Ruby master - Bug #18627
segmentation fault when doing a lot of redundant Module#include

03/12/2022 07:39 AM - Ethan (Ethan -)

**Status:** Closed  
**Priority:** Normal  
**Assignee:**  
**Target version:**  
**ruby -v:** ruby 3.1.1p18 (2022-02-18 revision 53f5fc4236) [x86_64-linux]  
**Backport:** 2.6: DONTNEED, 2.7: DONTNEED, 3.0: REQUIRED, 3.1: REQUIRED

**Description**
I'm adding support for ruby 3 and consistently encountering segfaults.

my library does a fair bit of extending objects with modules in an #initialize. I instantiate objects corresponding to nodes in a JSON document. each one extends itself with several modules, depending on its role in the document. some of these dynamically create a module, include some other modules into that module, and then extend themself with that module.

at some point (when seems nondeterministic, but the code path is consistent), ruby segfaults while including from a module (which was dynamically created in #initialize) with another module. this happens on 3.0 and 3.1. it doesn't seem to on 3.2.0-dev, my tests pass fine there.

I'm not sure how much to try to explain the code - it is a fairly complex, and massively inefficient. in investigating this issue I identified a way to go from $O(n^2)$ calls to the segfaulting code path down to $O(1)$, and it didn't segfault anymore. I also realized that, in the unoptimized version, almost all of the calls to the segfaulting code path (all but the $O(1)$) are redundant - i'm calling Module#include when the receiver module already includes the argument module. adding a check to skip the redundant Module#include stopped segfaulting in the inefficient version, too.

so, it looks like Module#include is somehow segfaulting when it should be doing a noop, skipping inclusion of an argument module because its reciever already includes that module.

I've made some attempt at minimally reproducing this, but haven't made it as far as separating it from the rest of the library. so my steps to reproduce at the moment involve cloning my library on the branch where I've added ruby 3 support (branch splat+msim - commit 2a719a23) and invoking that:

```bash
git clone -b splat+msim https://github.com/notEthan/jsi.git
ruby -Ijsi/lib -rjsi -e 'JSI::JSONSchemaOrgDraft06.new_schema({items: {items: {items: {items: {items: {}}}}}})'
```

it may a little bit intermittent - that consistently segfaults for me; shallower depths of the object passed to new_schema are less consistent. in case it does not segfault on another computer, adding further depth should trigger it.

attached are outputs of that with segfault backtrace on 3.0 and 3.1.

finally, here is a high level description of what is occurring when segfault occurs, to hopefully give some idea of the context. I can explain any part in more detail if it is helpful.

- JSI::MetaschemaNode#initialize - called an excessive number of times (hundreds to low thousands)
  - extends this JSI::MetaschemaNode with 1-3 modules (namely: JSI::PathedHashNode, JSI::PathedArrayNode, JSI::Metaschema)
  - for certain nodes:
    - dynamically creates a new module (named jsi_schema_module)
    - calls #include on this jsi_schema_module with 1+ other modules (named metaschema_instance_modules)
      - this is the part that segfaults
      - almost all of these include calls are redundant and should be noop
      - may be relevant that this module metaschema_instance_module includes, directly and indirectly, some 39 other modules
  - instantiates zero or more other JSI::MetaschemaNode instances
  - extends this MetaschemaNode with one or more other modules

Associated revisions
Revision 97426e15 - 03/18/2022 01:19 PM - peterzhu2118 (Peter Zhu)
[Bug #18627] Fix crash when including module

During lazy sweeping, the iclass could be a dead object that has not yet been swept. However, the chain of superclasses of the iclass could already have been swept (and become a new object), which would cause a crash when trying to read the object.

History

#1 - 03/14/2022 10:14 PM - jeremyevans0 (Jeremy Evans)
- Status changed from Open to Feedback

I think the best way to address this would be to take your existing code that segfaults in earlier versions, and bisect commits between 3.1.0 and 3.2.0 to determine the commit that fixes it. Once you have found the commit that fixes the segfaults, we can determine if it makes sense to backport that commit to earlier versions.

#2 - 03/15/2022 01:57 AM - mame (Yusuke Endoh)
I applied git bisect and it said:

```
98fb0ab60eb14e74a48920bd904a3edd4ba52eb is the first bad commit
commit 98fb0ab60eb14e74a48920bd904a3edd4ba52eb
Author: Peter Zhu <peter@peterzhu.ca>
Date:   Tue Jan 11 15:21:56 2022 -0500

Enable Variable Width Allocation by default
```

I guess the commit didn't fix the issue but just "hid" it.

#3 - 03/16/2022 07:53 PM - peterzhu2118 (Peter Zhu)
I debugged this and have a fix here: https://github.com/ruby/ruby/pull/5671

#4 - 03/16/2022 07:56 PM - peterzhu2118 (Peter Zhu)
- Status changed from Feedback to Open

#5 - 03/18/2022 01:19 PM - peterzhu2118 (Peter Zhu)
- Backport changed from 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN to 2.6: DONTNEED, 2.7: DONTNEED, 3.0: REQUIRED, 3.1: REQUIRED

#6 - 03/18/2022 01:19 PM - peterzhu2118 (Peter Zhu)
- Status changed from Open to Closed

Applied in changeset git|97426e15d721119738a548ecfa7232b1d027cd34.

[Bug #18627] Fix crash when including module

During lazy sweeping, the iclass could be a dead object that has not yet been swept. However, the chain of superclasses of the iclass could already have been swept (and become a new object), which would cause a crash when trying to read the object.

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>git_2a719a_ruby_3.0.2p107-b.log</td>
<td>88.7 KB</td>
<td>03/12/2022</td>
<td>Ethan (Ethan -)</td>
</tr>
<tr>
<td>git_2a719a_ruby_3.1p18-a.log</td>
<td>67.7 KB</td>
<td>03/12/2022</td>
<td>Ethan (Ethan -)</td>
</tr>
</tbody>
</table>