Ruby trunk - Bug #8826

BigDecimal#div and #quo different behavior and inconsistencies

08/28/2013 03:43 AM - karatedog (Földes László)

Status:	Assigned		
Priority:	Normal		
Assignee:	mrkn (Kenta Murata)		
Target version:			
ruby -v:	ruby 2.0.0p247 (2013-06-27 revision 41674) [i686-linux]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN

Description

BigDecimal's #div and #quo method behave differently despite the #div documentation says: "See BigDecimal#quo"

#div returns Fixnum if there is no precision argument for #div (this parameter is not documented):

```
2.0.0-p247 :018 > BigDecimal(5).div(5).class
 => Fixnum
2.0.0-p247 :031 > BigDecimal(5).div(5.1).class
 => Fixnum
```

#div returns Fixnum even for a Float argument:

```
2.0.0-p247 :118 > BigDecimal(5).div(5.01)
 => 0
```

It returns Fixnum even if every argument is BigDecimal:

```
2.0.0-p247 :043 > BigDecimal(5).div(BigDecimal(5.1,5)).class
 => Fixnum
```

When provided the precision argument, #div returns BigDecimal:

```
2.0.0-p247 :036 > BigDecimal(5).div(5,8).class
 => BigDecimal
2.0.0-p247 :131 > BigDecimal(5).div(BigDecimal(5.1,5),8).class
 => BigDecimal
```

But first argument cannot be Float along with precision:

```
2.0.0-p247 :032 > BigDecimal(5).div(5.1,8).class
ArgumentError: Float can't be coerced into BigDecimal without a precision
    from (irb):32:in `div'
    from (irb):32
    from /home/karatedog/.rvm/rubies/ruby-2.0.0-p247/bin/irb:13:in `<main>'
```

Whereas #quo does not accept a precision argument and returns BigDecimal (hence no configurable precision here, although the documentation says that #quo applies round operation):

```
2.0.0-p247 :121 > BigDecimal(5).quo(5.01)
 => #<BigDecimal:8bacf68,'0.9980039920 1596806387 225549E0',27(45)>
```

Circumventing the precision with class method does not work on #quo, it's like the limit is maxed:

```
2.0.0-p247 :135 > BigDecimal::limit(5)
 => 5
2.0.0-p247 :136 > BigDecimal(1).quo(3)
 => #<BigDecimal:899c778,'0.33333E0',9(36)>
2.0.0-p247 :080 > BigDecimal::limit(50)
 => 5
2.0.0-p247 :081 > BigDecimal(1).quo(3)
 => #<BigDecimal:8a92d94,'0.3333333333 33333333E0',18(36)>
```

Precision does not seem to be automatic:

```
2.0.0-p247 :141 > BigDecimal::limit(500)
 => 100
2.0.0-p247 :142 > BigDecimal(1).quo(229)
 => #<BigDecimal:8be67f4,'0.4366812227 074236E-2',18(36)>
```

229 is a full period prime, its reciprocal yields 228 fractional digits before repetition.

Whereas #div's precision can be larger than #div's:

```
2.0.0-p247 :109 > BigDecimal(1).div(3,19)
 => #<BigDecimal:8acb2d4,'0.3333333333 333333333E0',27(54)>
```

And for 229:

```
2.0.0-p247 :144 > BigDecimal(1).div(229,250)
 => #<BigDecimal:8bc8b28,'0.4366812227 0742358078 6026200873 3624454148 4716157205 2401746724 8908
296943 2314410480 3493449781 6593886462 8820960698 6899563318 7772925764 1921397379 9126637554 585
1528384 2794759825 3275109170 3056768558 9519650655 0218340611 3537117903 9301310043 6681222707...
```

Expected behavior:

- One division method to rule them all :-) (#Division)
- Never truncate a result (aka no Fixnum/Bignum as result). If someone uses BigDecimal, they probably wanted large precision instead of truncating the results by default.
- #Division should accept Float, Rational, String, Complex, Integer, BigDecimal as divisor, even Float w/o precision. (This is intended as a full list of acceptable classes, #div and #quo can already take different classes as arguments).
- #Division should accept a precision argument which would override ::Limit (as this happens in many instance method), this argument is optional. Without precision argument, use ::Limit

as for now proper calculation only happens if:

- method is #div
- the divisor is converted to BigDecimal
- a precision argument is given to #div

History

#1 - 09/03/2013 01:23 PM - zzak (Zachary Scott)

- Category set to ext

- Status changed from Open to Assigned

- Assignee set to mrkn (Kenta Murata)

The documentation for BigDecimal#div is aliases under #quo, sorry for the confusion. I will fix this!

mrkn (Kenta Murata) Can you also comment?

#2 - 12/10/2016 07:26 AM - mrkn (Kenta Murata)

- Backport changed from 1.9.3: UNKNOWN, 2.0.0: UNKNOWN to 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN

- Description updated

#3 - 12/20/2016 03:26 AM - mrkn (Kenta Murata)

Sorry for the late response.
I will fix this bug in the version of bigdecimal after releasing 1.3.

#4 - 05/20/2018 06:55 PM - karatedog (Földes László)

How should #div work? The above behavior is still in Ruby 2.5.1.