the hamster gem

“Efficient, immutable, and thread-safe collection classes for Ruby.”

“Efficient, immutable, and thread-safe collection classes for Ruby.”

“Efficient, immutable, and thread-safe collection classes for Ruby.”

containers for objects

tuple

```
(1, 2, 3)
```

set

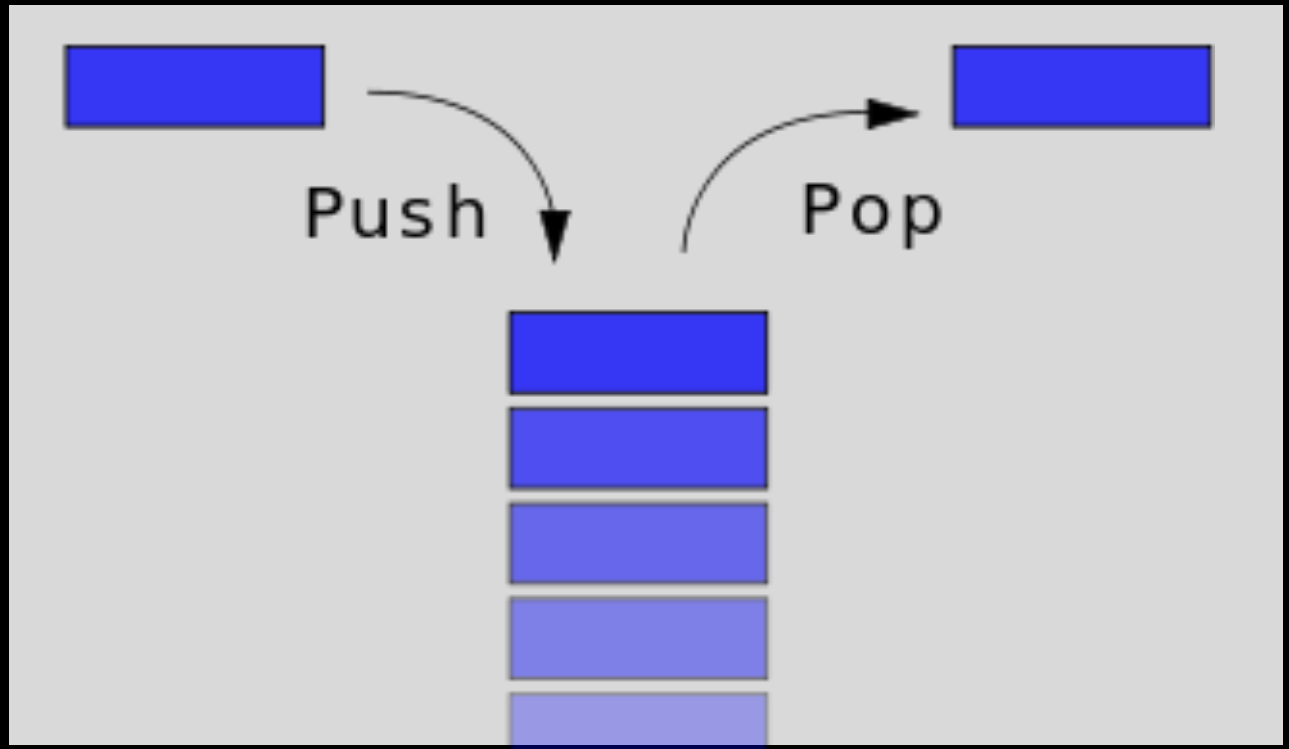
hash

```
{:name => "James", :gender => :male}
```

vector (like array)

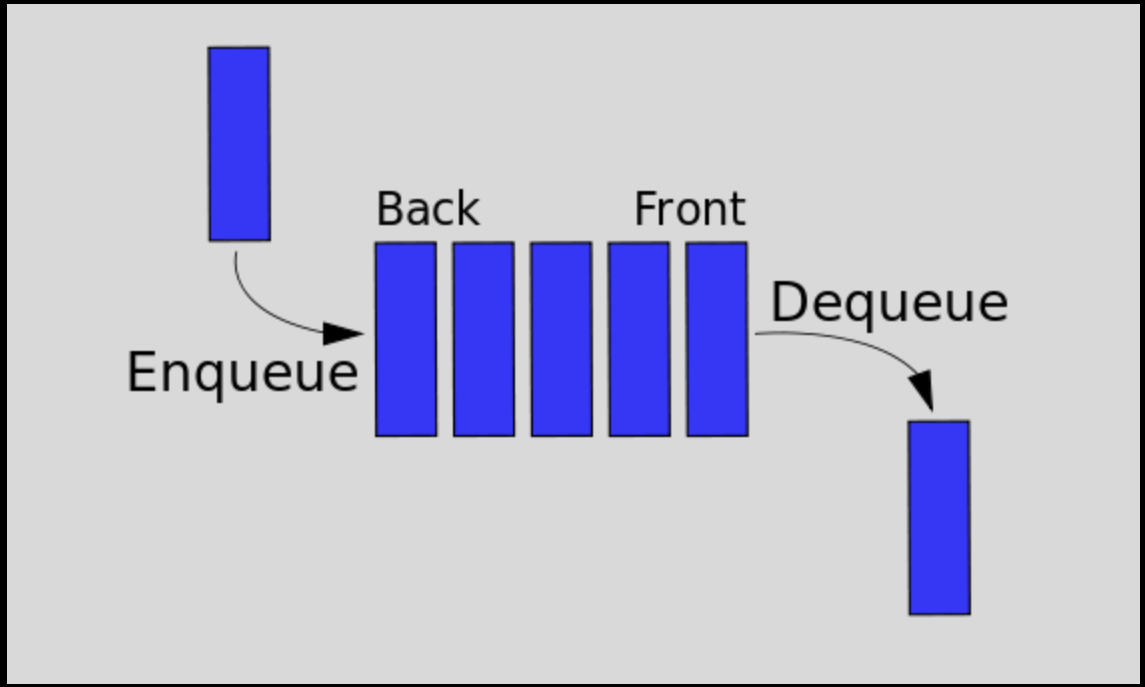
[1, 2, 3, 4, 5]

stack



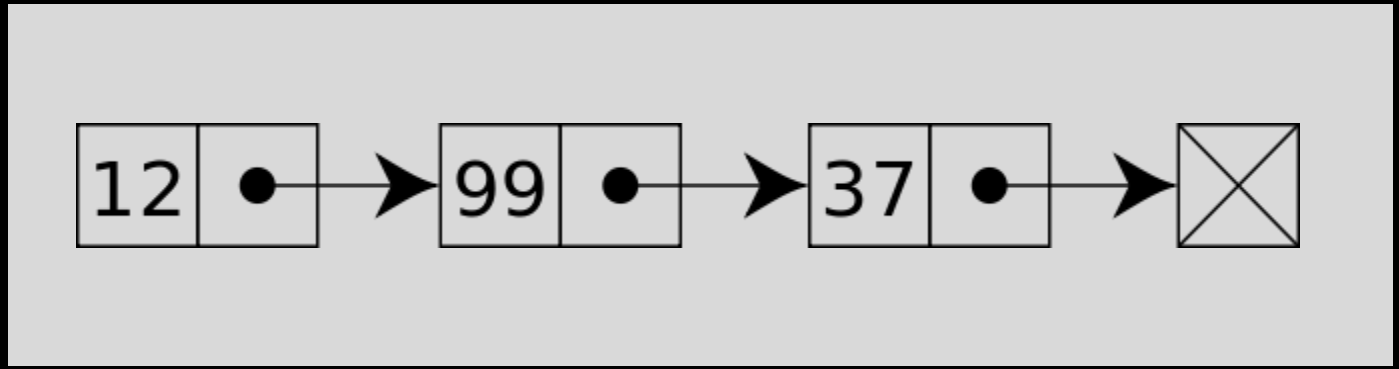
User:Boivie http://en.wikipedia.org/wiki/Stack_(abstract_data_type)#mediaviewer/File:Data_stack.svg

queue



User:Vegpuff on wikipedia http://en.wikipedia.org/wiki/Queue_(abstract_data_type)#mediaviewer/File:Data_Queue.svg

(linked) list



“Efficient, immutable, and thread-safe collection classes for Ruby.”

immutability

```ruby
require "hamster"
person = Hamster.hash(:name => "Simon", :gender => :male)
person[:name]
  # => "Simon"
friend = person.put(:name, "James")
  # => {:name => "James", :gender => :male}
male = person.delete(:name)
  # => {:gender => :male}
person
  # => {:name => "Simon", :gender => :male}
male.key?(:name)
  # => false
```

the sales pitch

immutability is a good thing™

“Mutable stateful objects are the new spaghetti code:
- hard to understand, test, reason about
- Concurrency disaster

Rich Hickey - Clojure

> Immutable objects are simple. Classes should be immutable unless there's a very good reason to make them mutable. If a class cannot be made immutable, limit its mutability as much as possible.
>
> Josh Bloch, in Effective Java

you are using immutable data

git,
bitcoin blockchain,
:ruby

benefits

readability

concurrency (thread-safety)

“Efficient, immutable, and thread-safe collection classes for Ruby.”

```
person = Hamster.hash(
    :name => "Simon",
    :gender => :male,
    #... 1000 other key-value pairs
    )
person[:name]
  # => "Simon"
friend = person.put(:name, "James")
  # => {:name => "James", :gender => :male, #...}
```

Does friend have to copy everything from person?

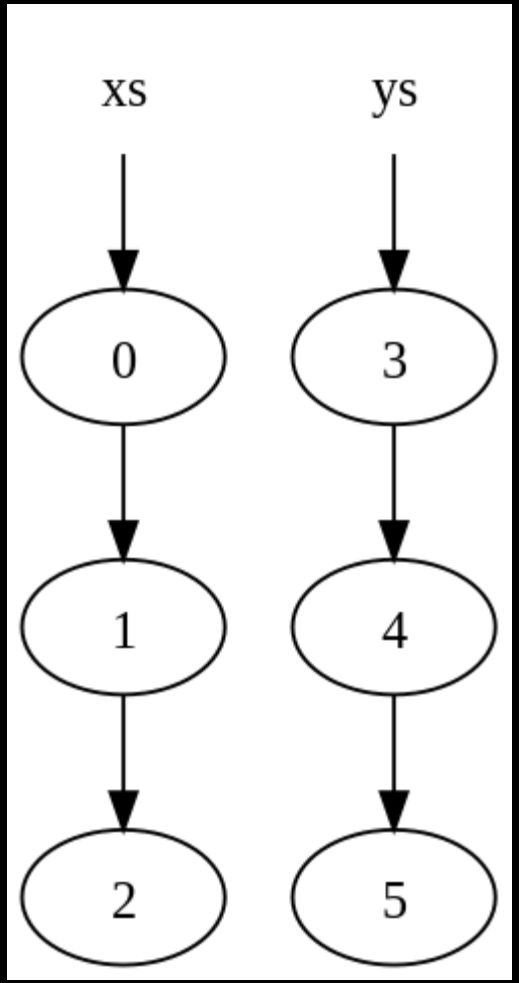
NO!

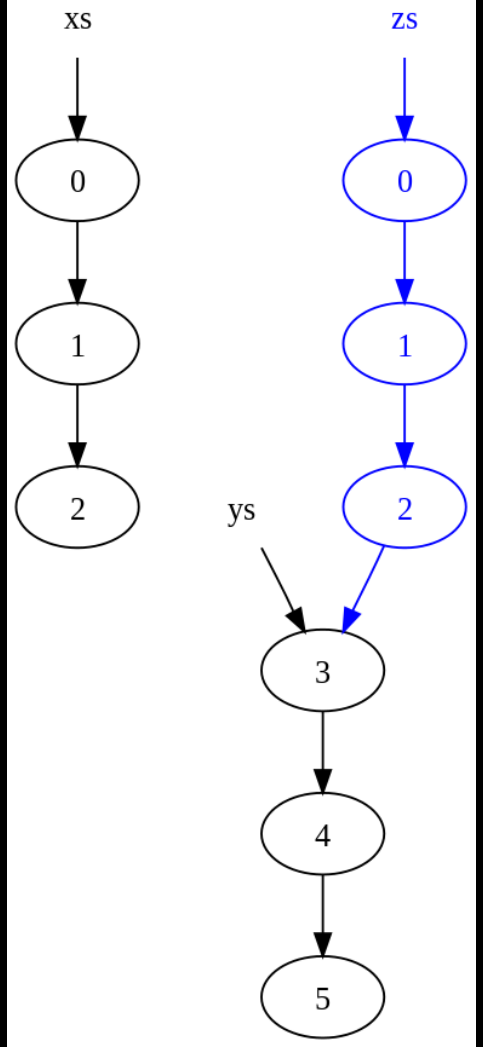
NO!

"Efficient, immutable, and thread-safe collection classes for Ruby."

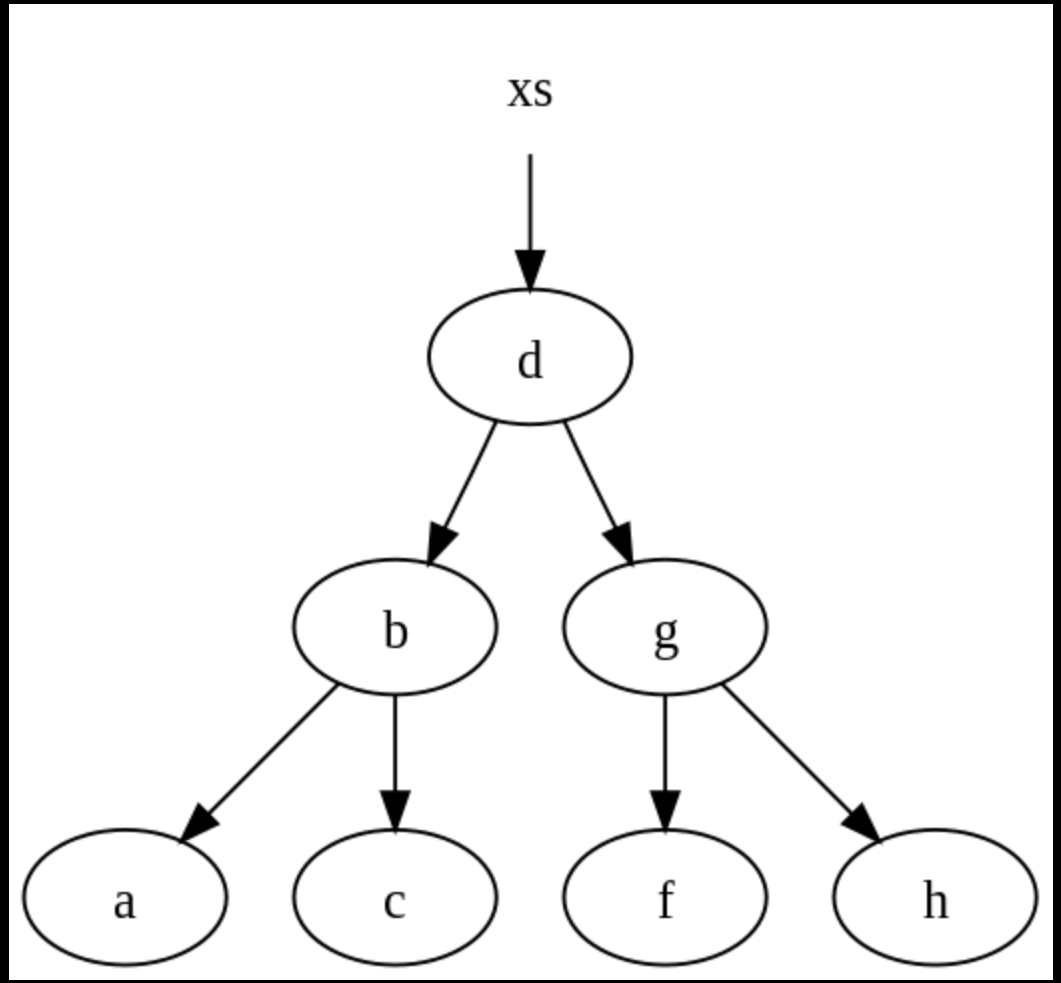
NO!

"Persistent, immutable, and thread-safe collection classes for Ruby."

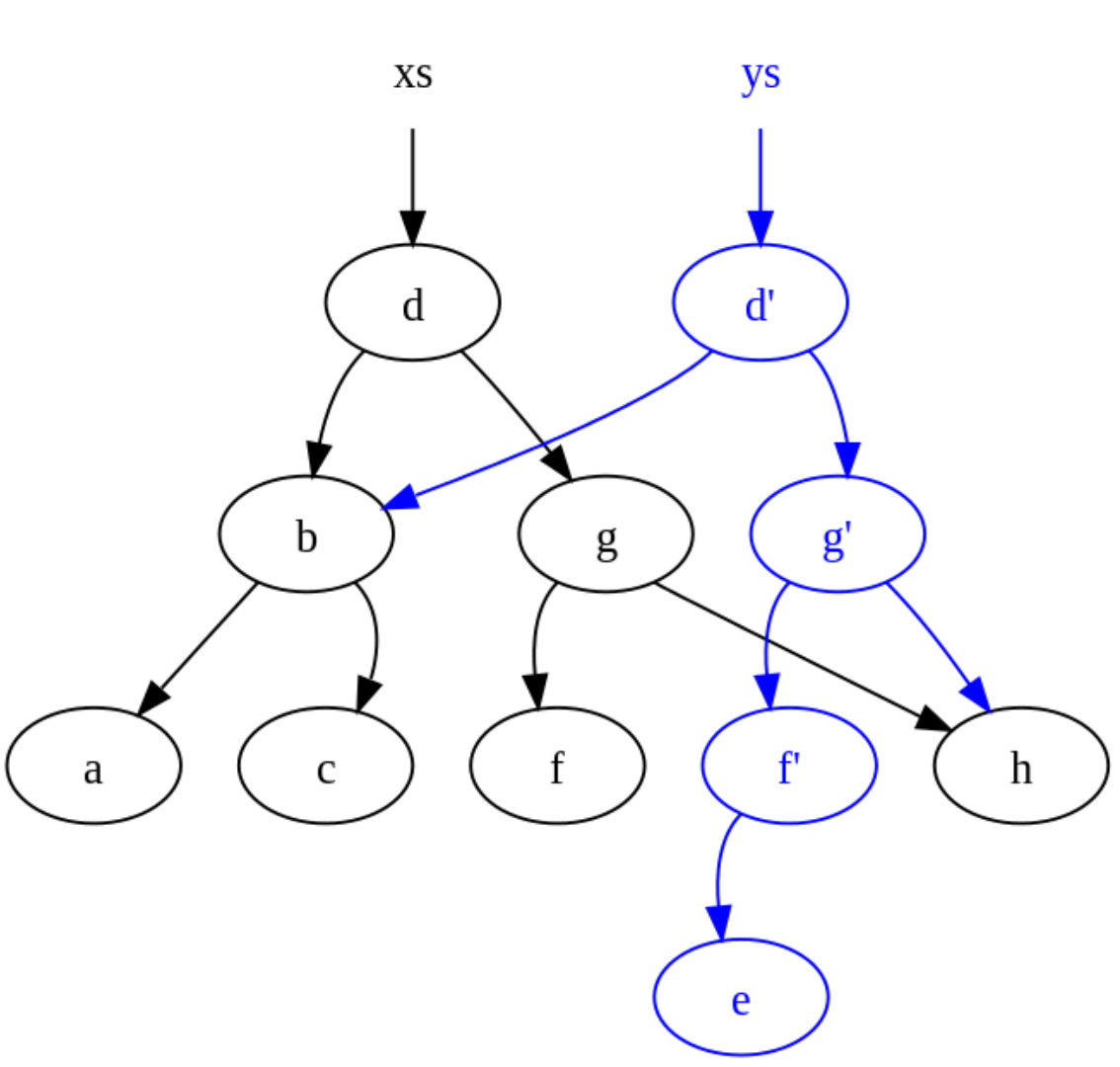




zs = xs + ys


xs
d
b
g
a
c
f
h

adding e



“Efficient, immutable, and thread-safe collection classes for Ruby.”

immutability

=>

thread safety

warning: references immutable but contents might be mutable

```ruby
require 'hamster'
v = "A"
h = Hamster.hash(:k=>v) #=>{:k => "A"}
v[0] = "B"
h #=> {:k => "B"}
```

“Efficient, immutable, and thread-safe collection classes for Ruby.”

references

https://deveo.com/blog/2013/03/22/immutability-in-ruby-part-1/

https://deveo.com/blog/2013/03/28/immutability-in-ruby-part-2/

http://clojure.org/state

http://en.wikipedia.org/wiki/Persistent_data_structure

https://github.com/hamstergem/hamster

slides

https://github.com/ryantm/lvrug_hamster