ADSA
Problem Set 1

1. Let T be a BST. Describe an $O(n)$ time algorithm that on input $T.root$ can find the minimum absolute difference of any two keys of T. For instance, if keys of T are $3,8,1,12,7,15$, then answer will be $8 - 7 = 1$.

2. Consider a BST T. Let x and y be two of the keys of T. Is it the case that BST we get after deleting x and then y is the same as the BST we get after deleting y and then x? Either argue for it or give a counterexample.

3. Prove that Inorder-Tree-Walk of a binary tree of n elements takes $\Theta(n)$ time.

4. An array A is called k-unique if it does not contain a pair of duplicate elements within k positions of each other, that is, there is no i and j such that $A[i] = A[j]$ and $|j - i| \leq k$. Design an $O(n \log k)$ time algorithm to test if A is k-unique.

5. Argue that in a red-black tree, a red node cannot have exactly one non-NIL child.

6. Consider a red-black tree formed by inserting n nodes using the procedure discussed in the lecture. Argue that if $n > 1$, the tree has at least one red node.

7. A node x is inserted into a red-black tree and then is immediately deleted using the procedures discussed in the class. Is the resulting red-black tree always the same as the initial red-black tree? Justify your answer.

Solutions

Solution 1

Idea: Find the inorder traversal of the tree in $\Theta(n)$ time. Then, in the inorder traversal return the minimum difference of two consecutive elements.

Solution 2

See 12.3-4 here.

Solution 3

Read Theorem 12.1 from CLRS (4th edition).

Solution 4

Create an RB-tree of first k elements by inserting them one by one. But before every insertion, search whether that element is already present in the tree. If yes, then array is not k-unique. Otherwise, proceed in the following manner. In the ith step remove the ith element of the array from RB tree and before inserting $(k+i)$th element search whether it exists in RB tree. If there exists, then array is not k-unique. If we are able to process all the elements of the array without finding any duplicates, then array is k-unique.

Solution 5

A red node cannot have exactly one non-NIL child. This is because what colour can you assign to that exactly one non-NIL child. It cannot be red as its parent is also red, it can also not be black as it will disturb the black height consistency property of parent.

Solution 6

You need to follow the Insertion cases and observe that every case leaves at least one red node after the fix up.

Solution 7

See 13.4-7 here.