Ruby master - Misc #13840

Collection methods - stability

08/27/2017 02:27 PM - MSP-Greg (Greg L)

Status:	Rejected	
Priority:	Normal	
Assignee:		

Description

I'm trying to fix some method code (Gem::Resolver#search_for) in rubygems.

Regardless, in simplifying the code, I was left with one question regarding all of the sort/filter group methods in ruby.

Which ones are considered stable, or, is original order maintained where applicable?

Obviously, this pertains to sort and sort_by, but also has meaning in group_by, select, reject, and similar methods.

I'm not proposing one or another, although I'd prefer stable. Docs for all these methods should note this, and if stability is guaranteed, tests should verify it. Happy to help.

As an aside, Gem::Resolver#search_for exists in both ruby and rubygems, but is different. Also, a test for it exists in rubygems, but not in ruby. Seems odd.

History

#1 - 08/30/2017 06:37 AM - duerst (Martin Dürst)

MSP-Greg (Greg L) wrote:

> Regardless, in simplifying the code, I was left with one question regarding all of the sort/filter group methods in ruby.
>
> Which ones are considered stable, or, is original order maintained where applicable?

As described below, this seems to lump together two different families of methods.

> Obviously, this pertains to sort and sort_by, but also has meaning in group_by, select, reject, and similar methods.
>
> I'm not proposing one or another, although I'd prefer stable. Docs for all these methods should note this,

The purpose of sorting is to change the order of the elements (except for trivial input), and *stable* refers only to those elements that compare equal under the sort order. For sorting, efficient implementations (quicksort being the most famous one) are not stable, and therefore, "stable or not stable" is a standard question for a sorting algorithm. Array#sort, Array#sort!, Array#sort_by!, Enumerable#sort, and Enumerable#sort_by all are not stable, and all clearly say so in the documentation. I would expect any other sort-like methods in Ruby to also not guarantee stability, and to say so explicitly. If you find one that doesn't, please tell us explicitly.

In contrast, none of the partitioning methods in any way 'intends' to change the order of the elements, and the principle of least astonishment implies that order doesn't change. (Non-parallel) implementations of partitioning methods are straightforward and automatically stable. I have never heard of a non-stable filter/select method, and using the term 'stable' for these kinds of methods isn't customary because it's default/obvious/automatic. That may be the reason why the docs don't say anything for these methods. Many readers would have to think twice what was meant by 'stable' in such a method's context, and might wonder why this is mentioned at all. I would therefore expect all the partitioning methods (group_by, select, reject, partition and friends) to be 'stable' in your sense, without the need for explicit documentation.

> and if stability is guaranteed, tests should verify it. Happy to help.

Tests can't verify. If a method that's intended to be stable got unstable as a result of a bug, it may be good to have a regression test. Also, tests may be a good idea for methods where actual effort is needed to make them stable (such as a stable sort, but Ruby doesn't have one).

> As an aside, Gem::Resolver#search_for exists in both ruby and rubygems, but is different. Also, a test for it exists in rubygems, but not in ruby. Seems odd.

Separate issue, please (if necessary).

#2 - 10/20/2017 02:03 AM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Rejected

(Rejecting for now but please don't hesitate to open a new one for the Gem::Resolver issue if necessary.)

As Martin already pointed out our documents clearly say that sorting methods are not stable.