How about adding Integer#bitsize (actually Fixnum#bitsize and Bignum#bitsize)?

Integer#bitsize returns the position of the most significant bit in the absolute value. (The position of the least significant bit is 1.)
It returns 0 if no bit set (i.e. the value 0).

Mathematically, n.bitsize is ceil(log2(abs(n)+1)).

Sometimes we want to know the size of a integer.

- Determine the size of an integer in some format.
  Although there are various formats, bitsize is a key property to determine the result size.
  Several examples:
  - If a format is 4 bytes for absolute value, it overflows if 32 <= n.bitsize.
  - If a format is 4 bytes for sign bit with absolute value, it overflows if 31 <= n.bitsize.
  - If a format is 4 bytes for 2's complement format, it overflow if 31 <= n.bitsize && n != -2**31.
  - BER-compressed integer needs (n.bitsize+6)/7 bytes when n > 0. BER-compressed integer is an example of VLQ.
    [URL](http://en.wikipedia.org/wiki/Variable-length_quantity)
  - Elias gamma coding needs 2*n.bitsize-1 bits.
    [URL](https://en.wikipedia.org/wiki/Elias_gamma_coding)
  - Elias delta coding needs 2*n.bitsize.bitsize+n.bitsize-2 bits.
    [URL](https://en.wikipedia.org/wiki/Elias_delta_coding)

- bitsize may be used to estimate the time or space cost of an algorithm.
  For example, the result size of integer multiplication, x*y, is x.bitsize + y.bitsize.
  The number of comparisons of binary search is sorted_array.length.bitsize, etc.
  This is because n.bitsize is an approximation of log2(abs(n)).
  So Math.log2 can be used for this purpose too.
  However bitsize may be preferable if floating point error is not desirable.

There are several software which has similar feature.

- Python 3.1 has int.bit_length().
  [URL](http://docs.python.org/dev/library/stdlib.html)
  [URL](http://docs.python.org/3.1/whatsnew/3.1.html)
  [URL](http://bugs.python.org/issue3439)

- Java java.math.BigInteger has bitLength() method.
  [URL](http://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html#bitLength())

- Mathematica has BitLength.
  [URL](http://reference.wolfram.com/mathematica/ref/BitLength.html)

- GMP has mpz_sizeinbase(n, base).
  [URL](http://gmplib.org/manual/Miscellaneous-Integer-Functions.html)

- NetBSD 5.0 has ilog2().
  [URL](http://netbsd.gw.com/cgi-bin/man-cgi?ilog2++NetBSD-6.0)

I think there are two concerns for this issue.

- method name
behavior for zero and negative number

I named the method as bitsize, mainly because there is Fixnum#size and Bignum#size. However I'm open for other names such as:

- bitlength
- numbits
- ilog2
- maxbit

Some names may suggest different behavior, though.

The behavior for zero and negative number is not trivial.

Python adopts ceil(log2(abs(n)+1)) but
Java and Mathematica adopts ceil(log2(n < 0 ? -n : n+1)).
The difference is absolute number v.s. 2's complement number.

Some people may prefer ilog2, which name suggests ilog2(0) raise an error.

I choose ceil(log2(abs(n)+1)). (i.e. absolute number, same as Python).
I think absolute number is easier to understand than 2’s complement for many people.

I attached the implementation as bitsize.patch.
The patch implements both Bignum#bitsize and Fixnum#bitsize in bignum.c.
It is because Fixnum#bitsize uses bitsize macro and it is defined in bignum.c.
Maybe, the macro should be moved to internal.h and the implementation of Fixnum#bitsize should be moved to numeric.c.

Any comments?

Associated revisions

Revision 89744e75 - 08/31/2013 03:09 PM - akr (Akira Tanaka)

- bignum.c (rb_big_bit_length): New method. (rb_fix_bit_length): Ditto. [ruby-core:56247] [Feature #8700]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@42746 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 42746 - 08/31/2013 03:09 PM - akr (Akira Tanaka)

- bignum.c (rb_big_bit_length): New method. (rb_fix_bit_length): Ditto. [ruby-core:56247] [Feature #8700]

Revision 42746 - 08/31/2013 03:09 PM - akr (Akira Tanaka)

- bignum.c (rb_big_bit_length): New method. (rb_fix_bit_length): Ditto. [ruby-core:56247] [Feature #8700]

Revision 42746 - 08/31/2013 03:09 PM - akr (Akira Tanaka)

- bignum.c (rb_big_bit_length): New method. (rb_fix_bit_length): Ditto. [ruby-core:56247] [Feature #8700]

Revision 42746 - 08/31/2013 03:09 PM - akr (Akira Tanaka)

- bignum.c (rb_big_bit_length): New method. (rb_fix_bit_length): Ditto. [ruby-core:56247] [Feature #8700]

Revision 42746 - 08/31/2013 03:09 PM - akr (Akira Tanaka)

- bignum.c (rb_big_bit_length): New method. (rb_fix_bit_length): Ditto. [ruby-core:56247] [Feature #8700]

Revision 42746 - 08/31/2013 03:09 PM - akr (Akira Tanaka)

- bignum.c (rb_big_bit_length): New method. (rb_fix_bit_length): Ditto. [ruby-core:56247] [Feature #8700]

Revision 42746 - 08/31/2013 03:09 PM - akr (Akira Tanaka)

- bignum.c (rb_big_bit_length): New method. (rb_fix_bit_length): Ditto. [ruby-core:56247] [Feature #8700]

Revision 42746 - 08/31/2013 03:09 PM - akr (Akira Tanaka)

- bignum.c (rb_big_bit_length): New method. (rb_fix_bit_length): Ditto. [ruby-core:56247] [Feature #8700]

History

#1 - 07/29/2013 08:43 AM - headius (Charles Nutter)
"bitlength" seems more in line with other platforms. Also, Fixnum#size represents octet size, not bit size.

For zero, bitlength == 0, always.

For negative numbers...we either decide to always return the bit length for a specific representation (two's complement or something else) or we provide a way to also query the representation. I prefer the former.

#2 - 07/29/2013 09:02 AM - phluid61 (Matthew Kerwin)

headius (Charles Nutter) wrote:

+1.

"bitlength" seems more in line with other platforms. Also, Fixnum#size represents octet size, not bit size.

For zero, bitlength == 0, always.

For negative numbers...we either decide to always return the bit length for a specific representation (two's complement or something else) or we provide a way to also query the representation. I prefer the former.

Alternatively, although no one else does it, the bitlength of a negative number could be negative the bitlength of the absolute value. E.g

-1.bitlength==-1, -7.bitlength==-3

#3 - 08/01/2013 08:47 PM - akr (Akira Tanaka)

- File bitlength.patch added

akr (Akira Tanaka) wrote:

There are several software which has similar feature.

- Python 3.1 has int.bit_length().
  [http://docs.python.org/dev/library/stdtypes.html](http://docs.python.org/dev/library/stdtypes.html)
  [http://docs.python.org/3.1/whatsnew/3.1.html](http://docs.python.org/3.1/whatsnew/3.1.html)
  [http://bugs.python.org/issue3439](http://bugs.python.org/issue3439)

- Java java.math.BigInteger has bitLength() method.
  [http://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html#bitLength()](http://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html#bitLength())

- Mathematica has BitLength.

- GMP has mpz_sizeinbase(n, base).
  [http://gmplib.org/manual/Miscellaneous-Integer-Functions.html](http://gmplib.org/manual/Miscellaneous-Integer-Functions.html)

- NetBSD 5.0 has ilog2().

I look out more.

- Go has BitLen.

- OpenSSL has BN_num_bits.
  [http://www.openssl.org/docs/crypto/BN_num_bytes.html](http://www.openssl.org/docs/crypto/BN_num_bytes.html)

- LibTomMath has mp_count_bits.
  [http://libtom.org/](http://libtom.org/)

- gcrypt has gcry_mpi_get_nbits.

- Scala has bitLength.

- CommonLisp has integer-length.

- CLN has integer_length.
  [http://www.ginac.de/CLN/cln.html#Exact-numbers](http://www.ginac.de/CLN/cln.html#Exact-numbers)
They behave on negative values for absolute value or two's complement as follows.

absolute value, ceil(log2(abs(n)+1)):
Python (bit_length)
Go (BitLen)
GMP (mpz_sizeinbase)
OpenSSL (BN_num_bits)
LibTomMath (mp_count_bits)
gcrypt (gcry_mpi_get_nbits)

two's complement, ceil(log2(n < 0 ? -n : n+1)):
Java (bitLength)
Scala (bitLength)
Mathematica (BitLength)
CommonLisp (integer-length)
CLN (integer_length)

It seems "bit length" is more common than other names.
So I changed the method name to "bitlength".

Both absolute value and two's complement are common.
I think it's difficult to say one is better.
(My patch's bitlength is absolute value.)

How do you think, matz?

#4 - 08/05/2013 09:48 PM - akr (Akira Tanaka)
- File bit_length.patch added

I updated the patch because I change the method name to bit_length.
(I added an under score.)

#5 - 08/17/2013 01:03 AM - akr (Akira Tanaka)

akr (Akira Tanaka) wrote:

- bitsize may be used to estimate the time or space cost of an algorithm. For example, the result size of integer multiplication, x*y, is x.bitsize + y.bitsize. The number of comparisons of binary search is sorted_array.length.bitsize, etc. This is because n.bitsize is an approximation of log2(abs(n)). So Math.log2 can be used for this purpose too. However bitsize may be preferable if floating point error is not desirable.

I found another reason bit_length is preferable over Math.log2.

Math.log2(n) returns Infinity when n is not representable as double.

% ./ruby -e 'n = 3**4; 10.times { n = n * n; p [n.class, n.size*8, n.bit_length, Math.log2(n)] }'

[Fixnum, 64, 13, 12.679700005769249]
[Fixnum, 64, 26, 25.359400011538497]
[Fixnum, 64, 51, 50.718800023076994]
[Bignum, 104, 102, 101.43760004615399]
[Bignum, 208, 203, 202.87520009230798]
[Bignum, 408, 406, 405.75040016461596]
[Bignum, 816, 812, 811.5008003692319]
[Bignum, 1624, 1624, Infinity]
[Bignum, 3248, 3247, Infinity]
[Bignum, 6496, 6493, Infinity]

#6 - 08/31/2013 03:47 PM - matz (Yukihiro Matsumoto)

Accepted. It should be work as 2's complement for negative numbers.

Matz.

#7 - 09/01/2013 12:09 AM - akr (Akira Tanaka)
- Status changed from Open to Closed
- % Done changed from 0 to 100

This issue was solved with changeset r42746.
Akira, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- bignum.c (rb_big_bit_length): New method. (rb_fix_bit_length): Ditto. [ruby-core:56247] [Feature #8700]

#8 - 09/24/2013 05:23 AM - fuadksd (Fuad Saud)
I like it. Pretty neat for low level bit brushing stuff.

--
Fuad Saud
Sent with Sparrow (http://www.sparrowmailapp.com/?sig)

On Saturday, August 31, 2013 at 3:47 AM, matz (Yukihiro Matsumoto) wrote:

Issue #8700 has been updated by matz (Yukihiro Matsumoto).

Accepted. It should be work as 2's complement for negative numbers.

Matz.

Feature #8700: Integer#bitsize (actually Fixnum#bitsize and Bignum#bitsize)
https://bugs.ruby-lang.org/issues/8700#change-41474

Author: akr (Akira Tanaka)
Status: Open
Priority: Normal
Assignee:
Category:
Target version:

How about adding Integer#bitsize (actually Fixnum#bitsize and Bignum#bitsize)?

Integer#bitsize returns the position of the most significant bit in the absolute value.
(The position of the least significant bit is 1.)
It returns 0 if no bit set (i.e. the value 0).

Mathematically, n.bitsize is ceil(log2(abs(n)+1)).

Sometimes we want to know the size of a integer.

- Determine the size of an integer in some format. Although there are various formats, bitsize is a key property to determine the result size.
  Several examples:
  - If a format is 4 bytes for absolute value, it overflows if 32 <= n.bitsize.
  - If a format is 4 bytes for sign bit with absolute value, it overflows if 31 <= n.bitsize.
  - If a format is 4 bytes for 2's complement format, it overflow if 31 <= n.bitsize & n != -2**31.
  - BER-compressed integer needs (n.bitsize+6)/7 bytes when n > 0. BER-compressed integer is an example of VLQ.
    http://en.wikipedia.org/wiki/Variable-length_quantity

  - Elias delta coding needs 2^n.bitsize<bitsize+n.bitsize-2 bits.
    https://en.wikipedia.org/wiki/Elias_delta_coding

  - bitsize may be used to estimate the time or space cost of an algorithm.
    For example, the result size of integer multiplication, x*y, is x.bitsize + y.bitsize. 
The number of comparisons of binary search is sorted_array.length<bitsize, etc.
    This is because n.bitsize is an approximation of log2(abs(n)).
    So Math.log2 can be used for this purpose too.
    However bitsize may be preferable if floating point error is not desirable.

There are several software which has similar feature.

- Python 3.1 has int.bit_length().
  http://docs.python.org/dev/library/stdtypes.html
  http://docs.python.org/3.1/whatsnew/3.1.html
  http://bugs.python.org/issue3439

- Java java.math.BigInteger has bitLength() method.
  http://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html#bitLength()

- Mathematica has BitLength.
  http://reference.wolfram.com/mathematica/ref/BitLength.html

03/18/2022
GMP has mpz_sizeinbase(n, base).
http://gmplib.org/manual/Miscellaneous-Integer-Functions.html

NetBSD 5.0 has ilog2().
http://netbsd.gw.com/cgi-bin/man-cgi?ilog2++NetBSD-6.0

I think there are two concerns for this issue.

- method name
- behavior for zero and negative number

I named the method as bitsize, mainly because there is Fixnum#size and Bignum#size. However I'm open for other names such as:

- bitlength
- numbits
- ilog2
- maxbit Some names may suggest different behavior, though.

The behavior for zero and negative number is not trivial.

Python adopts ceil(log2(abs(n)+1)) but
Java and Mathematica adopts ceil(log2(n < 0 ? -n : n+1)).
The difference is absolute number v.s. 2's complement number.

Some people may prefer ilog2, which name suggests ilog2(0) raise an error.

I choose ceil(log2(abs(n)+1))). (i.e. absolute number, same as Python).
I think absolute number is easier to understand than 2's complement for many people.

I attached the implementation as bitsize.patch.
The patch implements both Bignum#bitsize and Fixnum#bitsize in bignum.c.
It is because Fixnum#bitsize uses bitsize macro and it is defined in bignum.c.
Maybe, the macro should be moved to internal.h and the implementation of Fixnum#bitsize should be moved to numeric.c.

Any comments?
--
http://bugs.ruby-lang.org/

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>bitsize.patch</td>
<td>4.88 KB</td>
<td>07/28/2013</td>
<td>akr (Akira Tanaka)</td>
</tr>
<tr>
<td>bitlength.patch</td>
<td>4.94 KB</td>
<td>08/01/2013</td>
<td>akr (Akira Tanaka)</td>
</tr>
<tr>
<td>bit_length.patch</td>
<td>4.59 KB</td>
<td>08/05/2013</td>
<td>akr (Akira Tanaka)</td>
</tr>
</tbody>
</table>