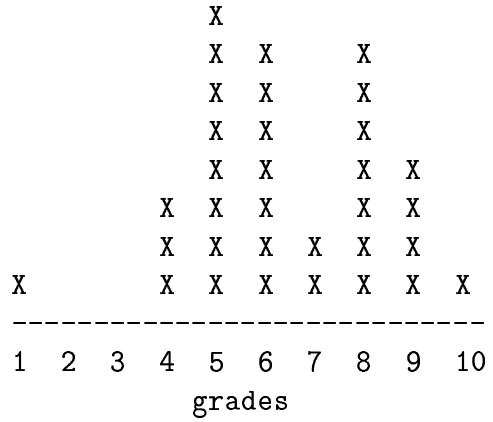
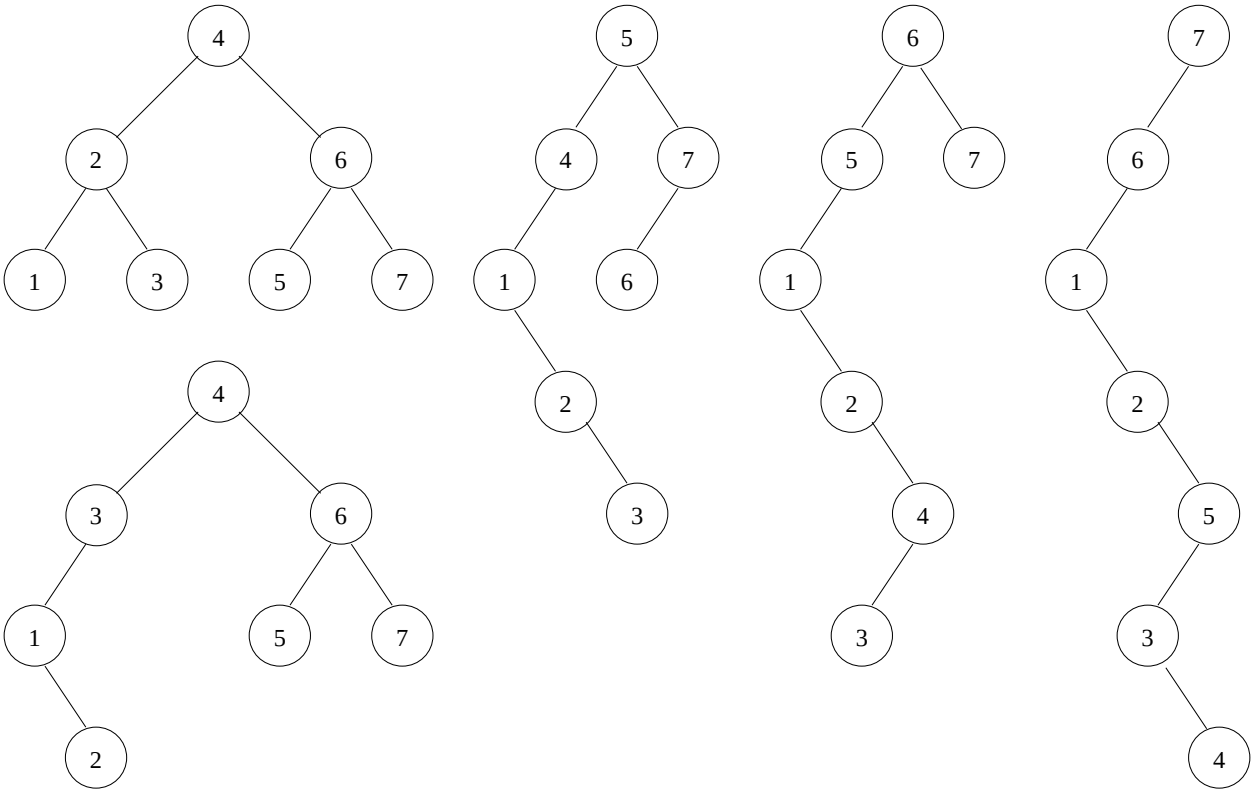
Analysis of Algorithms: Solutions 5



Problem 1

Draw five binary search trees, with heights two, three, four, five, and six. Each tree should have seven nodes, and the keys of these nodes should be 1, 2, 3, 4, 5, 6, and 7.



Problem 2
Give a recursive version of the TREE-INSERT procedure.

```
RECURSIVE-INSERT(T, z)
if root[T] = NIL
   then parent[z] ← NIL
        root[T] ← z
   else ADD-NODE(root[T], z)

ADD-NODE(x, z)
if key[z] < key[x]
   then if left-child[x] ≠ NIL
           then ADD-NODE(left-child[x], z)
           else parent[z] ← x
                left-child[x] ← z
   else if right-child[x] ≠ NIL
           then ADD-NODE(right-child[x], z)
           else parent[z] ← x
                right-child[x] ← z
```

Problem 3
Give a nonrecursive version of INORDER-TREE-WALK.

```
ITERATIVE-TREE-WALK(T)
x ← TREE-MINIMUM(root[T])
while x ≠ NIL
      do print key[x]
         x ← TREE-SUCCESSOR(x)
```

Problem 4
Write a procedure INORDER-RANGE-WALK(x, *min*, *max*) that recursively prints all nodes between *min* and *max* in the subtree rooted at x. It should be *more efficient* than INORDER-TREE-WALK; thus, it should not traverse the whole subtree.

```
INORDER-RANGE-WALK(x, min, max)
if x ≠ NIL
   then if min ≤ key[x]
           then INORDER-RANGE-WALK(left-child[x], min, max)
        if min ≤ key[x] ≤ max
           then print key[x]
        if key[x] ≤ max
           then INORDER-RANGE-WALK(right-child[x], min, max)
```