Chanh Dao Le – 986178

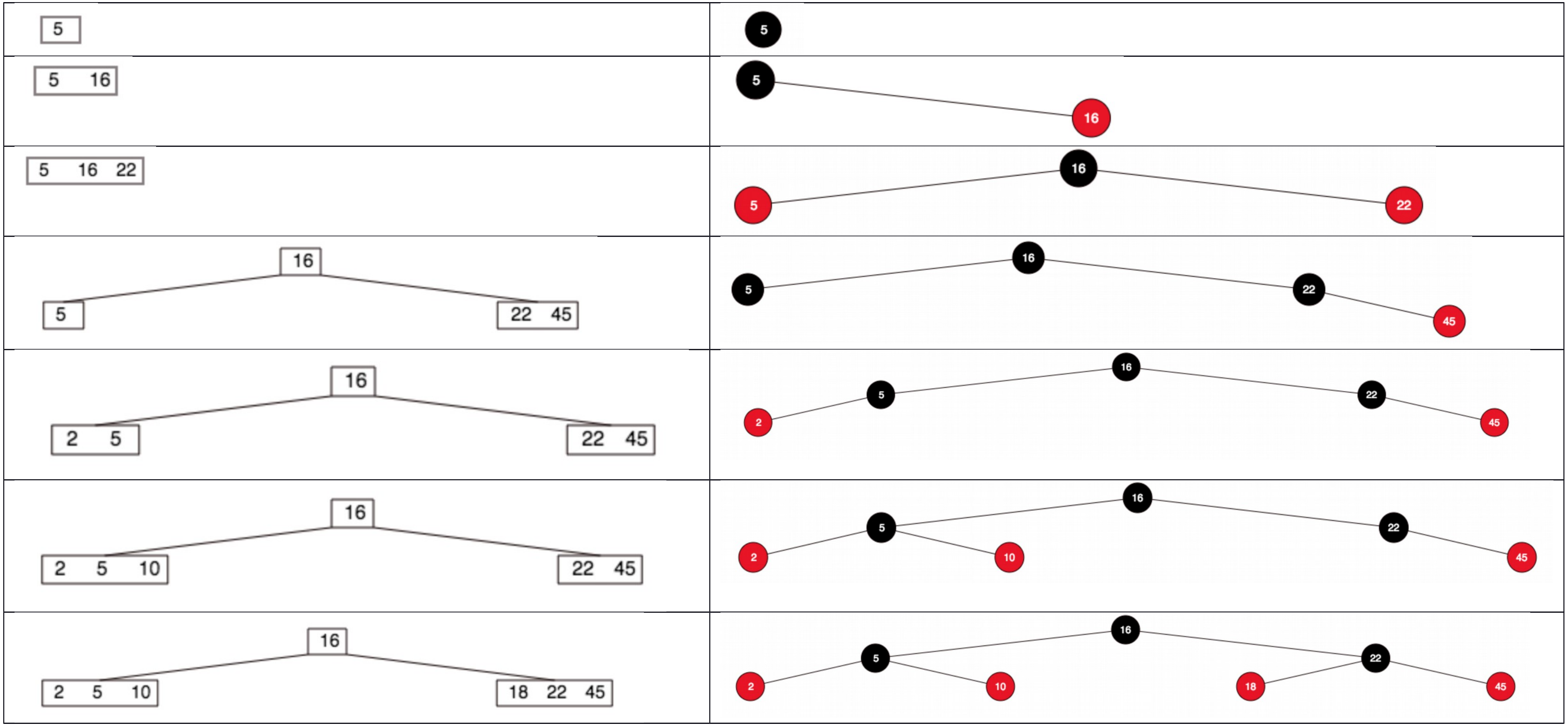
Assignment 9

R-3.11 Consider the following sequence of keys: (5, 16, 22, 45, 2, 10, 18, 30, 50, 12, 1) Consider the insertion of items with this set of keys, in the order given, into:

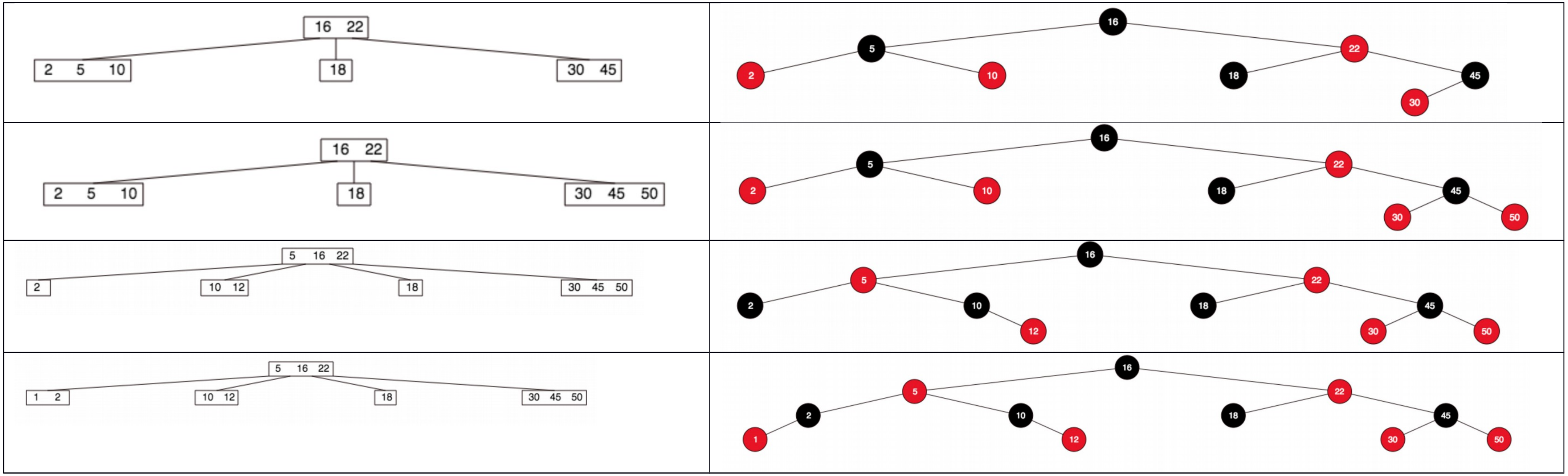
a. An initially empty (2,4) tree T’.

b. An initially empty red-black tree T’’.

Draw T’ and T’’ after each insertion



This study source was downloaded by 100000866505673 from CourseHero.com on 05-09-2023 19:26:16 GMT -05:00



R-3.14 For each of the following statements about red-black trees, determine whether it is true or false. If you think if it is true, provide a justification. If you think it is false, give a counterexample.

a. A subtree of a red-black tree is itself a red-black tree.

b. The sibling of an external node is either external or it is red.

c. Given a red-black tree T, there is an unique (2,4) tree T’ associated with T.

d. Given a (2,4) tree T, there is an unique red-black tree T’ associated with T.

a. False. Because the root of the sub tree may be red.

b. True. Because if the sibling were black the property of black depth would become invalid.

c. True. Because there are only 1 way of mapping nodes of a red-black tree into 2-nodes, 3-nodes or 4-nodes of a (2, 4) tree.

d. False. Because there are 2 ways to present a 3-node of a (2, 4) tree in a red-black tree.

Design a pseudo code algorithm isValidAVL(T) that decides whether or not a binary tree is a valid AVL tree. For this problem, we define valid to mean that the height of the left and right sub-trees of every node do not differ by more than one. What is the time complexity of your algorithm?

Algorithm isValidAVL(T)

    Input: Tree T
    Output: Whether the tree T is valid
    h <- isValidAVLHelper(T, T.root())
    if h = -1 then

This study source was downloaded by 100000866505673 from CourseHero.com on 05-09-2023 19:26:16 GMT -05:00

```
        return false
    return true
Algorithm isValidAVLHelper(T, v)
    Input: Tree T and node v
    Output: Whether the tree T is valid
    if T.isExternal(v) then
        return 0
    hl <- isValidAVLHelper(T.leftChild(v))
    if hl = -1 then
        return -1
    hr <- isValidAVLHelper(T.rightChild(v))
    if hr = -1 then
        return -1
    if |hl - hr| > 1 then
        return - 1
    return 1 + max(hl, hr)
```

The time complexity is O(n).

Design an algorithm, isPermutation(A,B) that takes two sequences A and B and determines whether or not they are permutations of each other, i.e., they contain same elements but possibly occurring in a different order. Assume the elements in A and B cannot be sorted. Hint: A and B may contain duplicates. Same problem as in previous homework, but this time use a dictionary to solve the problem. What is the worst-case time complexity of your algorithm? Justify your answer.

```
Algorithm isPermutation(A,B)
    Input: Sequences A and B
    Output: Whether they are permutations of each other
    DA <- Dictionary with hash table implementation
    If A.size() ¬= B.size() then
        return false
    for each element e in A.elements() do
        count <- DA.removeElement(e)
        if count = NO_SUCH_KEY then
            count <- 0
        DA.insertItem(e, count + 1)
    DB <- Dictionary with hash table implementation
    for each element e in B.elements() do
        count <- DB.removeElement(e)
```

This study source was downloaded by 100000866505673 from CourseHero.com on 05-09-2023 19:26:16 GMT -05:00

```
        if count = NO_SUCH_KEY then
            count <- 0
        DA.insertItem(e, count + 1)
    for each key k in DA.keys()
        countB <- DB.findElement(k)
        if countB = NO_SUCH_KEY then
            return false
        countA <- DA. findElement(k)
        if countB ¬= countA then
            return false
    return true
```

The complexity is O(n).

C-3.10 Let D be an ordered dictionary with n items implemented by means of an AVL tree (or a Red-Black tree). Show how to implement the following operation on D in time O(log n + s), where s is the size of the iterator returned:
FindAllInRange(k1, k2): Return an iterator of all the elements in D with key k such that k1 <= k <= k2

```
Algorithm findAllInRange(k1, k2)
    Input: Dictionary D and range values k1 and k2
    Output: A sequence containing all the elements in D with key k such that k1 <= k <= k2
    S <- new Sequence
    findAllInRangeHelper(T, T.root(), k1, k2, S)
    return S
Algorithm findAllInRangeHelper(T, v, k1, k2, S)
    Input: Tree T, node v, range values k1 and k2, and sequence S to contain all the elements in D with key k such that k1 <= k <= k2
    if T.isExternal(v) then
        return
    k <- v.key()
    if k > k1 then
        findAllInRangeHelper(T, T.leftChild(v), k1, k2, S)
    if k1 <= k and k <= k2
        S.insertLast(k)
    if k < k2 then
        findAllInRangeHelper(T, T.rightChild(v), k1, k2, S)
```

This study source was downloaded by 100000866505673 from CourseHero.com on 05-09-2023 19:26:16 GMT -05:00