**Ruby master - Feature #11090**

**Enumerable#each_uniq and #each_uniq_by**

04/23/2015 07:45 AM - Hanmac (Hans Mackowiak)

<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>nobu (Nobuyoshi Nakada)</td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

currently if you want to iterate the first uniq elements you either need to call uniq and create a big array or you need to count the elements yourself

if you have an Enumerable with an indifferent size (maybe something with cycle or something you can't rewind) calling the Array#uniq method might not what you want.

the idea is adding each_uniq which does only iterate through the elements which are not already send (it does count for you) a second each_uniq_by does work similar with chunk and does takes a block using a generated Enumerator

IceDragon200 did make the following gist/sample in ruby, it might be written in C later to make it faster/better. https://gist.github.com/IceDragon200/5b1c205b4b38665c308e for better view i also added it as attachment.

**Related issues:**

Related to Ruby master - Feature #1153: Enumerable#uniq  
Closed

**Associated revisions**

Revision 42c6a513 - 07/20/2016 08:39 AM - nobu (Nobuyoshi Nakada)

enum.c: Enumerable#uniq

- enum.c (enum_uniq): new method Enumerable#uniq. [Feature #11090]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@55709 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 55709 - 07/20/2016 08:39 AM - nobu (Nobuyoshi Nakada)

enum.c: Enumerable#uniq

- enum.c (enum_uniq): new method Enumerable#uniq. [Feature #11090]

Revision 55709 - 07/20/2016 08:39 AM - nobu (Nobuyoshi Nakada)

enum.c: Enumerable#uniq

- enum.c (enum_uniq): new method Enumerable#uniq. [Feature #11090]

Revision 55709 - 07/20/2016 08:39 AM - nobu (Nobuyoshi Nakada)

enum.c: Enumerable#uniq

- enum.c (enum_uniq): new method Enumerable#uniq. [Feature #11090]

Revision 55709 - 07/20/2016 08:39 AM - nobu (Nobuyoshi Nakada)

enum.c: Enumerable#uniq

- enum.c (enum_uniq): new method Enumerable#uniq. [Feature #11090]

Revision 6882b354 - 07/20/2016 08:44 AM - nobu (Nobuyoshi Nakada)

e numerator.c: Enumerator::Lazy#uniq

- enumerator.c (lazy_uniq): new method Enumerator::Lazy#uniq. [Feature #11090]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@55710 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 55710 - 07/20/2016 08:44 AM - nobu (Nobuyoshi Nakada)

e numerator.c: Enumerator::Lazy#uniq

- enumerator.c (lazy_uniq): new method Enumerator::Lazy#uniq. [Feature #11090]

03/18/2022
enumerator.c (lazy_uniq): new method Enumerator::Lazy#uniq. [Feature #11090]

Revision 55710 - 07/20/2016 08:44 AM - nobu (Nobuyoshi Nakada)
enumerator.c: Enumerator::Lazy#uniq
   enumerator.c (lazy_uniq): new method Enumerator::Lazy#uniq. [Feature #11090]

Revision 55710 - 07/20/2016 08:44 AM - nobu (Nobuyoshi Nakada)
enumerator.c: Enumerator::Lazy#uniq
   enumerator.c (lazy_uniq): new method Enumerator::Lazy#uniq. [Feature #11090]

Revision 55710 - 07/20/2016 08:44 AM - nobu (Nobuyoshi Nakada)
enumerator.c: Enumerator::Lazy#uniq
   enumerator.c (lazy_uniq): new method Enumerator::Lazy#uniq. [Feature #11090]

Revision 39622db0 - 07/20/2016 09:00 PM - nobu (Nobuyoshi Nakada)
NEWS: Enumerable#uniq [ci skip]
   NEWS: mention Enumerable#uniq and Enumerator::Lazy#uniq. [Feature #11090]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@55714 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 55714 - 07/20/2016 09:00 PM - nobu (Nobuyoshi Nakada)
NEWS: Enumerable#uniq [ci skip]
   NEWS: mention Enumerable#uniq and Enumerator::Lazy#uniq. [Feature #11090]

Revision 55714 - 07/20/2016 09:00 PM - nobu (Nobuyoshi Nakada)
NEWS: Enumerable#uniq [ci skip]
   NEWS: mention Enumerable#uniq and Enumerator::Lazy#uniq. [Feature #11090]

Revision 55714 - 07/20/2016 09:00 PM - nobu (Nobuyoshi Nakada)
NEWS: Enumerable#uniq [ci skip]
   NEWS: mention Enumerable#uniq and Enumerator::Lazy#uniq. [Feature #11090]

Revision 55714 - 07/20/2016 09:00 PM - nobu (Nobuyoshi Nakada)
NEWS: Enumerable#uniq [ci skip]
   NEWS: mention Enumerable#uniq and Enumerator::Lazy#uniq. [Feature #11090]

Revision e9f61faa - 07/20/2016 09:15 PM - nobu (Nobuyoshi Nakada)
enum.c: [DOC] Enumerable#uniq [ci skip]
   enum.c (enum_uniq): add rdoc, reference to Array#uniq. [Feature #11090]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@55715 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 55715 - 07/20/2016 09:15 PM - nobu (Nobuyoshi Nakada)
enum.c: [DOC] Enumerable#uniq [ci skip]
   enum.c (enum_uniq): add rdoc, reference to Array#uniq. [Feature #11090]

Revision 55715 - 07/20/2016 09:15 PM - nobu (Nobuyoshi Nakada)
enum.c: [DOC] Enumerable#uniq [ci skip]
   enum.c (enum_uniq): add rdoc, reference to Array#uniq. [Feature #11090]

Revision 55715 - 07/20/2016 09:15 PM - nobu (Nobuyoshi Nakada)
enum.c: [DOC] Enumerable#uniq [ci skip]
   enum.c (enum_uniq): add rdoc, reference to Array#uniq. [Feature #11090]
module Enumerable
  def uniq
    result = []
    uniq_map = {}
    if block_given?
      each do |value|
        key = yield value
        next if uniq_map.has_key?(key)
        uniq_map[key] = true
        result << value
      end
    else
      each do |value|
        next if uniq_map.has_key?(value)
        uniq_map[value] = true
        result << value
      end
    end
    result
  end
end

class Enumerator::Lazy
  def uniq
    uniq_map = {}
    if block_given?
      Enumerator::Lazy.new(self) do |yielder, value|
        key = yield value
        next if uniq_map.has_key?(key)
        uniq_map[key] = true
        yielder << value
      end
    else
      Enumerator::Lazy.new(self) do |yielder, value|
        next if uniq_map.has_key?(value)
        uniq_map[value] = true
        yielder << value
      end
    end
  end
end

olimpics = {1896 => 'Athens', 1900 => 'Paris', 1904 => 'Chikago', 1906 => 'Athens', 1908 => 'Rome'}
each_city_first_time = olimpics.uniq{|k,v | v }
# [{1896, 'Athens'}, [1900, "Paris"], [1904, "Chikago"], [1908, "Rome"]]
{1..Float::INFINITY}.lazy.uniq{|x| (x**2) % 10 }.first(6)
# => [1, 2, 3, 4, 5, 10]

While I propose another solution for the problem, I'm totally agree that we need a way to work with unique elements of collections without creating intermediate array. In heavy data processing it is a very common problem.
As Ilya proposed, Enumerable#uniq and Enumerable::Lazy#uniq is reasonable.

Matz.

- Status changed from Open to Closed

Applied in changeset r55709.

enum.c: Enumerable#uniq
  - enum.c (enum_uniq): new method Enumerable#uniq. [Feature #11090]

- Related to Feature #1153: Enumerable#uniq added

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>each uniq.rb</td>
<td>830 Bytes</td>
<td>04/23/2015</td>
<td>Hanmac (Hans Mackowiak)</td>
</tr>
</tbody>
</table>