# Ruby trunk - Feature #4553

## Add Set#pick and Set#pop

04/05/2011 08:10 AM - adgar (Michael Edgar)

| | |
|---|---|
| **Status:** | Rejected |
| **Priority:** | Normal |
| **Assignee:** | knu (Akinori MUSHA) |
| **Target version:** | 2.6 |

**Description**

=begin
A very common operation on sets is to take an arbitrary element from them at O(1) cost. I know typically it's suggested that library additions go out as a gem to see community review. However, to me it seems to be a glaring omission to lack such an operation on a built-in, fundamental data structure, so I've gone straight to the bug tracker.

I wasn't too sure which method names to use as I've often heard "take" in lieu of "pop," so I just used the names Wikipedia uses. Consider them flexible. "Pick" selects an arbitrary element, and "pop" selects and deletes it.

I've provided a very simple patch that implements the necessary behavior. Thoughts?
=end

### History

**#1 - 04/08/2011 03:40 AM - marcandre (Marc-Andre Lafortune)**

*- Priority changed from Normal to 3*

=begin
Hi.

There is no difference between your Set#pick and the existing Set#first. As such, there is no need for pick.

One can get the effect of Set#pop by calling the existing Set#delete with the result of Set#first (with the same O(1) efficiency).

I find problematic that a method called pop would take the first element; it could be renamed to shift, say, or take the last element instead. More importantly, though, I am not convinced that this functionality is that frequently used.
=end

**#2 - 04/12/2011 09:05 AM - adgar (Michael Edgar)**

=begin
The ruby-core e-mail I sent appeared to not attach to the issue on Redmine. I apologize for the noise.

---

The existence of Set#first is an artifact of Set including Enumerable and the very idea of "first" on a Set is an odd notion. A set by itself does not have ordering. I have implemented it to pick the first element because it is simplest, but someone calling "pick" on a set does not care which element it is. If one would prefer implementing it as an alias to first, I'd be fine with that.

On a personal note to emphasize why I find this issue compelling: one reason I enjoy using Ruby because often my code reads just as I would describe it to a colleague - reading "element = set.first" is not something many would agree is a sensible way to describe the act of picking an arbitrary element from a mathematical set.

Yes, one could implement #pop with #delete and #first, which again relies on a notion of ordering in a set. One can implement #proper_subset with #subset and a comparison, as can one implement #proper_superset with #superset and a comparison. Yet they are in the Set class because they are fundamental operations. #add? and #delete? can also be implemented with #add and #include?/#delete and #include? respectively, yet they are included in the Set class because they are useful and common operations on a set.

If pop is too close the Array method of the same name, I understand the confusion, but I don't get why picking first/last makes a difference. Anyone asking to remove an arbitrary element from a Set surely doesn't care where it lies in an underlying hash table. I am surprised one would argue that the Set class rarely needs the notion of selecting an element from it. Rarely am I presented with an algorithm using sets that does not require simply getting an element or picking one out of a set.

=end

**#3 - 04/12/2011 09:16 PM - zimbatm (zimba tm)**

=begin
#pop is often associated to stack operations, which implies an order. Unless a better name is found, isn't set.delete(set.take) enough ?

#take can be an alias of #first but I'm not sure if Enumerable should be included in Set in the first place. In my POV, Enumerable is for elements that have a particular order.

=end

#### #4 - 06/27/2011 03:23 AM - adgar (Michael Edgar)

Any update on this? It seems like a reasonable hole to fill in Ruby 1.9.3.

#### #5 - 03/18/2012 07:26 PM - shyouhei (Shyouhei Urabe)

*- Description updated*

*- Status changed from Open to Assigned*

*- Assignee set to knu (Akinori MUSHA)*

#### #6 - 03/18/2012 09:53 PM - trans (Thomas Sawyer)

I would expect Array#pick to work too. Which then leads one to think Array might support the equivalent of Set#pop as well. But since Array already has #pop it's not really a good idea to reuse same term, although in the case of Set, #pop might be an alias for such method.

So what might such methods be called then? Well, if we look at hand-dandy facets/random:

https://github.com/rubyworks/facets/blob/master/lib/standard/facets/random.rb

Looks like it uses #at_rand/#at_rand! and #pick/pick!, where #pick has the additional option of returning a random subset if a size is given as argument.

#### #7 - 11/20/2012 09:29 PM - mame (Yusuke Endoh)

*- Target version set to 2.6*

#### #8 - 10/22/2017 05:49 AM - knu (Akinori MUSHA)

*- Status changed from Assigned to Rejected*

As marcandre says, this pick method does exactly what first does, and I'm not sure if pop would be an important primitive method.

Array now has sample that picks a random element(s) from an array without deleting it which may or may not be the function you want, but I can't think of a way to randomly pick a key from a hash at O(1) cost, at least with a pure ruby implementation.

### Files

| | | | |
|---|---|---|---|
| pick_pop.diff | 1.06 KB | 04/05/2011 | adgar (Michael Edgar) |