Ruby trunk - Feature #15281

Speed up Set#intersect with size check.

11/03/2018 01:08 PM - RGBD (Oleg Zubchenko)

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

Description

Current implementation computes set intersection s1 & s2 in O(s2.size) time.
It can be reduced to O([s1.size, s2.size].min) time.

Additional speedup comes from using #each instead of #do_with_enum.

See files attached for benchmarks.

Pull Request

P.S. using benchmark-ips gem

History

#1 - 11/03/2018 01:09 PM - RGBD (Oleg Zubchenko)

- Description updated

#2 - 11/03/2018 01:10 PM - RGBD (Oleg Zubchenko)

- Description updated

#3 - 11/03/2018 01:19 PM - RGBD (Oleg Zubchenko)

- File benchmark_results.txt added

#4 - 11/03/2018 03:51 PM - marcandre (Marc-Andre Lafortune)

Thanks for the patch.

Sadly, I don't think we can do that as Sets are ordered. That optimization would change the order of the resulting Set in some cases.

#5 - 11/03/2018 05:10 PM - RGBD (Oleg Zubchenko)

marcandre (Marc-Andre Lafortune) wrote:

> Thanks for the patch.
>
> Sadly, I don't think we can do that as Sets are ordered. That optimization would change the order of the resulting Set in some cases.

Where in the documentation can I read that?

this says order is uncertain. I thought it means noone should make assumptions about it.
Top of the page states:

> Set implements a collection of unordered values with no duplicates. This is a hybrid of Array's intuitive inter-operation facilities and Hash's fast lookup.

#6 - 11/03/2018 06:42 PM - marcandre (Marc-Andre Lafortune)

- Assignee set to matz (Yukihiro Matsumoto)

RGBD (Oleg Zubchenko) wrote:

> > Set implements a collection of unordered values with no duplicates. This is a hybrid of Array's intuitive inter-operation facilities and Hash's fast lookup.

Good point.

Note that this documentation is 15 years old and predates Hash being ordered.

I'd be enclined to say that Set should be officially ordered, even if "mathematically" speaking that doesn't make sense. I'm assigning this to Matz.

#7 - 11/04/2018 12:08 AM - duerst (Martin Dürst)

marcandre (Marc-Andre Lafortune) wrote:

> I'd be enclined to say that Set should be officially ordered, even if "mathematically" speaking that doesn't make sense. I'm assigning this to Matz.

I'd be inclined to say that Set should be officially UNordered, because they are mathematically unordered.

#8 - 11/06/2018 08:35 PM - ahorek (Pavel Rosický)

> we can do that as Sets are ordered

IMO if they are, it's an implementation detail that you shouldn't rely on. Also there's more room to optimize unordered sets.

if you need an ordered (or indexed?) set, it should be a subclass that implements this behaviour on the top of the generic unordered set. Something like https://ruby-doc.org/stdlib-2.5.3/libdoc/set/rdoc/SortedSet.html

#9 - 04/27/2019 08:01 PM - Eregon (Benoit Daloze)

I think making Set unordered would be a big breaking change, similar to making Hash unordered (as it was in 1.8).
Maybe people forgot, but I find it pretty bad to work with unordered Hashes,
e.g., every iteration method like each yields a confusing order changing on potentially every mutation.

I'm also fairly confident there is code out there relying on Set ordering.

This proposal doesn't propose to make Set unordered though, but it would introduce non-determinism in the order of elements returned by Set#&.
That seems not ideal.

Also, this trivial optimization can already be done at the application level if needed, and there the change of ordering is obvious and conscious:

```
set1, set2 = ...
if set1.size < set2.size
  set1 & set2
else
  set2 & set1
end
```

So I don't think it's worth introducing non-determinism/changing order in Set for something that can be expressed so simply in application code, with the advantage of the ordering trade-off being clear if done in application code.

#10 - 05/11/2019 01:40 PM - nobu (Nobuyoshi Nakada)

- Assignee changed from matz (Yukihiro Matsumoto) to knu (Akinori MUSHA)

The author of set.rb is knu.

#11 - 05/11/2019 01:40 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Assigned

#12 - 05/21/2019 02:52 AM - knu (Akinori MUSHA)

It is clearly stated that Set is an unordered collection and it doesn't even guarantee any determinacy about the order of elements, and as you know, it's just it happened to become ordered and deterministic when the underlying data structure (Hash) got an order.

That being said, I can understand the need for an ordered set if there are already some users using Set as such. We could probably introduce OrderedSet for easier migration before making this "breaking" change (and maybe some others to follow) to Set, I guess?

#13 - 05/21/2019 08:25 AM - sawa (Tsuyoshi Sawada)

There is a mathematical term "ordered set", which means a different thing: an algebra in which the elements are given intrinsic order.

The concept in question should be named Tuple or UniqueTuple.

#14 - 05/21/2019 09:42 AM - chrisseaton (Chris Seaton)

I'm not sure it should be called Tuple - to me that implies that elements can appear more than once - that it's a multi-set - and this isn't the case.

#15 - 05/21/2019 01:09 PM - Eregon (Benoit Daloze)

Just a note, for compatibility there is often a gap between what the documentation states and what Ruby programs assume.
So even though the documentation says it's unordered, I would guess there are programs relying on Sets being ordered since Ruby 1.9.

Maybe not so many programs, but I think we should evaluate before changing.

Files

intersect.rb	1.91 KB	11/03/2018	RGBD (Oleg Zubchenko)
intersect_standalone.rb	671 Bytes	11/03/2018	RGBD (Oleg Zubchenko)
benchmark_results.txt	3.68 KB	11/03/2018	RGBD (Oleg Zubchenko)