Ruby master - Misc #15925

Speed up SortedSet#min, #max, #sum etc.?

06/15/2019 11:10 AM - janosch-x (Janosch Müller)

Status:	Assigned	
Priority:	Normal	
Assignee:	knu (Akinori MUSHA)	

Description

this issue is somewhat similar to https://bugs.ruby-lang.org/issues/15807

current situation, using the example of SortedSet#min (without rbtree):

- SortedSet#min calls Enumerable#min
- Enumerable#min calls SortedSet#each
- SortedSet#each calls SortedSet#to_a
- #to_a returns an Array which is guaranteed to be sorted
- Enumerable#min wastefully goes through this whole Array anyway

so complexity can be reduced from O(n) to O(1) for #min/#max/#minmax.

other methods may be sped up by delegating to faster implementations on Array.

for instance, SortedSet.new(1..1000).to_a.sum is an order of magnitude faster than SortedSet.new(1..1000).sum.

suggestion:

```
class SortedSet < Set
  # [ ... ]
  # non-rbtree case
  # [ ... ]

  def min
    to_a.first
  end

  def max
    to_a.last
  end

  def minmax
    [min, max]
  end

  def sum
    to_a.sum
  end

  # maybe more?
end
```

History

#1 - 06/17/2019 09:29 PM - jeremyevans0 (Jeremy Evans)

- Assignee set to knu (Akinori MUSHA)

- Status changed from Open to Assigned

- File sorted-set-min-max-sum.patch added

Your recommended implementation greatly improves performance. From the benchmark in the attached patch:

```
Calculating -------------------------------------
                     ../ruby2/run_ruby    run_ruby
                min            4.811k      1.166M i/s -     14.501k times in 3.014412s 0.012441s
```

```
                max            4.761k      1.215M i/s -     14.187k times in 2.979777s 0.011680s
             minmax            4.530k    321.515k i/s -     13.505k times in 2.980916s 0.042004s
                sum            5.924k    113.139k i/s -     17.646k times in 2.978920s 0.155968s

Comparison:
                            min
            run_ruby:   1165552.1 i/s
   ../ruby2/run_ruby:      4810.6 i/s - 242.29x  slower

                            max
            run_ruby:   1214686.2 i/s
   ../ruby2/run_ruby:      4761.1 i/s - 255.13x  slower

                         minmax
            run_ruby:    321514.7 i/s
   ../ruby2/run_ruby:      4530.5 i/s - 70.97x  slower

                            sum
            run_ruby:    113138.6 i/s
   ../ruby2/run_ruby:      5923.6 i/s - 19.10x  slower
```

#2 - 06/18/2019 08:39 AM - janosch-x (Janosch Müller)

jeremyevans0 (Jeremy Evans) wrote:

> Your recommended implementation greatly improves performance.

I fear this benchmark result "over-idealizes" the performance improvements because SortedSet#to_a is memoized here on the first run, and only cleared when set contents are changed.

Real world performance improvements should end up between 2x (at most) for a fresh or modified set (by virtue of avoiding the extra iteration mentioned in the report), and the results of this benchmark.

Files

sorted-set-min-max-sum.patch	1.85 KB	06/17/2019	jeremyevans0 (Jeremy Evans)