Problem 002 - Fibonacci Numbers

Oliver Krischer

October 2021

Contents

1 Problem 002

Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:

$$1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...$$

By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms!

1.1 Recursive Implementation with Memoization

```
{-# OPTIONS_GHC -Wno-incomplete-patterns #-}
import Criterion.Main ( defaultMain, bench, bgroup, whnf )
```

This recursive implementation with memoization ist substancially faster then a naive recursive implementation, which would follow the mathematical rule: $fib(n) = fib(n-1) + fib(n-2)$.

```haskell
fibMem :: Int -> Int
fibMem limit = sum $ filter even $ run limit [1,1]
  where
  run limit memo@(n1:n2:_)
    | next > limit = memo
    | otherwise = run limit (next:memo)
    where next = n1 + n2
```

1.2 Imperative Implementation

While the recursive implementation was based on working with lists, the following implementation mimics an imperative solution in which the values of $(a = n - 2)$ and $(b = n - 1)$ and the accumulated `sum` are passed to the next recursive call:

```haskell
fibImp :: Int -> Int
fibImp limit = run limit (1,1) 0
  where
  run limit (a,b) sum
    | c > limit = sum
    | otherwise =
      if even c then run limit (b,c) (sum+c)
      else run limit (b,c) sum
    where c = a + b
```

1.3 Further Improving

Looking at the Fibonacci sequence

$$1, 1, \mathbf{2}, 3, 5, \mathbf{8}, 13, 21, \mathbf{34}, 55, 89, \mathbf{144}, ...$$

we can easily see that every third Fibonacci number is even. If this holds true for all Fibonacci numbers, we can get rid of the test for `even` like this:

```haskell
fibOpt :: Int -> Int
fibOpt limit = run limit (1,1,2) 0
  where
  run limit (a, b, c) sum
    | c > limit = sum
    | otherwise = run limit (a', b', c') (sum + c)
    where
    a' = c  + b
    b' = a' + c
    c' = a' + b'
```

1.4 Fibonacci Numbers

1.4.1 Theorem

Every third Fibonacci number is even.

1.4.2 Proof

Following the rule for Fibonacci numbers that every next number is the sum of it's two predecessors or more rigourus

$$fib(n) = fib(n-1) + fib(n-2), where fib\{0,1\} = 1$$

we get an *even* number if both preceeding numbers are odd, and an *odd* number if only one of the predecessors is odd. Given the starting values of $fib(n)$ with $\{1,1\}$ (both *odd*), we get 2 as the first successor, which is *even*. We now have a sequence of $\{1,1,2\}$, which is $\{odd, odd, even\}$. Given this sequence of the first three Fibonacci numbers $\{a,b,c\}$, we can show that every following sequence of three numbers $\{a',b',c'\}$ must also be $\{odd, odd, even\}$:

$$\{a,b,c\} = \{odd, odd, even\} \implies \{a',b',c'\} = \{odd, odd, even\} \tag{1a}$$

$$\begin{aligned}
\{a',b',c'\} &= \{(c+b), (a'+c), (a'+b')\} && (1b)\\
&= \{(even+odd), (a'+c), (a'+b')\} && (1c)\\
&= \{odd, (odd+even), (a'+b')\} && (1d)\\
&= \{odd, odd, (odd+odd)\} && (1e)\\
&= \{odd, odd, even\} && (1f)
\end{aligned}$$

1.5 Benchmarking the solutions

```
main :: IO ()
main = defaultMain [
  bgroup "solution" [ bench "fibMem" $ whnf fibMem 4000000
                    , bench "fibImp" $ whnf fibImp 4000000
                    , bench "fibOpt" $ whnf fibOpt 4000000
                    ]
  ]
```