Ruby master - Feature #16428

Add Array#uniq?, Enumerable#uniq?

12/17/2019 01:24 PM - kyanagi (Kouhei Yanagita)

Status:	Feedback
Priority:	Normal
Assignee:	
Target version:	

Description

I propose Array#uniq?.

I often need to check if an array have duplicate elements.

This method returns true if no duplicates are found in self, otherwise returns false.
If a block is given, it will use the return value of the block for comparison.

This is equivalent to array.uniq.size == array.size, but faster.

```
% ~/tmp/r/bin/ruby -rbenchmark/ips -e 'a = Array.new(100) { rand(1000) }; Benchmark.ips { |x| x.re
port("uniq") { a.uniq.size == a.size }; x.report("uniq?") { a.uniq? } }'
Warming up --------------------------------------
                uniq    25.765k i/100ms
               uniq?    76.544k i/100ms
Calculating -------------------------------------
                uniq    278.144k (± 4.1%) i/s -      1.391M in   5.010858s
               uniq?    981.868k (± 5.1%) i/s -      4.975M in   5.081611s
```

I think the name uniq? is natural because Array already has uniq.

patch: https://github.com/ruby/ruby/pull/2762

History

#1 - 12/17/2019 02:31 PM - shevegen (Robert A. Heiler)

> I often need to check if an array have duplicate elements.

Makes sense to me; I have had situations where I needed this
too in the past (including situations for non-unique entries
in an Array), so I agree on the general use case opportunities
in this regard.

#2 - 12/18/2019 05:02 AM - duerst (Martin Dürst)

I seem to member that many years ago, I made the same proposal, and Nobu created a patch, but unfortunately, I didn't find any traces anymore on this tracker or in my mail.

Anyway, I support this proposal. It's definitely an useful functionality, and it's clearly faster than doing it indirectly via #uniq.

#3 - 12/18/2019 06:50 AM - kyanagi (Kouhei Yanagita)

- Subject changed from Add Array#uniq? to Add Array#uniq?, Enumerable#uniq?

Following a suggestion of Enumerable#uniq?, I also added Enumerable#uniq? to my patch.
Array#uniq? is left because it is faster than Enumerable#uniq?.

#4 - 04/10/2020 04:41 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Feedback

You said, "I often need to check if an array have duplicate elements". But we cannot think of the real-world use-case.
Could you elaborate on how to use the proposed #uniq? and its benefit?

Matz.

#5 - 04/10/2020 02:25 PM - kyanagi (Kouhei Yanagita)

I was developing mobile games, and I met these situations:

A card deck can't have duplicate characters.
i.e. deck.cards.map(&:character_id).uniq.size == deck.cards.size
-> deck.cards.map(&:character_id).uniq? or deck.cards.uniq?(&:character_id)

When players compose items, each of them should be different.
i.e. materials.map(&:item_id).uniq.size == materials.size
-> materials.map(&:item_id).uniq? or materials.uniq?(&:item_id)

Another situation:

I developed a registration form for relay runners.
A request body is like this:

```
# Missing sections are allowed. You can send them later.
[
  { section: 1, name: 'aaa' },
  { section: 3, name: 'bbb' },
  { section: 5, name: 'ccc' },
]
```

In this case, duplication of section is not allowed.
runners.map(&:section).uniq.size == runners.size
-> runners.map(&:section).uniq? or runners.uniq?(&:section)

I think uniq? is easier to write and read than x.uniq.size == x.size
for expression of no duplication. It's even faster.

This check is also found in Ruby's repository (bundler):
https://github.com/ruby/ruby/blob/master/spec/bundler/support/matchers.rb#L84

#6 - 04/10/2020 02:42 PM - shyouhei (Shyouhei Urabe)

kyanagi (Kouhei Yanagita) wrote in #note-5:

> I was developing mobile games, and I met these situations:
>
> A card deck can't have duplicate characters.
> i.e. deck.cards.map(&:character_id).uniq.size == deck.cards.size
> -> deck.cards.map(&:character_id).uniq? or deck.cards.uniq?(&:character_id)

So you just want to test? Why doesn't deck.cards.map(...).uniq!'s return value work?

> When players compose items, each of them should be different.
> i.e. materials.map(&:item_id).uniq.size == materials.size
> -> materials.map(&:item_id).uniq? or materials.uniq?(&:item_id)

So you just want to test? Don't you want to show the duplicated materials to the players? Does uniq? help then?

> Another situation:
>
> I developed a registration form for relay runners.
> A request body is like this:
>
> ```
> # Missing sections are allowed. You can send them later.
> [
>   { section: 1, name: 'aaa' },
>   { section: 3, name: 'bbb' },
>   { section: 5, name: 'ccc' },
> ]
> ```
>
> In this case, duplication of section is not allowed.
> runners.map(&:section).uniq.size == runners.size
> -> runners.map(&:section).uniq? or runners.uniq?(&:section)

So you just want to test? Don't you want to render error message about what is the duplicated section? Does uniq? help then?

> I think uniq? is easier to write and read than x.uniq.size == x.size
> for expression of no duplication. It's even faster.

My main question is: it isn't faster when you render error messages. How do you use it?

> This check is also found in Ruby's repository (bundler):
> https://github.com/ruby/ruby/blob/master/spec/bundler/support/matchers.rb#L84

HonestIt I don't understand what this matcher is trying to achieve.

#7 - 04/10/2020 04:06 PM - kyanagi (Kouhei Yanagita)

In my cases, I (server side) only had to check duplication because a client also have validations.
Legal users can't send a request with duplicates, so detailed error message was not required.
(If needed, I could investigate logged request.)

uniq!'s return value is also usable, but I think uniq? is more fitting.
(I'd like to check duplication, not to get uniq array.)