Homework 6

OPTIONAL PROBLEMS
(No due date)

1 Written Problems

Balanced Tree

Implement a function to check if a tree is balanced. For the purposes of this question, a balanced tree is defined to be a tree such that no two leaf nodes differ in distance from the root by more than one.

Note: You may want to write additional helper methods that can be used in your solution.

Solution:

The idea is very simple: the difference of min depth and max depth should not exceed 1, since the difference of the min and max depth is the maximum distance difference possible in the tree.

Pseudocode:

```
public int maxDepth(TreeNode root){
    if (root==null) {
        return 0; }
    return 1+Math.max(maxDepth(root.left), maxDepth(root.right));
}

public int minDepth(TreeNode root){
    if (root==null) {
        return 0; }
    return 1+Math.min(minDepth(root.left), minDepth(root.right));
}

public boolean isBalanced(TreeNode root) {
    return (maxDepth(root) − minDepth(root) <= 1);
}
```

Binary Tree from sorted array

Given a sorted (increasing order) array, write an algorithm to create a binary tree with minimal height.

Solution:

We will try to create a binary tree such that for each node, the number of nodes in the left subtree and the right subtree are equal, if possible.

Algorithm:

1. Insert into the tree the middle element of the array.
2. Insert (into the left subtree) the left subarray elements
3. Insert (into the right subtree) the right subarray elements
4. Recurse

Psuedocode:

```
public TreeNode addToTree(int arr[], int start, int end){
    if (end < start) {
        return null;
    }
    int mid = (start + end) / 2;
    TreeNode n = new TreeNode(arr[mid]);
    n.left = addToTree(arr, start, mid − 1);
    n.right = addToTree(arr, mid + 1, end);
    return n;
}

public static TreeNode createMinimalBST(int array[]) {
    return addToTree(array, 0, array.length − 1);
}
```