99 questions/Solutions/31

From HaskellWiki

< 99 questions | Solutions

(**) Determine whether a given integer number is prime.

Well, a natural number *k* is a prime number if it is larger than **1** and no natural number *n >= 2* with *n^2 <= k* is a divisor of *k*. However, we don't actually need to check all natural numbers *n <= sqrt k*. We need only check the ***primes*** *p <= sqrt k*:

```
isPrime :: Integral a => a -> Bool
isPrime k = k > 1 &&
   foldr (\p r -> p*p > k || k `rem` p /= 0 && r)
      True primesTME
```

This uses

```
{-# OPTIONS_GHC -O2 -fno-cse #-}
-- tree-merging Eratosthenes sieve
--  producing infinite list of all prime numbers
primesTME = 2 : gaps 3 (join [[p*p,p*p+2*p..] | p <- primes'])
  where
    primes' = 3 : gaps 5 (join [[p*p,p*p+2*p..] | p <- primes'])
    join  ((x:xs):t)        = x : union xs (join (pairs t))
    pairs ((x:xs):ys:t)     = (x : union xs ys) : pairs t
    gaps k xs@(x:t) | k==x  = gaps (k+2) t
                    | True  = k : gaps (k+2) xs
```

The tree-merging Eratosthenes sieve here seems to strike a good balance between efficiency and brevity. More at Prime numbers haskellwiki page. The semi-standard `union` function is readily available from

`Data.List.Ordered`

package, put here just for reference:

```
-- duplicates-removing union of two ordered increasing lists
union (x:xs) (y:ys) = case (compare x y) of
           LT -> x : union  xs  (y:ys)
           EQ -> x : union  xs     ys
           GT -> y : union (x:xs)  ys
```

Here is another solution, intended to be extremely short while still being reasonably fast.

```
isPrime :: (Integral a) => a -> Bool
isPrime n | n < 4 = n > 1
isPrime n = all ((/=0).mod n) $ 2:3:[x + i | x <- [6,12..s], i <- [-1,1]]
            where s = floor $ sqrt $ fromIntegral n
```

This one does not go as far as the previous, but it does observe the fact that you only need to check numbers of the form 6k +/- 1 up to the square root. And according to some quick tests (nothing extensive) this version can run a bit faster in some cases, but slower in others; depending on optimization settings and the size of the input.

There is a subtle bug in the version above. I'm new here (the wiki and the language) and don't know how corrections are best made (here, or on discussion?). Anyway, the above version will fail on 25, because the bound of s is incorrect. It is x+i that is bounded by the sqrt of the argument, not x. This version will work correctly:

```
isPrime n | n < 4 = n /= 1
isPrime n = all ((/=0) . mod n) $ takeWhile (<= m) candidates
        where candidates = (2:3:[x + i | x <- [6,12..], i <- [-1,1]])
              m = floor . sqrt $ fromIntegral n
```

Retrieved from "https://wiki.haskell.org/index.php?title=99_questions/Solutions/31&oldid=57439"
Category:

- Programming exercise spoilers

- This page was last modified on 18 January 2014, at 19:42.
-