99 questions/Solutions/60

From HaskellWiki

< 99 questions | Solutions

(**) Construct height-balanced binary trees with a given number of nodes

Consider a height-balanced binary tree of height H. What is the maximum number of nodes it can contain?

Clearly, MaxN = 2**H - 1. However, what is the minimum number MinN? This question is more difficult. Try to find a recursive statement and turn it into a function

`minNodes`

that returns the minimum number of nodes in a height-balanced binary tree of height H. On the other hand, we might ask: what is the maximum height H a height-balanced binary tree with N nodes can have? Write a function

`maxHeight`

that computes this.

Now, we can attack the main problem: construct all the height-balanced binary trees with a given nuber of nodes. Find out how many height-balanced trees exist for N = 15.

```haskell
hbalTreeNodes _ 0 = [Empty]
hbalTreeNodes x n = concatMap toFilteredTrees [minHeight .. maxHeight]
    where toFilteredTrees = filter ((n ==) . countNodes) . hbalTree x

          -- Similar to the Fibonacci sequence but adds 1 in each step.
          minNodesSeq = 0:1:zipWith ((+).(1+)) minNodesSeq (tail minNodesSeq)
          minNodes = (minNodesSeq !!)

          minHeight = ceiling $ logBase 2 $ fromIntegral (n+1)
          maxHeight = (fromJust $ findIndex (>n) minNodesSeq) - 1

          countNodes Empty = 0
          countNodes (Branch _ l r) = countNodes l + countNodes r + 1
```

Another solution generates only the trees we want:

```haskell
-- maximum number of nodes in a weight-balanced tree of height h
maxNodes :: Int -> Int
maxNodes h = 2^h - 1

-- minimum height of a weight-balanced tree of n nodes
minHeight :: Int -> Int
minHeight n = ceiling $ logBase 2 $ fromIntegral (n+1)

-- minimum number of nodes in a weight-balanced tree of height h
minNodes :: Int -> Int
```

```haskell
minNodes h = fibs !! (h+2) - 1

-- maximum height of a weight-balanced tree of n nodes
maxHeight :: Int -> Int
maxHeight n = length (takeWhile (<= n+1) fibs) - 3

-- Fibonacci numbers
fibs :: [Int]
fibs = 0 : 1 : zipWith (+) fibs (tail fibs)

hbalTreeNodes :: a -> Int -> [Tree a]
hbalTreeNodes x n = [t | h <- [minHeight n .. maxHeight n], t <- baltree h n]
  where
        -- baltree h n = weight-balanced trees of height h with n nodes
        -- assuming minNodes h <= n <= maxNodes h
        baltree 0 n = [Empty]
        baltree 1 n = [Branch x Empty Empty]
        baltree h n = [Branch x l r |
                (hl,hr) <- [(h-2,h-1), (h-1,h-1), (h-1,h-2)],
                let min_nl = max (minNodes hl) (n - 1 - maxNodes hr),
                let max_nl = min (maxNodes hl) (n - 1 - minNodes hr),
                nl <- [min_nl .. max_nl],
                let nr = n - 1 - nl,
                l <- baltree hl nl,
                r <- baltree hr nr]
```