Ruby master - Feature #14476

Adding same_all? for checking whether all items in an Array are same

02/14/2018 03:41 PM - mrkn (Kenta Murata)

Status:	Assigned	
Priority:	Normal	
Assignee:	mrkn (Kenta Murata)	
Target version:		

Description

In this issue, I propose to introduce same_all? instance method of Array class.
This new method checks whether the all items in the receiver are same.

Today, I needed to write the code to judge whether all items in an Array are same.
I wanted to make the following expression, that I've written at first, more efficiently.

```
ary.all? {|e| e.foo == ary[0].foo }
```

I thought the following another simpler case, too.

```
ary.all? {|e| e == ary[0] }
```

As I discussed with some CRuby committers and my colleagues at Speee, Inc., I found that both cases have the following efficient expressions:

```
# for the 1st case
ary.empty? || ary[0].foo.yield_self {|e0| ary[1..-1].all? {|e| e.foo == e0 } }‰

# for the 2nd case
ary.empty? || ary[0..-2] == ary[1..-1]
```

Both expressions aren't easy to understand the original purpose that is checking whether all the items are same.

I want to give this feature a clear name to make the code readable.
And I think it should be provided as a core feature because it can be more efficient by implementing in C language.

The benchmark script is: https://gist.github.com/mrkn/26a0fcfc431a45fe809fbbef95aceaf5
I used it to find the efficient expressions.
The example result of this benchmark on my MacBook Pro is here.

```
$ ruby -v bench.rb
ruby 2.6.0dev (2018-02-14 trunk 62402) [x86_64-darwin16]
----------------------------------------
Benchmark case: shuffle
                 user     system      total        real
all?-method-1  0.001525   0.000088   0.001613 (  0.001632)
all?-method-2  0.000489   0.000031   0.000520 (  0.000565)
all?-method-3  0.000444   0.000039   0.000483 (  0.000482)
all?-item    0.000325   0.000078   0.000403 (  0.000402)
opt-method   0.655959   0.033814   0.689773 (  0.708515)
opt-item     0.000316   0.000001   0.000317 (  0.000317)

----------------------------------------
Benchmark case: tail0
                 user     system      total        real
all?-method-1  9.412810   0.231126   9.643936 (  9.681118)
all?-method-2  5.375075   0.137908   5.512983 (  5.754550)
all?-method-3  5.226132   0.167640   5.393772 (  5.507031)
all?-item    0.873700   0.007545   0.881245 (  0.917210)
opt-method   5.319648   0.172547   5.492195 (  5.633140)
opt-item     0.174349   0.001974   0.176323 (  0.183002)

----------------------------------------
Benchmark case: head0
```

```
                 user     system      total        real
all?-method-1  0.002421   0.000068   0.002489 (  0.002489)
all?-method-2  0.002169   0.000213   0.002382 (  0.002382)
all?-method-3  0.001624   0.000026   0.001650 (  0.001651)
all?-item    0.000623   0.000001   0.000624 (  0.000624)
opt-method   4.779120   0.146312   4.925432 (  4.951167)
opt-item     0.000629   0.000001   0.000630 (  0.000629)

----------------------------------------
Benchmark case: all1
                 user     system      total        real
all?-method-1  9.379650   0.255865   9.635515 (  9.683078)
all?-method-2  4.950280   0.150659   5.100939 (  5.140174)
all?-method-3  4.857898   0.129125   4.987023 (  5.003142)
all?-item    0.694113   0.001295   0.695408 (  0.702370)
opt-method   5.032373   0.121708   5.154081 (  5.189599)
opt-item     0.170540   0.002069   0.172609 (  0.180343)
```

History

#1 - 02/14/2018 04:32 PM - dsferreira (Daniel Ferreira)

Usually I use the following code to achieve the same purpose:

```
ary.uniq.size < 2
```

I didn't test but if you need to use that complex code I expect this example to be less performant.

Instead of Array#same_all? why not Array#uniq?

#2 - 02/14/2018 04:37 PM - Eregon (Benoit Daloze)

Why would it be more efficient in C?

A sufficiently smart JIT (or of course it can be done manually) could turn

```
ary.all? {|e| e.foo == ary[0].foo }
```

into (assuming foo has no side effects):

```
e0 = ary[0].foo
ary.all? {|e| e.foo == e0 }
```

and then the only performance difference is the extra comparison of ary[0].foo with itself,
which could be avoided by:

```
e0 = ary[0].foo
(1...ary.size).all? {|i| ary[i].foo == e0 }
```

#3 - 02/14/2018 04:45 PM - dsferreira (Daniel Ferreira)

I believe we could even do:

```
ary.uniq?(&:foo)
```

#4 - 02/14/2018 05:15 PM - mrkn (Kenta Murata)

uniq scans all elements, whereas all? and == don't.
And uniq allocates a new array, so uniq is always slower than all?-method-3 and opt-item in all cases of my benchmark.

Eregon, if we write it in C, we can avoid using all?. It reduces the number of block calls.

#5 - 02/14/2018 05:45 PM - dsferreira (Daniel Ferreira)

mrkn (Kenta Murata) wrote:

> uniq scans all elements, whereas all? and == don't.
> And uniq allocates a new array, so uniq is always slower than all?-method-3 and opt-item in all cases of my benchmark.

That makes sense.

I suppose that Array#uniq? could work as Array#all? returning once false?

#6 - 02/15/2018 12:38 AM - mrkn (Kenta Murata)

> I suppose that Array#uniq? could work as Array#all? returning once false?

We don't have Array#uniq?.
Do you suggest adding such a new method?

#7 - 02/15/2018 12:45 AM - dsferreira (Daniel Ferreira)

mrkn (Kenta Murata) wrote:

> We don't have Array#uniq?.
> Do you suggest adding such a new method?

That is what I meant with:

"Instead of Array#same_all? why not Array#uniq?"

Array#uniq? makes sense to me and maps well with the logic of ary.uniq.size < 2 that I mentioned earlier.

If we return earlier from the method at the first false we will have what you are looking for isn't that right, or am I missing something?

I also think it is a method that will simplify some code and people will use it.
I will use it for sure.
Developing it in C is even better because it will guarantee the best performance results.

#8 - 02/15/2018 04:53 AM - duerst (Martin Dürst)

Four points.

First, re. uniq?:

mrkn (Kenta Murata) wrote:

> We don't have Array#uniq?.
> Do you suggest adding such a new method?

I remember having suggested a method named uniq?. Unfortunately, I didn't find the issue (yet). But the meaning was different. The method checked whether all elements were unique, not whether all elements were the same. In other words, it was:

```
def uniq?
  uniq.length == length
end
```

In an efficient implementation, it would return false as soon as it detected two elements that were the same, and true otherwise.

Second, about naming:
same_all? doesn't sound good to me. The two words 'same' and 'all' don't appear in this order in English. What about all_same?, or better even, all_equal??

Third, about need:
It's clear that when you implement something in C, it should get faster. If that were the only criterion for accepting new methods into Ruby, we'd have many more methods. The way you describe it, my understanding is that you only just today had a need for such functionality, not before. I think there should be some indication that this is needed regularly in order to accept it.

Fourth, about alternative ways to write it. What about

```
ary.each_cons(2).all? { |a, b| a == b }
```

#9 - 02/15/2018 09:52 AM - nobu (Nobuyoshi Nakada)

Array#uniq? sounds like that all elements are unique, ary.uniq.size == ary.size.

#10 - 02/15/2018 03:29 PM - dsferreira (Daniel Ferreira)

nobu (Nobuyoshi Nakada) wrote:

> Array#uniq? sounds like that all elements are unique, ary.uniq.size == ary.size.

I agree it makes more sense for that situation.

#11 - 02/18/2018 03:44 AM - shevegen (Robert A. Heiler)

I agree with Martin. Perhaps alternatives such as all_same? or
even better, all_equal? could be explored/used.

Nobu wrote:

> Array#uniq? sounds like that all elements are unique, ary.uniq.size == ary.size.

Agreed.

Perhaps someone can ask matz in the dev meeting about the names. :)

#12 - 03/15/2018 05:42 AM - matz (Yukihiro Matsumoto)

- Status changed from Assigned to Rejected

Rejected. Unfortunately, the incompatibility this proposal would bring is too big.
Besides that, we have performance concern too.

Matz.

#13 - 03/15/2018 05:44 AM - matz (Yukihiro Matsumoto)

- Assignee changed from matz (Yukihiro Matsumoto) to mrkn (Kenta Murata)

- Status changed from Rejected to Assigned

Ah, sorry. Posted to the wrong proposal.

Regarding this issue, we have the naming issue. I agree to add the functionality.

Matz.

#14 - 03/15/2018 07:19 AM - matz (Yukihiro Matsumoto)

I am not satisfied with any of the candidates.

- same_all? - weird word order
- all_same? - word order is OK, but there's an ambiguity that "same" means equality or identical.
- all_equal? - the name suggests comparison is done by equal?
- uniform? - I like this best, but can uniform mean equality?

Matz.

#15 - 03/15/2018 07:42 AM - duerst (Martin Dürst)

matz (Yukihiro Matsumoto) wrote:

> - uniform? - I like this best, but can uniform mean equality?

When seeing uniform?, one thing I think about is that it means "Are all elements of the same type?".

Example:
[1, 2, 3].uniform? % => true
[:a, :b, :c].uniform? % => true
[1, :b, 'c'].uniform? % => true

#16 - 03/15/2018 07:54 AM - duerst (Martin Dürst)

duerst (Martin Dürst) wrote:

> Example:
> [1, 2, 3].uniform? % => true
> [:a, :b, :c].uniform? % => true
> [1, :b, 'c'].uniform? % => true

Sorry, this should have been:

```
[1, 2, 3].uniform? %    => true
[:a, :b, :c].uniform? % => true
[1, :b, 'c'].uniform? % => false
```

#17 - 12/10/2018 07:09 AM - naruse (Yui NARUSE)

- Target version deleted (2.6)