Ruby master - Bug #13754

bigdecimal with lower precision that Float

07/19/2017 09:50 AM - lionel_perrin (Lionel PERRIN)

Status:	Assigned		
Priority:	Normal		
Assignee:	mrkn (Kenta Murata)		
Target version:			
ruby -v:	ruby 2.4.1p111 (2017-03-22 revision 58053) [x64-mingw32]	**Backport:**	2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN

Description

Hello,

I'm not sure if I've misunderstood the bigdecimal class but in the following example, I only get 12 significant digits using bigdecimal while using Float, I get a correct value with 17 significant digits.

```
# using floats
101/0.9163472602589686 # 110.22022368622177 (OK: floating point computation)

# using bigdecimal
a = BigDecimal('101'); a.precs # [9, 18]
b = BigDecimal('0.9163472602589686'); b.precs # [18, 27]
c = a/b; c.precs
# [18, 36]              (OK: I understand that c is computed with 18 significant digits)
c.to_s # "0.110220223686e3"              (Mmm: I see only 12 significant digits)
c - BigDecimal('0.110220223686e3')
# 0.0 (Looks like c only stores 12 significant digits and not 18)
```

Using the Rational class, I've seen that the value I'm expecting is about:

```
BigDecimal.new(Rational(101/Rational('0.9163472602589686')), 25) # 0.11022022368622177467993 12e3
```

Related issues:

Related to Ruby master - Bug #8826: BigDecimal#div and #quo different behavio...	**Assigned**

History

#1 - 07/19/2017 09:55 AM - lionel_perrin (Lionel PERRIN)

- Description updated

#2 - 07/19/2017 10:12 AM - lionel_perrin (Lionel PERRIN)

- Description updated

#3 - 07/19/2017 10:12 AM - lionel_perrin (Lionel PERRIN)

- Description updated

#4 - 07/19/2017 10:13 AM - lionel_perrin (Lionel PERRIN)

- Description updated

#5 - 08/30/2017 08:54 AM - mrkn (Kenta Murata)

- Assignee set to mrkn (Kenta Murata)

- Status changed from Open to Assigned

#6 - 12/23/2017 11:20 PM - mrkn (Kenta Murata)

- Target version set to 2.6

https://github.com/ruby/bigdecimal/issues/94

#7 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#8 - 05/20/2018 09:16 PM - karatedog (Földes László)

That is the same problem as here: https://bugs.ruby-lang.org/issues/8826
#/ is the same method as #quo (according to documentation both methods are defined in 'bigdecimal.c' at line 1281). Currently you can divide a bigdecimal by using #/, #quo and #div but I don't really understand the design behind these methods (on a "which should do what" level).

#div accepts a precision argument, while #quo does not. Without precision argument #div returns Fixnum even if its first argument is a Float, it even returns Fixnum if both divisor and dividend are Float..
Thus far I don't know any method that could be able to calculate a division AND set the proper precision on the result. What you can do is to manually set precision by using #div. If you set the precision to the same amount as the divisor, you will not miss any significant digits, the drawback is that you will see a lot of digit repetition for most of the numbers.

(1019 is a long prime, its reciprocal has 1018 significant digits)

```
> BigDecimal(1).div(1019,1019).to_s
```

#9 - 09/11/2020 06:39 PM - jeremyevans0 (Jeremy Evans)

- Related to Bug #8826: BigDecimal#div and #quo different behavior and inconsistencies added