Ruby - Feature #19839

Need a method to check if two ranges overlap

08/18/2023 01:30 AM - shouichi (Shouichi Kamiya)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		

Description

It would be convenient to have a method that checks if two ranges overlap. For example,

```
(0..10).overlap?(5..15) #=> true
(0..10).overlap?(20..30) #=> false
```

Related issues:

Related to Ruby - Feature #13933: Add Range#empty?	**Rejected**
Related to Ruby - Feature #15976: Add Array#overlap? for whether the intersec...	**Closed**

Associated revisions

Revision e9b503f1bb9692eda1d1f55f62c19d861b88a0d5 - 09/16/2023 05:57 AM - shouichi (Shouichi Kamiya)

[Feature #19839] Add Range#overlap?

Add a method that returns true if two range overlap, otherwise false.

```
(0..10).overlap?(5..15) #=> true
(0..10).overlap?(20..30) #=> false
```

Revision e9b503f1bb9692eda1d1f55f62c19d861b88a0d5 - 09/16/2023 05:57 AM - shouichi (Shouichi Kamiya)

[Feature #19839] Add Range#overlap?

Add a method that returns true if two range overlap, otherwise false.

```
(0..10).overlap?(5..15) #=> true
(0..10).overlap?(20..30) #=> false
```

Revision e9b503f1 - 09/16/2023 05:57 AM - shouichi (Shouichi Kamiya)

[Feature #19839] Add Range#overlap?

Add a method that returns true if two range overlap, otherwise false.

```
(0..10).overlap?(5..15) #=> true
(0..10).overlap?(20..30) #=> false
```

Revision b4213a73b807cf8c8884e29d37308c46ca80352a - 09/16/2023 08:24 AM - nobu (Nobuyoshi Nakada)

[Feature #19839] Fix Range#overlap? for empty ranges

Empty ranges do not overlap with any range.

Regarding benchmarks, PR#8242 is significantly faster in some cases,
but one of these two cases is a wrong result.

	ActiveSupport	PR#8242	built-ruby
(2..3).overlap?(1..1)	7.761M	15.053M	32.368M
	-	1.94x	4.17x
(2..3).overlap?(2..4)	25.720M	55.070M	21.981M
	1.17x	2.51x	-
(2..3).overlap?(4..5)	7.616M	15.048M	21.730M
	-	1.98x	2.85x
(2..3).overlap?(2..1)	25.585M	56.545M	32.786M

	ActiveSupport	PR#8242	built-ruby
	-	2.21x	1.28x
(2..3).overlap?(0..1)	7.554M	14.755M	32.545M
	-	1.95x	4.31x
(2..3).overlap?(...1)	6.681M	5.843M	32.255M
	1.14x	-	5.52x
(2...3).overlap?(..2)	6.676M	5.817M	21.572M
	1.15x	-	3.71x
(2...3).overlap?(3...)	7.392M	14.755M	31.805M
	-	2.00x	4.30x
(2..3).overlap?('a'..'d')	3.675M	3.482M	17.009M
	1.06x	-	4.89x

Revision b4213a73b807cf8c8884e29d37308c46ca80352a - 09/16/2023 08:24 AM - nobu (Nobuyoshi Nakada)

[Feature #19839] Fix Range#overlap? for empty ranges

Empty ranges do not overlap with any range.

Regarding benchmarks, PR#8242 is significantly faster in some cases,
but one of these two cases is a wrong result.

	ActiveSupport	PR#8242	built-ruby
(2..3).overlap?(1..1)	7.761M	15.053M	32.368M
	-	1.94x	4.17x
(2..3).overlap?(2..4)	25.720M	55.070M	21.981M
	1.17x	2.51x	-
(2..3).overlap?(4..5)	7.616M	15.048M	21.730M
	-	1.98x	2.85x
(2..3).overlap?(2..1)	25.585M	56.545M	32.786M
	-	2.21x	1.28x
(2..3).overlap?(0..1)	7.554M	14.755M	32.545M
	-	1.95x	4.31x
(2..3).overlap?(...1)	6.681M	5.843M	32.255M
	1.14x	-	5.52x
(2...3).overlap?(..2)	6.676M	5.817M	21.572M
	1.15x	-	3.71x
(2...3).overlap?(3...)	7.392M	14.755M	31.805M
	-	2.00x	4.30x
(2..3).overlap?('a'..'d')	3.675M	3.482M	17.009M
	1.06x	-	4.89x

Revision b4213a73 - 09/16/2023 08:24 AM - nobu (Nobuyoshi Nakada)

[Feature #19839] Fix Range#overlap? for empty ranges

Empty ranges do not overlap with any range.

Regarding benchmarks, PR#8242 is significantly faster in some cases,
but one of these two cases is a wrong result.

	ActiveSupport	PR#8242	built-ruby
(2..3).overlap?(1..1)	7.761M	15.053M	32.368M
	-	1.94x	4.17x
(2..3).overlap?(2..4)	25.720M	55.070M	21.981M

	ActiveSupport	PR#8242	built-ruby
	1.17x	2.51x	-
(2..3).overlap?(4..5)	7.616M	15.048M	21.730M
	-	1.98x	2.85x
(2..3).overlap?(2..1)	25.585M	56.545M	32.786M
	-	2.21x	1.28x
(2..3).overlap?(0..1)	7.554M	14.755M	32.545M
	-	1.95x	4.31x
(2..3).overlap?(...1)	6.681M	5.843M	32.255M
	1.14x	-	5.52x
(2...3).overlap?(..2)	6.676M	5.817M	21.572M
	1.15x	-	3.71x
(2...3).overlap?(3...)	7.392M	14.755M	31.805M
	-	2.00x	4.30x
(2..3).overlap?('a'..'d')	3.675M	3.482M	17.009M
	1.06x	-	4.89x

History

#1 - 08/18/2023 02:09 AM - baweaver (Brandon Weaver)

I've made several helpers for this exact same problem, as well as a Range#merge in the past. I would very much be in favor of this change.

Example usage from an interview problem, though I have some internal usecases which have done similar:

https://gist.github.com/baweaver/5dbca4296db0651de41267c2c1267c68

The idea for this was to bold targeted segments of a String like:

```
Emboldener.new(text: "aaabbcc", targets: ["aaa","aab","bc"]).run
```

...and have intersections merged. Granted this also brings up an interesting follow-up potential of merging ranges and if they happen to be left or right biased:

```
private def merge_range(a, b)
  return Range.new(a.begin, b.end) if a.cover?(b.begin)
  return Range.new(b.begin, a.end) if b.cover?(a.begin)

  nil
end
```

In any case the overlap? method would be very handy to me for a number of use-cases.

#2 - 08/18/2023 04:24 AM - nobu (Nobuyoshi Nakada)

What do you expect when it is called with a non-Range object, TypeError, same as cover? or something else?

#3 - 08/18/2023 06:56 AM - shouichi (Shouichi Kamiya)

Thank you for your feedback.

> I've made several helpers for this exact same problem, as well as a Range#merge in the past.

Yes, Range class can have methods such as Range#merge but I wanted to start small and see what happens.

> What do you expect when it is called with a non-Range object, TypeError, same as cover? or something else?

Yes. Raising TypeError seems correct. I'm going to fix the PR.

#4 - 08/18/2023 05:05 PM - Dan0042 (Daniel DeLorme)

Related to #16757

#5 - 09/12/2023 07:53 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Open to Feedback

ActiveSupport already have overlaps? method.

- https://api.rubyonrails.org/classes/Range.html#method-i-overlaps-3F
- https://github.com/rails/rails/blob/main/activesupport/lib/active_support/core_ext/range/overlap.rb#L7

Is it enough for your use-case?

#6 - 09/12/2023 08:54 AM - shouichi (Shouichi Kamiya)

Though ActiveSupport's overlaps? method is enough for our use-case. I think the feature is so fundamental that it should be supported by the standard library.

Slightly off-topic: It might be better to add a method that returns the overlapping range of two ranges. Then users can check if the overlapping range is empty or not. Although there's no method to check if a range is empty.

#7 - 09/12/2023 09:31 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Feedback to Open

Thanks. Basically, we didn't add new methods from ActiveSupport without performance reason.

Can you describe why we should add overlap? to ruby core?

#8 - 09/13/2023 05:19 AM - shouichi (Shouichi Kamiya)

Honestly, I can't give a formal reason. But I do believe it's a very basic operation. For example, postgres and boost provide such functionality.

- https://www.postgresql.org/docs/15/functions-range.html
- https://www.boost.org/doc/libs/1_83_0/libs/numeric/interval/doc/interval.htm

#9 - 09/13/2023 05:58 AM - baweaver (Brandon Weaver)

I believe there are a few reasons for this addition. The core one I see often in justifications is precedence in existing Ruby, such as:

Combinations

These cases have clear precedent for iterable / enumerable types:

- Array#concat - Merging two Arrays
- Hash#merge - Merging two Hashes
- Set#union - Merging two Sets (or Enumerable on other end)

Inclusions

Interestingly not as common are patterns like Enumerable#include?(other_enumerable) versus the very common Enumerable#include?(single_item) so we have a lot of "unnamed" approximations:

- Array - (array_two - array_one).empty? or array_one.intersection(array_two).any?
- Hash - (hash_two.keys - hash_one.keys).empty?

...which are not very performant, and I would bet are common patterns in non-Rails code. It also raises a potential for Hash#intersection and how that might work with same keys with different values.

Overlaps

For these two cases I would also almost consider overlaps?, which is synonymous to intersection(other).any? from Array:

```
[1, 2, 3].overlaps?([2, 3]) # true
{ a: 1, b: 2 }.overlaps?(a: 3, b: 5) # true? key vs value is complicated here

(1..10).overlaps?(5..15) # true
```

Point being I would argue there is a (minor) performance win potential here for existing code, it feels very much in the spirit of Ruby of creating a name for a common operation.

#10 - 09/14/2023 08:17 AM - matz (Yukihiro Matsumoto)

Accepted. It must be useful without ActiveSupport.

Matz.

#11 - 09/14/2023 08:36 AM - ko1 (Koichi Sasada)

I'm not sure what (n...n) means but on AS definition, the following code return true.

```
p (1..2).overlap?(2...2)  #=> true
p (2..2).overlap?(2...2)  #=> true
p (2...2).overlap?(2...2) #=> true
```

is it intentional?

#12 - 09/14/2023 08:53 AM - matz (Yukihiro Matsumoto)

Although I said “accepted”, we found some corner cases which are not clear (e.g. #note-11). We have to make these cases clear before merging it to the core.

Matz.

#13 - 09/14/2023 09:49 AM - sawa (Tsuyoshi Sawada)

I think the analogue for range1.overlap?(range2) in mathematics is interval1 ∩ interval2 ≠ ∅, which does not hold when either interval1 or interval2 is an empty interval.

Hence, I think that, when either range1 or range2 is an empty range (e.g., 2...2, 2..1), range1.overlap?(range2) should return false.

Instead of adopting the AS #note-5 definition,

```
def overlap?(other)
  other.begin == self.begin || cover?(other.begin) || other.cover?(self.begin)
end
```

I propose to define it as equivalent to:

```
def overlap?(other)
  return false if none? or other.none?
  other.begin == self.begin || cover?(other.begin) || other.cover?(self.begin)
end
```

#14 - 09/14/2023 01:48 PM - nobu (Nobuyoshi Nakada)

https://github.com/nobu/ruby/tree/Range%23overlap_p

#15 - 09/15/2023 08:46 AM - shouichi (Shouichi Kamiya)

I opened a PR about a month ago but should I close it...? https://github.com/ruby/ruby/pull/8242

#16 - 09/16/2023 05:57 AM - shouichi (Shouichi Kamiya)

- Status changed from Open to Closed

Applied in changeset git|e9b503f1bb9692eda1d1f55f62c19d861b88a0d5.

[Feature #19839] Add Range#overlap?

Add a method that returns true if two range overlap, otherwise false.

```
(0..10).overlap?(5..15) #=> true
(0..10).overlap?(20..30) #=> false
```

#17 - 09/16/2023 08:07 AM - mame (Yusuke Endoh)

@nobu (Nobuyoshi Nakada) is trying to fix the corner cases that @ko1 (Koichi Sasada) pointed: https://github.com/ruby/ruby/pull/8448

#18 - 09/18/2023 01:55 PM - akr (Akira Tanaka)

I found another corner case.

```
% ./ruby -e 'r = (...-Float::INFINITY); p r.overlap?(r)'
true
% ./ruby -e 'r = (...[]); p r.overlap?(r)'
true
% ./ruby -e 'r = (...""); p r.overlap?(r)'
true
% ./ruby -e 'r = (...true); p r.overlap?(r)'
true
```

```
% ./ruby -e 'r = (...false); p r.overlap?(r)'
true
% ./ruby -e 'r = (...Object.new); p r.overlap?(r)'
true
% ./ruby -v
ruby 3.3.0dev (2023-09-16T22:27:04Z master cd67c0d204) [x86_64-linux]
```

They are (...MINIMUM) where MINIMUM is a value that there is no value less than that.

(...MINIMUM) is empty because
(1) it doesn't contain MINIMUM itself because ... is used (exclude_end is true) and
(2) it doesn't contain other values because there is no value less than MINIMUM.

But (...MINIMUM).overlap?(...MINIMUM) returns true as shown above.
It means Ruby says there is a value contained in (...MINIMUM).
It is invalid.

Note that true, false, and Object.new has <=> method defined by Kernel.
It determines self is equal to itself and other objects are not comparable.
Thus, they can be considered as minimum values (and maximum values).

I think this problem is difficult to solve.
We have no clean way to detect minimum values.

Some options:
(1) introduce some protocol to detect minimum values such as minimum? method.
(2) hard code minimum values of builtin classes in Range#overlap? and use minimum? for user-defined classes.
(3) document this limitation.

#19 - 09/19/2023 12:31 AM - shouichi (Shouichi Kamiya)

Shouldn't none? handle empty ranges? Currently, it raises an error.

```
> (...-Float::INFINITY).none?
(irb):1:in `each': can't iterate from NilClass (TypeError)
```

If none? handles empty ranges, then overlap? can return false when one of the ranges is empty.

#20 - 09/19/2023 05:59 AM - nobu (Nobuyoshi Nakada)

shouichi (Shouichi Kamiya) wrote in #note-19:

> Shouldn't none? handle empty ranges? Currently, it raises an error.
>
> ```
> > (...-Float::INFINITY).none?
> (irb):1:in `each': can't iterate from NilClass (TypeError)
> ```
>
> If none? handles empty ranges, then overlap? can return false when one of the ranges is empty.

It's for nil, that means begin-less range.

#21 - 09/19/2023 09:34 AM - shouichi (Shouichi Kamiya)

Because (...-Float::INFINITY) is semantically empty, shouldn't none? return true?

#22 - 09/19/2023 04:25 PM - akr (Akira Tanaka)

Range has two semantics: succ-based and cover-based.

none? is succ-based because it is implemented in Enumerable which uses each method which uses succ in Range.

overlap? is cover-based.
It uses comparison (<=>).

The two semantics sometimes cause inconsistent behavior.
It is difficult to make them consistent completely.

#23 - 09/20/2023 12:51 PM - Dan0042 (Daniel DeLorme)

shouichi (Shouichi Kamiya) wrote in #note-21:

> Because (...-Float::INFINITY) is semantically empty, shouldn't none? return true?

I agree it should be semantically empty, but (...-Float::INFINITY).size == Infinity

akr (Akira Tanaka) wrote in #note-22:

> Range has two semantics: succ-based and cover-based.
> none? is succ-based because it is implemented in Enumerable which uses each method which uses succ in Range.

In that case it seems like Range should have cover-based #empty? method. It has been proposed before (#13933) but strangely it was never added?

#24 - 09/21/2023 02:13 AM - mame (Yusuke Endoh)

This method is going to have an incompatibility with Range#overlap? that ActiveSupport has been provided. Is it OK? I'd like to confirm with the Rails developers just to be sure.

#25 - 09/21/2023 06:10 AM - shouichi (Shouichi Kamiya)

Asked them in their discord channel. https://discord.com/channels/849034466856665118/974005005768069211/1154298190120624138

#26 - 09/21/2023 05:43 PM - rafaelfranca (Rafael França)

Thank you for checking with us. I believe the difference when the ranges are empty is acceptable. I don't think it was intentional in the Active Support implementation.

#27 - 10/13/2023 03:29 AM - mame (Yusuke Endoh)

Just FYI: Documentation has been added at c23b25f75f6180b7428f9650e063b1e50fc161e2 for the corner cases of minimum value.

#28 - 10/13/2023 03:29 AM - mame (Yusuke Endoh)

- Related to Feature #13933: Add Range#empty? added

#29 - 10/25/2023 04:56 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #15976: Add Array#overlap? for whether the intersection of 2 arrays is non empty? added