99 questions/Solutions/59

From HaskellWiki

< 99 questions | Solutions

(**) Construct height-balanced binary trees

In a height-balanced binary tree, the following property holds for every node: The height of its left subtree and the height of its right subtree are almost equal, which means their difference is not greater than one.

```
hbalTree x = map fst . hbalTree'
    where hbalTree' 0 = [(Empty, 0)]
          hbalTree' 1 = [(Branch x Empty Empty, 1)]
          hbalTree' n =
                let t = hbalTree' (n-2) ++ hbalTree' (n-1)
                in  [(Branch x lb rb, h) | (lb,lh) <- t, (rb,rh) <- t
                                         , let h = 1 + max lh rh, h == n]
```

Alternative solution:

```
hbaltree :: a -> Int -> [Tree a]
hbaltree x 0 = [Empty]
hbaltree x 1 = [Branch x Empty Empty]
hbaltree x h = [Branch x l r |
        (hl, hr) <- [(h-2, h-1), (h-1, h-1), (h-1, h-2)],
        l <- hbaltree x hl, r <- hbaltree x hr]
```

If we want to avoid recomputing lists of trees (at the cost of extra space), we can use a similar structure to the common method for computation of all the Fibonacci numbers:

```
hbaltree :: a -> Int -> [Tree a]
hbaltree x h = trees !! h
  where trees = [Empty] : [Branch x Empty Empty] :
                zipWith combine (tail trees) trees
        combine ts shortts = [Branch x l r |
                (ls, rs) <- [(shortts, ts), (ts, ts), (ts, shortts)],
                l <- ls, r <- rs]
```