Ruby master - Bug #6155

Enumerable::Lazy#flat_map raises an exception when an element does not respond to #each

03/16/2012 02:53 PM - dkubb (Dan Kubb)

Status: Closed
Priority: Normal
Assignee: shugo (Shugo Maeda)
Target version: 2.0.0
ruby -v: ruby 2.0.0dev (2012-03-15 trunk 35028) [x86_64-darwin11.3.0]

Backport:

Description
The following statement will raise "NoMethodError: undefined method `each' for 1:Fixnum":

[1, 2, 3].lazy.flat_map { |n| n }.to_a

It appears as if Enumerable::Lazy#flat_map is calling #each on every element, regardless of whether it can work or not.

As a reference, the equivalent statement using Enumerable#flat_map works:

[1, 2, 3].flat_map { |n| n }.to_a

Associated revisions
Revision ded27bf5 - 03/19/2012 08:22 AM - shugo (Shugo Maeda)

- enumerator.c (lazy_flat_map_func): convert the block value to Array if it doesn't respond to each. [ruby-core:43334] [Bug #6155]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@35092 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 35092 - 03/19/2012 08:22 AM - shugo (Shugo Maeda)

- enumerator.c (lazy_flat_map_func): convert the block value to Array if it doesn't respond to each. [ruby-core:43334] [Bug #6155]

Revision 35092 - 03/19/2012 08:22 AM - shugo (Shugo Maeda)

- enumerator.c (lazy_flat_map_func): convert the block value to Array if it doesn't respond to each. [ruby-core:43334] [Bug #6155]

Revision 35092 - 03/19/2012 08:22 AM - shugo (Shugo Maeda)

- enumerator.c (lazy_flat_map_func): convert the block value to Array if it doesn't respond to each. [ruby-core:43334] [Bug #6155]

Revision 35092 - 03/19/2012 08:22 AM - shugo (Shugo Maeda)

- enumerator.c (lazy_flat_map_func): convert the block value to Array if it doesn't respond to each. [ruby-core:43334] [Bug #6155]

Revision 35092 - 03/19/2012 08:22 AM - shugo (Shugo Maeda)

- enumerator.c (lazy_flat_map_func): convert the block value to Array if it doesn't respond to each. [ruby-core:43334] [Bug #6155]

Revision 35092 - 03/19/2012 08:22 AM - shugo (Shugo Maeda)

- enumerator.c (lazy_flat_map_func): convert the block value to Array if it doesn't respond to each. [ruby-core:43334] [Bug #6155]

Revision 35092 - 03/19/2012 08:22 AM - shugo (Shugo Maeda)

- enumerator.c (lazy_flat_map_func): convert the block value to Array if it doesn't respond to each. [ruby-core:43334] [Bug #6155]

Revision 35092 - 03/19/2012 08:22 AM - shugo (Shugo Maeda)

- enumerator.c (lazy_flat_map_func): convert the block value to Array if it doesn't respond to each. [ruby-core:43334] [Bug #6155]

History
#1 - 03/16/2012 03:19 PM - shugo (Shugo Maeda)

- Assignee set to shugo (Shugo Maeda)

Hello,
Dan Kubb wrote:

The following statement will raise "NoMethodError: undefined method `each' for 1:Fixnum":

```
[1, 2, 3].lazy.flat_map { |n| n }.to_a
```

It appears as if Enumerable::Lazy#flat_map is calling #each on every element, regardless of whether it can work or not.

As a reference, the equivalent statement using Enumerable#flat_map works:

```
[1, 2, 3].flat_map { |n| n }.to_a
```

I doubt that this behavior of Enumerable#flat_map is reasonable. 
flat_map is designed to concatenate multiple collections into one collection, so in the above example, map should be used instead of flat_map.

```
[1, 2, 3].concat(4) raises a TypeError, so [1, 2, 3].flat_map { |n| n } should raise an error, I think.
```

#2 - 03/16/2012 03:19 PM - shugo (Shugo Maeda)
- Status changed from Open to Assigned

#3 - 03/17/2012 12:48 AM - dkubb (Dan Kubb)

```
[1, 2, 3].flat_map { |n| n }.to_a
```

I doubt that this behavior of Enumerable#flat_map is reasonable.

I was writing rubyspec for Enumerable::Lazy#flat_map had the same behaviour as Enumerable#flat_map (besides the obvious differences in return value). Here is what I was using as a basis: [https://github.com/rubyspec/rubyspec/blob/master/core/enumerable/shared/collect_concat.rb](https://github.com/rubyspec/rubyspec/blob/master/core/enumerable/shared/collect_concat.rb)

The last example doesn't apply, since it should raise an exception when no block is provided (according to the current implementation), but the first 3 specs fail because they have mixed data, they are equivalent to:

```
[1, [2, 3], [4, [5, 6]], {:foo => :bar}].lazy.flat_map { |e| e }.to_a
[1, [], 2].lazy.flat_map { |e| e }.to_a
[:foo, Object.new.tap { |o| class << o; def to_a() end end }].lazy.flat_map { |e| e }.to_a
```

All of the above examples work when the enumerable is not lazy.

```
 flat_map is designed to concatenate multiple collections into one collection, so in the above example, map should be used instead of flat_map.
```

I was trying to demonstrate the simplest example that would cause an exception like the rubyspec examples but I probably wouldn't use it myself except for multiple collections.

#4 - 03/17/2012 06:42 AM - marcandre (Marc-Andre Lafortune)
- Category set to core
- Target version set to 2.0.0

Hi,

Shugo Maeda wrote:

```
I doubt that this behavior of Enumerable#flat_map is reasonable. 
flat_map is designed to concatenate multiple collections into one collection, so in the above example, map should be used instead of flat_map.
```

```
[1, 2, 3].concat(4) raises a TypeError, so [1, 2, 3].flat_map { |n| n } should raise an error, I think.
```

I understand your point of view. This behavior was clearly intended in Matz's original commit, though (r25456). Changing this could also introduce some compatibility issues.

I feel it might be best to keep the current behavior, but Matz will have the final word on this.

#5 - 03/17/2012 10:23 AM - matz (Yukihiro Matsumoto)

Hi,
In message "Re: [ruby-core:43357] [ruby-trunk - Bug #6155] Enumerable::Lazy#flat_map raises an exception when an element does not respond to #each" on Sat, 17 Mar 2012 06:42:22 +0900, Marc-Andre Lafortune ruby-core@marc-andre.ca writes:

|> [1, 2, 3].concat(4) raises a TypeError, so [1, 2, 3].flat_map { |n| n } should raise an error, I think.
| I understand your point of view. This behavior was clearly intended in Matz's original commit, though (r25456). Changing this could also introduce some compatibility issues.
| I feel it might be best to keep the current behavior, but Matz will have the final word on this.

I vote on keeping the current behavior.

matz.

#6 - 03/19/2012 04:57 PM - shugo (Shugo Maeda)
I'll fix lazy flat_map respecting Matz's opinion, but let me clarify one point.
dkubb (Dan Kubb) wrote:

```
[1, 2, 3].flat_map { |n| n }.to_a
```

I doubt that this behavior of Enumerable#flat_map is reasonable.

I was writing rubyspec for Enumerable::Lazy#flat_map had the same behaviour as Enumerable#flat_map (besides the obvious differences in return value). Here is what I was using as a basis: https://github.com/rubyspec/rubyspec/blob/master/core/enumerable/shared/collect_concat.rb

The last example doesn't apply, since it should raise an exception when no block is provided (according to the current implementation), but the first 3 specs fail because they have mixed data, they are equivalent to:

```
[1, [2, 3], [4, [5, 6]], {foo => :bar}].lazy.flat_map { |e| e }.to_a
[1, [[], 2]].lazy.flat_map { |e| e }.to_a
[[foo], Object.new.tap { |o| class << o; def to_a() end end }].lazy.flat_map { |e| e }.to_a
```

All of the above examples work when the enumerable is not lazy.

Why the last example defines to_a? flat_map doesn't call to_a, but to_ary.

```
p Object.new.tap { |o|
class << o
def to_a; [to_a] end
end
]}.flat_map { |e| e } #=> [Object:0x21aae8a0]
p Object.new.tap { |o|
class << o
def to_ary; [:to_ary] end
end
]}.flat_map { |e| e } #=> [:to_ary]
```

#7 - 03/19/2012 05:22 PM - shugo (Shugo Maeda)
- Status changed from Assigned to Closed
- % Done changed from 0 to 100

This issue was solved with changeset r35092.
Dan, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- enumerator.c (lazy_flat_map_func): convert the block value to Array if it doesn't respond to each. [ruby-core:43334] [Bug #6155]

#8 - 03/20/2012 02:27 AM - marcandre (Marc-Andre Lafortune)

Hi,
shugo (Shugo Maeda) wrote:

```
Why the last example defines to_a? flat_map doesn't call to_a, but to_ary.
```

08/07/2021
I think it's to spec that it doesn't call to_a.

#9 - 03/21/2012 08:53 AM - shugo (Shugo Maeda)

Hello,

Ah, I see. RubySpec defines to_a to assert that it's never called, right? That makes sense.

2012/03/20 2:27 "marcandre (Marc-Andre Lafortune)" <ruby-core@marc-andre.ca>

: Issue #6155 has been updated by marcandre (Marc-Andre Lafortune).

Hi,

shugo (Shugo Maeda) wrote:

Why the last example defines to_a? flat_map doesn't call to_a, but to_ary.

I think it's to spec that it doesn't call to_a.

Bug #6155: Enumerable::Lazy#flat_map raises an exception when an element does not respond to #each
https://bugs.ruby-lang.org/issues/6155#change-24948

Author: dkubb (Dan Kubb)
Status: Closed
Priority: Normal
Assignee: shugo (Shugo Maeda)
Category: core
Target version: 2.0.0
ruby -v: ruby 2.0.0dev (2012-03-15 trunk 35028) [x86_64-darwin11.3.0]

The following statement will raise "NoMethodError: undefined method 'each' for 1:Fixnum":

[1, 2, 3].lazy.flat_map { |n| n }.to_a

It appears as if Enumerable::Lazy#flat_map is calling #each on every element, regardless of whether it can work or not.

As a reference, the equivalent statement using Enumerable#flat_map works:

[1, 2, 3].flat_map { |n| n }.to_a

--

http://bugs.ruby-lang.org/