```
enum.size  # => int, nil or Float::INFINITY
```

Example behavior

```
perm = (1..100).to_a.permutation(4)
perm.size                 # => 94109400
perm.each_cons(2).size    # => 94109399
loop.size                 # => Float::INFINITY
[42].drop_while.size      # => nil
```

It's lazy
It's fast
It's cool
Almost all implemented too

Creating sized enumerators

```
def foo
  unless block_given?
    to_enum(:foo){ calculate_size }
  end
  # ...
end
```

or...

```
Enumerator.new(->{ calculate_size }){ ... }
```

Some use cases:

- libraries can get the size without iterating (e.g. a library to print progression)
- estimating time remaining for algorithms
- speed up #to_a by allocating the right size?

`enum#size`: always lazy, `enum#count`: never lazy

Return useful values: 65+ methods

Return *nil:* {take|drop}_while, {r}index, find{_index}, all IO

Feature #6636