Ruby - Feature #12612

Switch Range#=== to use cover? instead of include?

07/22/2016 12:54 PM - zverok (Victor Shepelev)

Status:	Rejected	
Priority:	Normal	
Assignee:		
Target version:		

Description

Currently, Range#=== is an alias of #include?, which works by #each-ing all the values and comparing them to the checked value. It may lead to fascinating inefficiencies, when working with strings or other similar classes (for example, IP class for allowing/disallowing IP ranges). Moreover, it is not consistent with behavior of numerical ranges (which is kinda special case, I guess):

```
(1...3) === 2.5 # true - though (1..3).to_a.include?(2.5) is false
('a'...'z') === 'foo' # false - because ('a'..'z').to_a.include?('foo') is false
```

As === is heavily used in case and grep, and there is no option to replace it manually with cover? there, maybe it would be reasonable to change the behavior?

To be honest, I could imagine no real cases when include? is preferable for identity check.

Related issues:

Related to Ruby - Feature #12996: Optimize Range#===	**Closed**
Has duplicate Ruby - Feature #14575: Switch Range#=== to use cover? instead o...	**Closed**

History

#1 - 09/24/2016 08:57 AM - zverok (Victor Shepelev)

Did anybody saw this?..

#2 - 11/25/2016 08:51 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Rejected

I see no real-world use-case for Range#=== with strings.
Besides that, using cover? behavior for string ranges would introduce incompatibility.

Matz.

#3 - 11/25/2016 09:11 AM - zverok (Victor Shepelev)

I encountered the problem in production code, it was not strings, but specialized IPAddress class, to filter "if IP is in range" -- it was SUDDENLY 0.4 sec on each request to server spent just to (IPAddress.new(from)..IPAddress.new(to)).to_a in "innocently" looking:

```
case ip
when admin_ip_range
...
```

Another real cases:

- ranges of Dates/Times/DateTimes
- Measure units (like https://github.com/joshwlewis/unitwise)
- Any other specialized *enumerable* AND *comparable* value class

Yes, proposed behavior is incompatible with current, but I am not sure it is a bad thing:

```
(Date.parse('2016-05-01')..Date.today) === DateTime.parse('2016-06-01 12:30')
# => false
(Date.parse('2016-05-01')..Date.today).cover?(DateTime.parse('2016-06-01 12:30'))
# => true

# Which is more logical? What you'll intuitively expect here:
case DateTime.parse('2016-06-01 12:30')
when Date.parse('2016-05-01')..Date.today
  # looks like we SHOULD be here, but...
```

#4 - 12/20/2016 03:34 AM - shyouhei (Shyouhei Urabe)

- Related to Feature #12996: Optimize Range#=== added

#5 - 05/17/2018 07:07 AM - matz (Yukihiro Matsumoto)

- Related to Feature #14575: Switch Range#=== to use cover? instead of include? added

#6 - 05/17/2018 07:07 AM - matz (Yukihiro Matsumoto)

- Related to deleted (Feature #14575: Switch Range#=== to use cover? instead of include?)

#7 - 05/17/2018 07:08 AM - matz (Yukihiro Matsumoto)

- Has duplicate Feature #14575: Switch Range#=== to use cover? instead of include? added