Ruby - Bug #15807

Range#minmax is slow and never returns for endless ranges

04/28/2019 12:14 PM - janosch-x (Janosch Müller)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	2.6.3p62	**Backport:**	2.4: UNKNOWN, 2.5: UNKNOWN, 2.6: UNKNOWN

Description

current situation:

- (1..).minmax runs forever
- (1..).max raises "cannot get the maximum of endless range"
- (1..Float::INFINITY).minmax runs forever
- (1..Float::INFINITY).max returns instantly
- (1..1_000_000_000).minmax takes one minute
- (1..1_000_000_000).max returns instantly

my suggestion:

- implement minmax in range.c, return [range_min, range_max]
- for endless ranges, this will trigger the same error as max does
- delegate to enum (rb_call_super) only if called with a block (?)

i could perhaps provide a PR if you can point me to some information on how to contribute.

cheers!

Related issues:

Related to Ruby - Bug #15867: Enumerable#minmax inconsistent behavior with Ti...	**Closed**

Associated revisions

Revision d5c60214c45bafc1cf2a516f852394986f9c84bb - 08/14/2019 06:22 PM - jeremyevans (Jeremy Evans)

Implement Range#minmax

Range#minmax was previous not implemented, so calling #minmax on
range was actually calling Enumerable#minmax. This is a simple
implementation of #minmax by just calling range_min and range_max.

Fixes [Bug #15867]
Fixes [Bug #15807]

Revision d5c60214c45bafc1cf2a516f852394986f9c84bb - 08/14/2019 06:22 PM - jeremyevans (Jeremy Evans)

Implement Range#minmax

Range#minmax was previous not implemented, so calling #minmax on
range was actually calling Enumerable#minmax. This is a simple
implementation of #minmax by just calling range_min and range_max.

Fixes [Bug #15867]
Fixes [Bug #15807]

Revision d5c60214 - 08/14/2019 06:22 PM - jeremyevans (Jeremy Evans)

Implement Range#minmax

Range#minmax was previous not implemented, so calling #minmax on
range was actually calling Enumerable#minmax. This is a simple
implementation of #minmax by just calling range_min and range_max.

Fixes [Bug #15867]
Fixes [Bug #15807]

History

#1 - 04/28/2019 12:32 PM - mame (Yusuke Endoh)

I'm never against fixing this issue but I have one concern. Currently, Range#max is not consistent with Enumerable#minmax.

```
p ("a".."aa").max    #=> "aa"
p ("a".."aa").minmax #=> ["a", "z"]
```

Thus, if Range#minmax is simply implemented, it will make it consistent. This is not always good because it means that the fix brings incompatibility.

Do you have any use case? Or, are you facing any concrete issue that is caused by this inconsistency? If so, we can believe that it would be worth fixing this issue. If not, we need to consider.

#2 - 04/28/2019 02:29 PM - janosch-x (Janosch Müller)

mame (Yusuke Endoh) wrote:

> Range#max is not consistent with Enumerable#minmax.

Thanks for pointing this out, I wasn't aware of that. Floats are another example:

```
(1..(5.5)).max # => 5.5
(1..(5.5)).minmax # => [1, 5]
```

Maybe we could fix / speedup minmax only for ranges where begin and end are NIL_P/is_integer_p/Float::INFINITY, and call super for all other cases?

A check for Float::INFINITY might be helpful here, too, while we're at it:
https://github.com/ruby/ruby/blob/ae6c195f30f76b1dc4a32a0a91d35fe80f6f85d3/range.c#L808

My use case has to do with Regexp quantification. I can go into more detail, but to describe it quickly, I want to provide information about how many chars a Regexp can match in https://github.com/ammar/regexp_parser. Ranges, some ending with Infinity, are the most natural choice for this, but minmax would be useful in the related code. Also, I don't want to hand out "dangerous" Ranges to gem users. Maybe I will #extend the Ranges with a safe minmax.

#3 - 05/22/2019 10:28 PM - nobu (Nobuyoshi Nakada)

- Related to Bug #15867: Enumerable#minmax inconsistent behavior with Time ranges added

#4 - 05/31/2019 07:31 PM - mohsen (Mohsen Alizadeh)

i had the same problem with minmax in Range.

I made a MR

https://github.com/ruby/ruby/pull/2216

#5 - 06/08/2019 05:24 AM - jeremyevans0 (Jeremy Evans)

- File range-minmax.patch added

I think this is a bug we should fix, even if it breaks code relying on this bug ("all bug fixes are incompatibilities" :)).

I worked on a patch for #15867, before realizing mohsen created a pull request for this. My patch ended up being very similar to mohsen's, it is attached. The main differences are that my patch calls range_min/range_max C functions directly instead of calling min and max using rb_funcall, and mohsen's patch includes tests and specs, and mine only tests.

Does anyone have an opinion on whether minmax should call overridden min and max methods? Or do we expect if you override min or max, you should also override minmax? I don't have a strong opinion either way.

#6 - 06/15/2019 09:50 AM - janosch-x (Janosch Müller)

jeremyevans0 (Jeremy Evans) wrote:

> I think this is a bug we should fix, even if it breaks code relying on this bug ("all bug fixes are incompatibilities" :)).

Yes, it is a bit reminiscent of https://xkcd.com/1172/ :)

> Does anyone have an opinion on whether minmax should call overridden min and max methods? Or do we expect if you override min or max, you should also override minmax? I don't have a strong opinion either way.

Other classes including Enumerable also require minmax to be overridden individually. Set or SortedSet are examples from the stdlib. So I guess it

would be more consistent to call the C functions.

#7 - 06/16/2019 01:26 PM - janosch-x (Janosch Müller)

Thinking about this a bit more generally, I'm wondering whether Enumerable#minmax should actually use rb_funcall to get min and max.

This would fix the problem described in this issue.

But perhaps more importantly, it eliminates the trap of forgetting to implement #minmax whenever you override #min and/or #max.

Right now, #minmax is effectively broken for every case where #min and/or #max is overridden to optimize performance, to use custom checks, or to return a custom value -- unless #minmax is also overridden with def minmax; [min, max] end.

As Range shows, arguably even ruby core coders have fallen into this trap ...

#8 - 06/18/2019 02:08 PM - Eregon (Benoit Daloze)

I think it would make sense for Enumerable#minmax to just be [min, max], and be overridden on some classes if useful (probably rare).

#9 - 08/14/2019 06:23 PM - jeremyevans (Jeremy Evans)

- Status changed from Open to Closed

Applied in changeset git|d5c60214c45bafc1cf2a516f852394986f9c84bb.

Implement Range#minmax

Range#minmax was previous not implemented, so calling #minmax on
range was actually calling Enumerable#minmax. This is a simple
implementation of #minmax by just calling range_min and range_max.

Fixes [Bug #15867]
Fixes [Bug #15807]

Files

range-minmax.patch	2.89 KB	06/08/2019	jeremyevans0 (Jeremy Evans)