Ruby - Feature #12810

Improve `Set#find_index` performance

10/05/2016 05:34 PM - Asche (Thomas Charbonnel)

Status:	Rejected	
Priority:	Normal	
Assignee:	knu (Akinori MUSHA)	
Target version:		

Description

Hello everyone!

I've toyed a bit with the Set class lately and have found some performance issues with method find_index. Github gist here:
https://gist.github.com/thomascharbonnel/f023ca137f2b2b7021cbe2d580485cd4

I'm thinking it would be possible to add an index as default value for each new item of Set (instead of a boolean like now), find_index would then be executed in O(1).
I can attach a patch in a few days if everybody is cool with the idea.

Thanks!
Thomas.

History

#1 - 10/11/2016 07:48 PM - herwin (Herwin W)

Quoting the first line of http://ruby-doc.org/stdlib-2.3.1/libdoc/set/rdoc/Set.html:

> Set implements a collection of unordered values

This would mean the set {1,2,3} is exactly the same as the sets {1,3,2}, {2,1,3}, {2,3,1}, {3,1,2} and {3,2,1}. This is a property of the mathematical concept of sets (see https://en.wikipedia.org/wiki/Set_%28mathematics%29, which also describes the order of the items as being irrelevant). So actually, I don't see how using Set#find_index would make any sense. The current implementation just accidentally preserves the order of the items in the set via undocumented behaviour, relying on that would be scary.

#2 - 10/31/2016 04:04 AM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

- Assignee set to knu (Akinori MUSHA)

Seems a design issue. Let me assign this to the library's maintainer.

#3 - 11/05/2016 02:27 AM - knu (Akinori MUSHA)

- Status changed from Assigned to Rejected

As commented by Herwin W and replied to the submitter in a personal mail, Set has no sense of index where elements are theoretically unordered. It's just that a Set happens to respond to find_index via the Enumerable module and have an order due to its implementation.

If you need find_index then Set is not likely the solution for your problem. You should use Hash directly.

Thanks for the feedback anyway!