TÖL212M Rökstudd Forritun - Hópverkefni 6

Andri Fannar Kristjánsson

24. febrúar 2025

Hópverkefni 6

1

Sækið skrána `H6-skeleton.java` og vistið hana hjá ykkur en breytið nafni hennar í H6.java. Sækið einnig `BSTNode.java` og þýðið þá skrá. Klárið að forrita klasann og keyrið einnig forritið og sýnið útkomuna.

1.1 Svar:

Hér fyrir neðan má sjá kóðann leystu útgáfuna. Þegar skráin er keyrð fæst `0 0 0 0 1 2 3 4 5 5 5 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19` . Einnig er hægt að skoða skrána hér.

```
// Höfundur/Author:     Snorri Agnarsson, snorri@hi.is
// Author of solution: Andri Fannar Kristjánsson, afk6@hi.is

public class H6
{
    // Notkun: BSTNode s = H6.insertBST(t,x);
    // Fyrir:  t er tvíleitartré, x er heiltala.
    // Eftir:  s er tvíleitartré sem inniheldur hnúta sem
    //         hafa sömu gildi og hnútarnir í t og auk þess
    //         hnút sem inniheldur gildið x.
    //         s inniheldur því einum hnút fleiri en t og
    //         einum fleiri hnúta sem innihalda gildið x.

    // Usage:  BSTNode s = H6.insertBST(t,x);
    // Pre:    t is a binary search tree, x is an int.
    // Post:   s is a binary search tree that contains nodes
    //         which have the same values as t and in
    //         addition contains a node which contains the
    //         value x. s therefore contains one more node
    //         than t does and one more node containing
    //         the value x.
    public static BSTNode insertBST( BSTNode t, int x )
    {
        if( t == null ) { return new BSTNode(null,x,null); }
        int val = BSTNode.rootValue(t);
        BSTNode left = BSTNode.left(t);
        BSTNode right = BSTNode.right(t);
        if( x < val )
            return new BSTNode(insertBST(left,x),val,right);
        else
            return new BSTNode(left,val,insertBST(right,x));
    }

    // Notkun: int x = H6.maxInBST(t);
```

```
// Fyrir:  t != null.
// Eftir:  x er stærsta gildi í t,
//         þ.e. gildið í hnút sem er lengst til hægri í t.

// Usage:  int x = H6.maxInBST(t);
// Pre:    t != null.
// Post:   x is the largest value in t, i.e. the value in the node
//         that is farthest to the right in t.
public static int maxInBST( BSTNode t )
{
    while( BSTNode.right(t) != null )
        // Fastayrðing lykkju: t er ekki tómt og hæsta gildið í
        //                     upprunalega trénu er í hluttrénu með
        //                     rótina t.
        // Loop invariant:     t is not empty and the maximum value
        //                     of the original tree is in the
        //                     subtree rooted at t.
        t = BSTNode.right(t);
    return BSTNode.rootValue(t);
}

// Notkun: BSTNode s = H6.deleteBST(t,x);
// Fyrir:  t er tvíleitartré, x er heiltala.
// Eftir:  s er nýtt tvíleitartré sem inniheldur hnúta sem
//         hafa sömu gildi og hnútarnir í t nema einn hnút með
//         gildið x, þ.e. multiset{s} = multiset{t} - multiset{x}.
//         Ef x er ekki í t er s eins og t.

// Usage:  BSTNode s = H6.deleteBST(t,x);
// Pre:    t is a binary search tree, x is an int.
// Post:   s is a new binary search tree that contains nodes
//         which have the same values as t except one node with
//         value x, i.e. multiset{s} = multiset{t} - multiset{x}.
//         If x is not in t, then s is identical to t.
public static BSTNode deleteBST( BSTNode t, int x )
{
    if( t == null ) return null;
    int val = BSTNode.rootValue(t);
    BSTNode left = BSTNode.left(t);
    BSTNode right = BSTNode.right(t);
    if( x < val )
        return new BSTNode(deleteBST(left,x),val,right);
    if( x > val )
        return new BSTNode(left,val,deleteBST(right,x));
    if( left == null ) return right;
    if( right == null ) return left;
    int maxInLeft = maxInBST(left);
    BSTNode newLeft = deleteBST(left,maxInLeft);
    return new BSTNode(newLeft,maxInLeft,right);
}

// Notkun: H6.sort(a);
// Fyrir:  a er heiltalnafylki.
// Eftir:  a hefur verið raðað í vaxandi röð.

// Usage:  H6.sort(a);
// Pre:    a is an int array.
// Post:   a has been sorted in ascending order.
```

```
    public static void sort( int[] a )
    {
        BSTNode t = null;
        int i = 0;
        while( i != a.length )
            // Fastayrðing lykkju: 0 <= i <= a.length.
            //                     t er tvíleitartré sem inniheldur
            //                     a[0],a[1],...,a[i-1].
            // Loop invariant:     0 <= i < a.length.
            //                     t is a binary search tree that
            //                     contains a[0],a[1],...,a[i-1].
            t = insertBST(t,a[i++]);
        while( i != 0 )
            // Fastayrðing lykkju: 0 <= i <= a.length.
            //                     a[i],a[i+1],...,a[a.length-1] eru í
            //                     vaxandi röð. t er tvíleitartré sem
            //                     inniheldur a[0],a[1],...,a[i-1]
            //                     sem hafa ekki enn verið sett inn í a.
            // Loop invariant:     0 <= i <= a.length.
            //                     a[i],a[i+1],...,a[a.length-1] is in
            //                     ascending order. t is a binary search
            //                     tree that contains
            //                     a[0],a[1],...,a[i-1] which have not
            //                     yet been placed in a.
        {
            a[--i] = maxInBST(t);
            t = deleteBST(t,a[i]);
        }
    }

    public static void main( String[] args )
    {
        int[] a = new int[]
            {0,9,1,8,2,7,3,6,4,5,10,11,12,13,19,18,17,16,15,14,0,0,0,5,5,5};
        H6.sort(a);
        for( int i=0 ; i!=a.length ; i++ )
            // Fastayrðing lykkju: Gildin í a[0],a[1],...,a[i-1] hafa
            //                     verið skrifuð.
            // Loop invariant: The values in a[0],a[1],...,a[i-1] have
            //                 been written.
            System.out.print(a[i]+" ");
    }
}
```