Ruby master - Bug #14573

rb_ary_or doesn't check objects hash when the array contains less than SMALL_ARRAY_LEN

03/03/2018 06:58 PM - Willian (Willian van der Velde)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.5.0p0 (2017-12-25 revision 61468) [x86_64-darwin17]	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN

Description

Hello,

While debugging a build failure on a 2.5 branch of graphql-ruby (https://github.com/rmosolgo/graphql-ruby/pull/1303) we found out a difference in behavior for the Array's union operator.

On Ruby 2.5:

```
irb(main):001:0> class X; def initialize(h); @h = h; end; def hash; @h; end; def eql?(o); true;
end; end
=> :eql?
irb(main):002:0> a = X.new(1) ; b = X.new(2)
=> #<X:0x00007fe0b00a4130 @h=2>
irb(main):003:0> a.hash
=> 1
irb(main):004:0> b.hash
=> 2
irb(main):005:0> [a] | [b]
=> [#<X:0x00007fe0b00a4158 @h=1>]
```

I would expect this union would result in two objects. On Ruby 2.3.6 I do get two objects:

```
irb(main):001:0> class X; def initialize(h); @h = h; end; def hash; @h; end; def eql?(o); true;
end; end
=> :eql?
irb(main):002:0> a = X.new(1) ; b = X.new(2)
=> #<X:0x00007f816a80eec0 @h=2>
irb(main):003:0> a.hash
=> 1
irb(main):004:0> b.hash
=> 2
irb(main):005:0> [a] | [b]
=> [#<X:0x00007f816a80eee8 @h=1>, #<X:0x00007f816a80eec0 @h=2>]
```

I think this change is related to #13884, and it looks like when small arrays (less than SMALL_ARRAY_LEN) are given the object's hash isn't checked. If a bigger array is given we get a correct union:

```
irb(main):001:0> class X; def initialize(h); @h = h; end; def hash; @h; end; def eql?(o); true;
end; end
=> :eql?
irb(main):002:0> a = 20.times.map { |i| X.new(i) }
=> ...
irb(main):003:0> b = 20.times.map { |i| X.new(100 + i) }
=> ...
irb(main):004:0> (a | b).size
=> 40
```

Thanks,

Willian

Related issues:

Related to Ruby master - Bug #14263: Array Intersection does not seem to use ...	**Closed**

History

#1 - 03/05/2018 03:44 PM - Willian (Willian van der Velde)

- ruby -v changed from ruby 2.6.0dev (2017-12-27 trunk 61500) [x86_64-darwin17] to ruby 2.5.0p0 (2017-12-25 revision 61468) [x86_64-darwin17]

#2 - 03/14/2018 11:54 PM - tenderlovemaking (Aaron Patterson)

I guess I can see how this is a behavior change, but these two objects shouldn't eql? each other. It seems like a bug in the eql? implementation of the X class. These instances won't deal with hash collisions correctly, so I'm not sure why we'd expect union operator to work too.

#3 - 03/21/2018 03:01 PM - lewispb (Lewis Buckley)

Related to https://bugs.ruby-lang.org/issues/14263. We have posted a note there about our investigation and the inconsistent behaviour between small and larger arrays.

#4 - 06/20/2019 06:09 PM - jeremyevans0 (Jeremy Evans)

- Related to Bug #14263: Array Intersection does not seem to use hash added

#5 - 08/26/2019 05:38 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed

I don't think this is a bug. If eql? returns true, hash must return the same value. Class X in this example violates that constraint, resulting in undefined behavior. If you fix class X, the behavior is correct in all ruby versions since at least 1.8:

```
class X; def initialize(h); @h = h; end; def hash; @h; end; def eql?(o); @h.eql?(o.instance_variable_get(:@h
)); end; end
```