Peter Braun

Fibonacci numbers in Scala

Veröffentlicht am 2012/06/20

Today, I spent some time to experiment with various ways to calculate Fibonacci numbers in Scala. Scala is a functional programming language, so the first version uses recursion:

```
1 def fib1( n : Int) : Int = n match {
2    case 0 | 1 => n
3    case _ => fib1( n-1 ) + fib1( n-2 )
4 }
```


Recursion only works well if `n` is small, otherwise you get a stack overflow exception. One way to overcome this problem is to use a loop like this:

```
1  def fib2( n : Int ) : Int = {
2    var a = 0
3    var b = 1
4    var i = 0
5
6    while( i < n ) {
7      val c = a + b
8      a = b
9      b = c
10     i = i + 1
11   }
12   return a
13 }
```

As you can see, this is not an elegant solution. A better way is to use so-called tail-recursion:

```
1 def fib3( n : Int) : Int = {
2   def fib_tail( n: Int, a:Int, b:Int): Int = n match {
3     case 0 => a
4     case _ => fib_tail( n-1, b, a+b )
5   }
6   return fib_tail( n, 0, 1)
7 }
```

For a compiler it is easy to translate tail-recursion to loops, see the Wikipedia article about tail-recursion for more information.

Now, we change the problem a little bit. We only want to compute the last six digits of the Fibonacci number. We modify our last algorithm a little bit:

```
1 def fib4( n : Int) : Int = {
2   def fib_tail( n: Int, a:Int, b:Int): Int = n match {
3     case 0 => a
4     case _ => fib_tail( n-1, b, (a+b)%1000000 )
5   }
6   return fib_tail( n, 0, 1)
7 }
```

To get the last six digits, we use a modulo operation. However, if we assume `n` beyond 1 billion, even this algorithm takes to much time. On my computer it takes about 7 seconds to computer the result for `n` = 1,000,000,000. Is there any way to improve the algorithm? Well, there is and the mathematical theory for it is called Pisano period. The Pisano period, is the period with which the sequence of Fibonacci numbers, modulo k repeats. For example, the period of Fibonacci numbers modulo k = 3 has length 8. The Pisano periods when k = 10^m with m > 2 equals 15·10^(m−1). Thus, for k = 10^6, the periodicity is 1,500,000. According to this, we change our last algorithm a last time:

```
1 def fib5( n : Int) : Int = {
2   def fib_tail( n: Int, a:Int, b:Int): Int = n match {
3     case 0 => a
4     case _ => fib_tail( n-1, b, (a+b)%1000000 )
5   }
```

```
6   return fib_tail( n%1500000, 0, 1)
7 }
```

By doing so, we can compute the Fibonacci number modulo 1,000,000 for n = 1 billion is about 10 milliseconds.

0
Coding

Kommentar hinterlassen

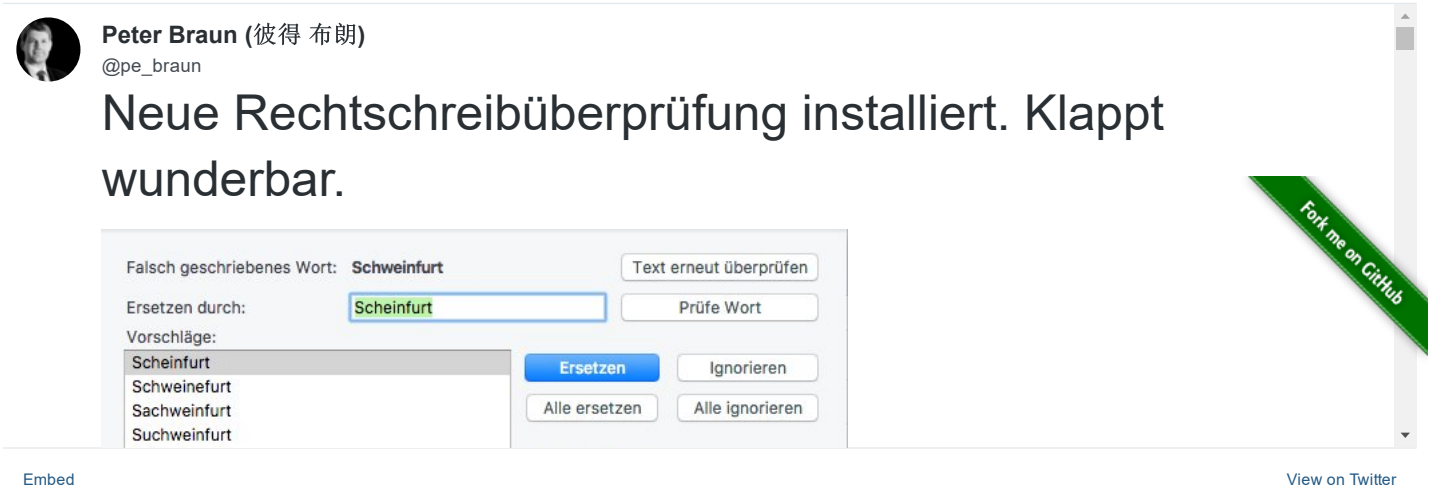
Du musst angemeldet sein, um einen Kommentar abzugeben.

Sprache auswählen Deutsch ▾

Follow me

Tweets

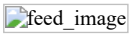
Tweets by @pe_braun



Embed View on Twitter

Instagram Feed

♡ 37 0



♡ 61 3

GitHub

braunpet @ GitHub

- OKLabWuerzburg

  Ideensammlung für die Gründung eines OK Lab in Würzburg

- Templates

  Templates for LaTeX, Maven, etc.

- GeMARA

  Model-driven development of REST based applications

- FHWS-Verteilte-Systeme

  Beispiele und Übungsaufgaben für das Modul Parallele und Verteilte Systeme an der FHWS

- braunpet.github.io

  Forward to my private home page

- migrate_git_repositories

  Script to migrate git repositories.

- FHWS-Rechnerarchitektur-Assembler-WS1213

  Einige Beispiele für Assembler Programmierung aus der Vorlesung und den Übungen

Erstellt mit WordPress und Momentous.