $\lfloor \log_2 7 \rfloor = 2$

n=15일 때, 트리 높이 $\lfloor \log_2 15 \rfloor = 3$

n=31일 때, 트리 높이 $\lfloor \log_2 31 \rfloor = 4$

노드 수가 십억개일 때, 트리 높이는 $\lfloor \log_2 10^9 \rfloor < \lfloor \log_2 2^{30} \rfloor = 30$

균형탐색트리

- (unbalanced) 이진탐색트리
  - 삽입(insert), 삭제(delete), 탐색(search) 시간복잡도
    - 평균 → O(log n)
    - 최악의 경우 → O(n)
- 균형탐색트리(self-balancing search tree)
  - 자바 클래스 TreeSet, TreeMap → Red-Black tree로 구현

연산	균형탐색트리	시간복잡도	
		평균	최악
삽입 삭제 탐색	Red-Black tree (이진탐색트리)	log n	log n
	AVL tree (이진탐색트리)	log n	log n
	2-3 tree	log n	log n
	B-tree	log n	log n

References

- C로 쓴 자료구조론 (Fundamentals of Data Structures in C, Horowitz et al.). 이석호 역. 사이텍미디어. 1993.
- 쉽게 배우는 알고리즘: 관계 중심의 사고법. 문병로. 한빛아카데미. 2013.
- C언어로 쉽게 풀어 쓴 자료구조. 천인국 외 2인. 생능출판사. 2017.
- 프로그래밍 콘테스트 챌린징, Akiba 등 공저, 로드북, 2011.
- https://introcs.cs.princeton.edu/
- Introduction to Algorithms, Cormen et al., 3rd Edition (The MIT Press)