Ruby master - Feature #6589

Set#rehash

06/14/2012 11:54 AM - marcandre (Marc-Andre Lafortune)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>knu (Akinori MUSHA)</td>
</tr>
<tr>
<td>Target version:</td>
<td>2.6</td>
</tr>
</tbody>
</table>

**Description**

There should be a way to rehash a Set.

```ruby
s = Set.new([[]])
s.first << 1
# s.rehash # Does not exist!
s.include? [1] # => false, want true
```

See also:


**Related issues:**

Related to Ruby master - Bug #12970: == Equality of recursive sets fails

**Associated revisions**

Revision 8c90432a - 10/22/2017 12:25 PM - knu (Akinori MUSHA)

Add Set#reset

This method resets the internal state of a set after modification to existing elements, reindexing and deduplicating them. [Feature #6589]

```
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60360 b2dd03c8-39d4-4d8f-98ff-823fe69b080e
```

Revision 60360 - 10/22/2017 12:25 PM - knu (Akinori MUSHA)

Add Set#reset

This method resets the internal state of a set after modification to existing elements, reindexing and deduplicating them. [Feature #6589]

Revision 60360 - 10/22/2017 12:25 PM - knu (Akinori MUSHA)

Add Set#reset

This method resets the internal state of a set after modification to existing elements, reindexing and deduplicating them. [Feature #6589]

Revision 60360 - 10/22/2017 12:25 PM - knu (Akinori MUSHA)

Add Set#reset

This method resets the internal state of a set after modification to existing elements, reindexing and deduplicating them. [Feature #6589]

**History**

#1 - 07/14/2012 06:35 PM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

#2 - 10/25/2012 07:44 PM - yhara (Yutaka HARA)

- Target version changed from 2.0.0 to 2.6

#3 - 11/12/2012 01:12 PM - marcandre (Marc-Andre Lafortune)

Comment about this trivial but needed feature would be appreciated.
Is it specified that Set must be hashtable-based forever? There are alternate ways to implement a Set.

Alternate ways of implementing Set with check/insertion in O(1) that would also work if structures change without a rehash functionality?

In any case, the documentation states that "Set uses Hash as storage", but more importantly that "The equality of each couple of elements is determined according to Object#eql? and Object#hash".

Actually, an undocumented "feature" is that Set does not support an element being modified once it is added.

Maybe we should "clarify" that in the document, or add such a method that recalculates identities of elements. I'm yet to decide which, and the name we could give it.

- rehash (let's be honest)
- reset (re-set the set)
- sync
- ...

I added some notes to the rdoc in r42265.

Add Set#reset

This method resets the internal state of a set after modification to existing elements, reindexing and deduplicating them. [Feature #6589]