# Ruby master - Feature #5378

## Prime.each is slow

09/29/2011 02:27 AM - mconigliaro (Mike Conigliaro)

| | |
|---|---|
| **Status:** | Assigned |
| **Priority:** | Normal |
| **Assignee:** | yugui (Yuki Sonoda) |
| **Target version:** | |

**Description**

See discussion here: https://gist.github.com/1246868

```
require 'benchmark'
require 'prime'

def primes_up_to(n)
s = [nil, nil] + (2..n).to_a
(2..(n ** 0.5).to_i).reject { |i| s[i].nil? }.each do |i|
(i ** 2).step(n, i) { |j| s[j] = nil }
end
s.compact
end

Benchmark.bm(12) do |x|
x.report('primes_up_to') { primes_up_to(2000000).inject(0) { |memo,obj| memo + obj } }
x.report('Prime.each') { Prime.each(2000000).inject(0) { |memo,obj| memo + obj } }
end

$ ruby -v
ruby 1.9.2p290 (2011-07-09 revision 32553) [x86_64-darwin10.8.0]
$ ruby lol.rb
user     system      total        real
primes_up_to  1.470000   0.020000   1.490000 (  1.491340)
Prime.each    7.820000   0.010000   7.830000 (  7.820969)
```

**Related issues:**

Has duplicate Ruby master - Feature #10354: Optimize Integer#prime? **Closed**

### History

**#1 - 10/01/2011 03:26 PM - h.shirosaki (Hiroshi Shirosaki)**

*- File prime.patch added*

It seems that converting from integer to bitmap tables in EratosthenesSieve class is slow.

This patch improves Prime performance.

```
require 'benchmark'
require 'prime'

def primes_up_to(n)
s = [nil, nil] + (2..n).to_a
(2..(n ** 0.5).to_i).reject { |i| s[i].nil? }.each do |i|
(i ** 2).step(n, i) { |j| s[j] = nil }
end
s.compact
end
Benchmark.bm(12) do |x|
x.report('primes_up_to') { p primes_up_to(1500000).inject(0) { |memo,obj| memo + obj } }
2.times do
x.report('Prime.each') { p Prime.each(1500000).inject(0) { |memo,obj| memo + obj } }
end
end
```

# before

```
$ ruby -v ~/prime_bench.rb
ruby 1.9.4dev (2011-10-01 trunk 33368) [x86_64-darwin11.1.0]
user     system      total        real
primes_up_to   2.530000   0.020000   2.550000 (  2.550595)
Prime.each     6.450000   0.010000   6.460000 (  6.461948)
Prime.each     0.880000   0.000000   0.880000 (  0.877138)
```

## after

```
$ ruby -v -Ilib ~/prime_bench.rb
ruby 1.9.4dev (2011-10-01 trunk 33368) [x86_64-darwin11.1.0]
user     system      total        real
primes_up_to   2.560000   0.020000   2.580000 (  2.583900)
Prime.each     4.630000   0.010000   4.640000 (  4.633154)
Prime.each     0.330000   0.000000   0.330000 (  0.325838)
```

**#2 - 10/03/2011 04:15 PM - calamitas (Peter Vanbroekhoven)**

Note that the primes_up_to method Mike posted is not quite optional in that the intended optimization in the form of the reject doesn't do anything. The reject is executed before the loop and so the loop is still executed for all numbers instead of just for the primes.

If you use the version below instead, it is over 2.5 times faster for 2 mil primes on my machine. That would make the new built-in version still almost 5 times slower than the pure-Ruby version. Note also that in my benchmarks I changed the inject block to just return memo and not calculate the sum because that skews the results by quite a bit; there's the extra summing, but the sum gets in the Bignum range and so it adds object creation and garbage collection.

```
def primes_up_to(n)
s = [nil, nil] + (2..n).to_a
(2..(n ** 0.5).to_i).each do |i|
if si.step(n, i) { |j| s[j] = nil }
end
end
s.compact
end
```

**#3 - 10/03/2011 11:08 PM - mame (Yusuke Endoh)**

*- Tracker changed from Bug to Feature*

**#4 - 10/03/2011 11:12 PM - mame (Yusuke Endoh)**

*- Status changed from Open to Assigned*

*- Assignee set to yugui (Yuki Sonoda)*

Hello,

Just slowness is not a bug unless it is a regression, I think.
So I moved this ticket to the feature tracker.

I believe that there is no perfect algorithm to enumerate
primes.  Any algorithm has drawback and advantage.  Note that
speed is not the single important thing.  I could be wrong,
but I guess that prime.rb does not priotize speed (especially,
linear-order cost), but high-abstract design.

Even in terms of speed, my version is about 2 times faster
than Peter's, though it uses extra memory.  So, there are
trade-offs.

```
def primes_up_to_yusuke(n)
primes = [2]
n /= 2
prime_table = [true] * n
i = 1
while i < n
if prime_table[i]
primes << j = i * 2 + 1
k = i + j
while k < n
prime_table[k] = false
k += j
end
end
```

i += 1
end
primes
end

```
                  user      system        total          real
```

primes_up_to_mike  1.720000   0.010000   1.730000 (  1.726733)
primes_up_to_peter  0.780000   0.020000   0.800000 (  0.795156)
primes_up_to_yusuke  0.410000   0.000000   0.410000 (  0.419209)
Prime.each    4.760000   0.010000   4.770000 (  4.765654)

I think every prime-aholic should implement their own favorite
algorithm by himself :-)

--
Yusuke Endoh mame@tsg.ne.jp

**#5 - 10/28/2012 12:02 AM - yhara (Yutaka HARA)**

*- Target version set to 2.6*

**#6 - 11/01/2012 05:21 AM - headius (Charles Nutter)**

JRuby numbers for the various implementations proposed (best times out of ten in-process iterations):

mconigliario's version:

```
                   user      system        total          real
```

primes_up_to   2.100000   0.000000   2.100000 (  1.062000)
Prime.each     0.980000   0.010000   0.990000 (  0.883000)

h.shirosaki's version:

```
                   user      system        total          real
```

primes_up_to   2.100000   0.010000   2.110000 (  1.014000)
Prime.each     1.030000   0.000000   1.030000 (  0.930000)

calamitas's version:

```
                   user      system        total          real
```

primes_up_to   1.130000   0.020000   1.150000 (  0.467000)
Prime.each     1.020000   0.000000   1.020000 (  0.908000)

mame's version:

```
                   user      system        total          real
```

primes_up_to   0.180000   0.000000   0.180000 (  0.143000)
Prime.each     0.970000   0.000000   0.970000 (  0.948000)

Ruby 1.9.3p286 running mame's version:

```
                   user      system        total          real
```

primes_up_to   0.380000   0.000000   0.380000 (  0.382392)
Prime.each     0.790000   0.000000   0.790000 (  0.793005)

Definitely some room for improvement over the base implementation.

**#7 - 02/05/2015 07:47 PM - marcandre (Marc-Andre Lafortune)**

*- Has duplicate Feature #10354: Optimize Integer#prime? added*

**#8 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)**

*- Target version deleted (2.6)*

## Files

| | | | |
|---|---|---|---|
| prime.patch | 4.06 KB | 10/01/2011 | h.shirosaki (Hiroshi Shirosaki) |