Ruby master - Feature #15590

Add dups to Array to find duplicates

02/06/2019 11:31 AM - xdmx (Eric Bloom)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Many times I find myself debugging data and the need of finding duplicated values inside of an Array.

Based on the amount of data it could be a simple array.detect { |value| array.count(value) > 1 } or a more performant way like

```
def dups_for(array)
  duplicated_values = []
  tmp = {}
  array.each do |value|
    duplicated_values << value if tmp[value]
    tmp[value] = true
  end
  duplicated_values
end
```

It would be awesome if there was a way directly from the core language to call dups (or another name, as it could be too similar to the current dup) on an array in order to get all the duplicated values.

I'd love to create a PR for this, but my C level is non-existent □□

History

#1 - 02/06/2019 12:52 PM - shevegen (Robert A. Heiler)

I am not entirely sure whether I understood the proposal or the code example.

What do you mean with duplicated values in an Array? Do you mean something "reversed" such as a complementary method to .uniq (Array#uniq)? Or is the suggestion related to Object#dup? https://ruby-doc.org/core-2.6.1/Object.html#method-i-dup

I assume that you more refer to a complementary method to .uniq but I am not completely sure, so I hope it's ok for you to clarify on that just to make sure, when you have some time.

(We may also have to look at the chosen name for the method; I am not sure if .dups would be an acceptable method due to potential confusion.)

#2 - 02/06/2019 12:54 PM - shevegen (Robert A. Heiler)

After re-reading, I think you may refer more to a method such as:

```
.duplicates?
```

on class Array, right?

If this is the case then I understand your example and proposal and I am slightly in favour (if it is meant as a complementary method to .uniq; at the least I remember that I had to do this a few times to detect the duplicate entries, e. g. faulty files that may keep track of dependencies for programs to compile, and had twice the same content in the same .yml file; I am sure others may have had somewhat similar use cases here and there - but again, right now I am not 100% sure if this is what Eric suggested actually).

#3 - 02/06/2019 01:07 PM - mame (Yusuke Endoh)

> Many times I find myself debugging data and the need of finding duplicated values inside of an Array

Could you elaborate the use case?

#4 - 02/06/2019 01:49 PM - xdmx (Eric Bloom)

> I assume that you more refer to a complementary method to .uniq but I am not completely sure, so I hope it's ok for you to clarify on that just to make sure, when you have some time.

Sorry for not having included an example!

Yes, I mean it as a complementary method of uniq, and actually duplicates would be much better than dups :)

> Could you elaborate the use case?

The use case is mostly: you have a list of data, which could be a list of ids, names, codes, or others and you want to know which ones of them are duplicated in the array.

So for example, you have a list of cities: ["Tokyo", "Paris", "London", "Miami", "Paris", "Orlando", "Dubai", "Tokyo", "Paris"] and it includes some duplicated values (Tokyo and Paris), and you want to find out which ones are duplicated. This would be the same for ids, or other values. If you increase the list to hundreds of values, it'd be harder to find it by just looking at the list :)

As the result I'd probably expect the list each duplicated values (["Paris", "Tokyo", "Paris"]) instead of a uniq version of them (["Paris", "Tokyo"]).

I personally do it many times to check data in the database or from other sources (csv, json) to discover duplicated records with the same name, code, or other values, especially while cleaning up legacy data and where there were no previous constraints/checks.

#5 - 02/06/2019 02:20 PM - tad (Tadashi Saito)

How about Set#add? ?

```
require 'set'

a = ["Tokyo", "Paris", "London", "Miami", "Paris", "Orlando", "Dubai", "Tokyo", "Paris"]
s = Set.new
p a.select{|e| !s.add?(e)} #=> ["Paris", "Tokyo", "Paris"]
```

#6 - 02/08/2019 09:19 AM - sawa (Tsuyoshi Sawada)

With the newly introduced tally, you can also do:

```
a = ["Tokyo", "Paris", "London", "Miami", "Paris", "Orlando", "Dubai", "Tokyo", "Paris"]

a.tally(&:itself).flat_map{|k, v| Array.new(v - 1, k)}
#=> ["Tokyo", "Paris", "Paris"]
```

#7 - 02/08/2019 10:00 AM - nobu (Nobuyoshi Nakada)

sawa (Tsuyoshi Sawada) wrote:

```
a.tally(&:itself).flat_map{|k, v| Array.new(v - 1, k)}
```

As tally does not take a block, &:itself is not used.
It's a mistake in the rdoc.