# Lecture 11

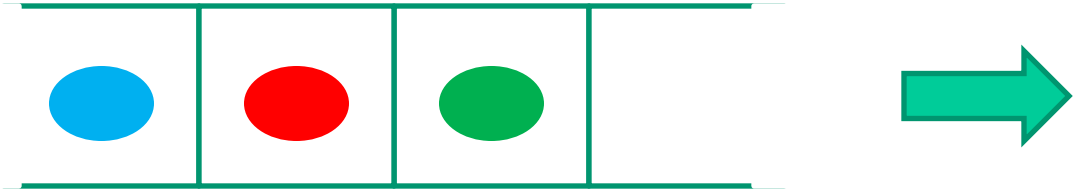
- Administration

# Correctness of Concurrent Objects

- Quiescent Consistency
- Sequential Consistency
- Linearizability

# Concurrent objects

Consider a FIFO queue:

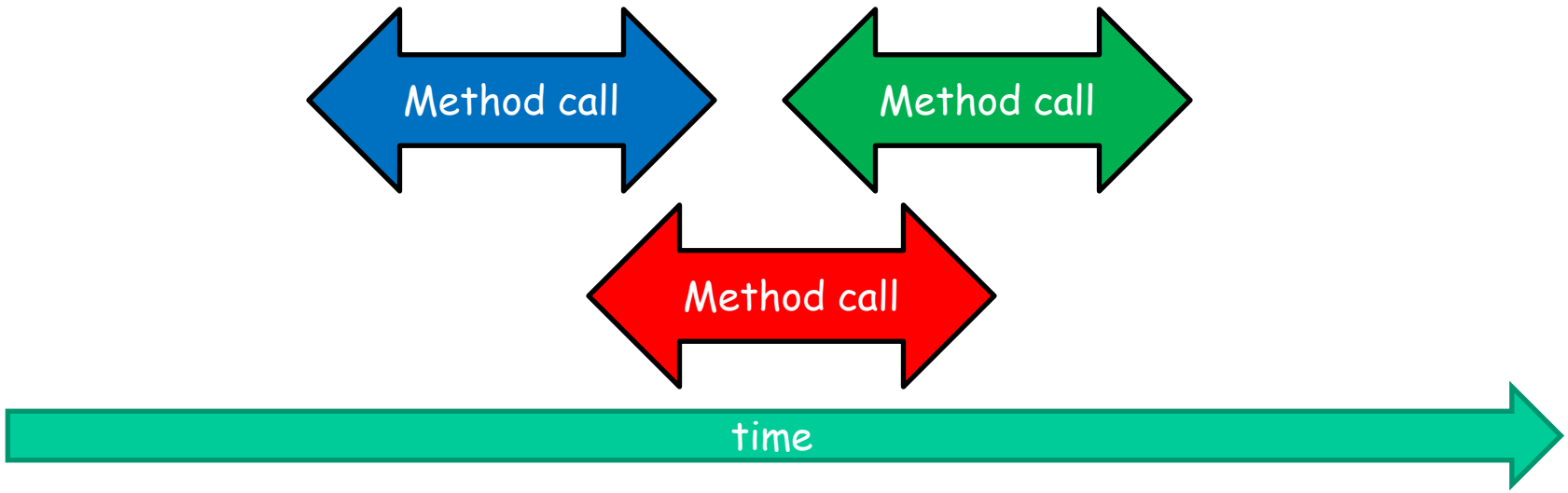


q.enq( )

q.deq( )

# Concurrent Specifications

- We need to understand that methods here "take time".
- In sequential computing, methods take time also, but we don't care.
  - Invocation (start of a method call).
  - Response (end of the method call).
- Methods can also take overlapping time.



# Sequential Specification

- We use pre-conditions and post-conditions.
- **Pre-condition** defines the state of the object before we call the method.
- **Post-condition** defines the state of the object after we call the method. Also defines returned value and thrown exception.

deq method

Pre-condition:

queue is not empty.

Post-condition:

- Returns first item in queue.
- Removes first item in queue.

Pre-condition:

queue is empty.

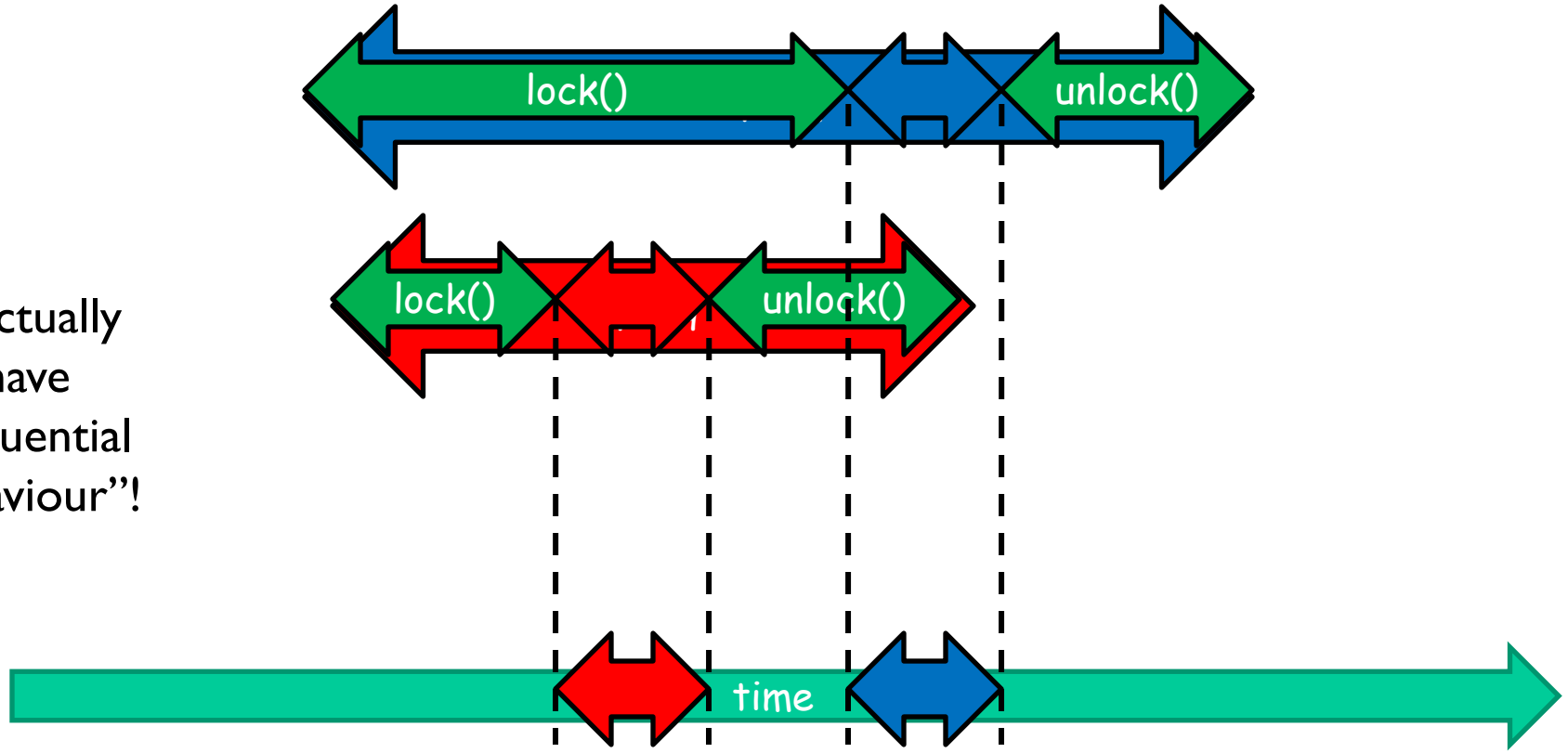
Post-condition:

- Throws EmptyException.
- Queue state is unchanged.

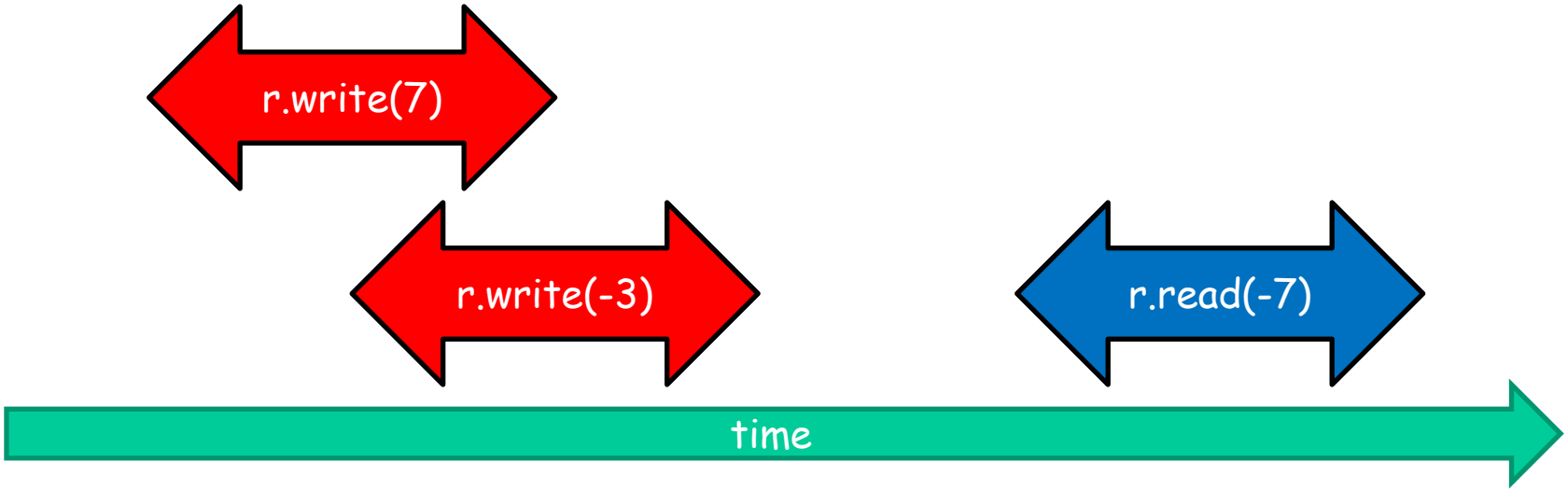
This makes your life easier, you don't need to think about interactions between methods and you can easily add new methods without changing descriptions of old methods.

# Let's try to explain the notion "concurrent" via "sequential":

So actually we have "sequential behaviour"!



# Executions



# Quiescent Consistency

- Principle 1

  Method call should appear to happen in a one-at-a-time sequential order

- Principle 2

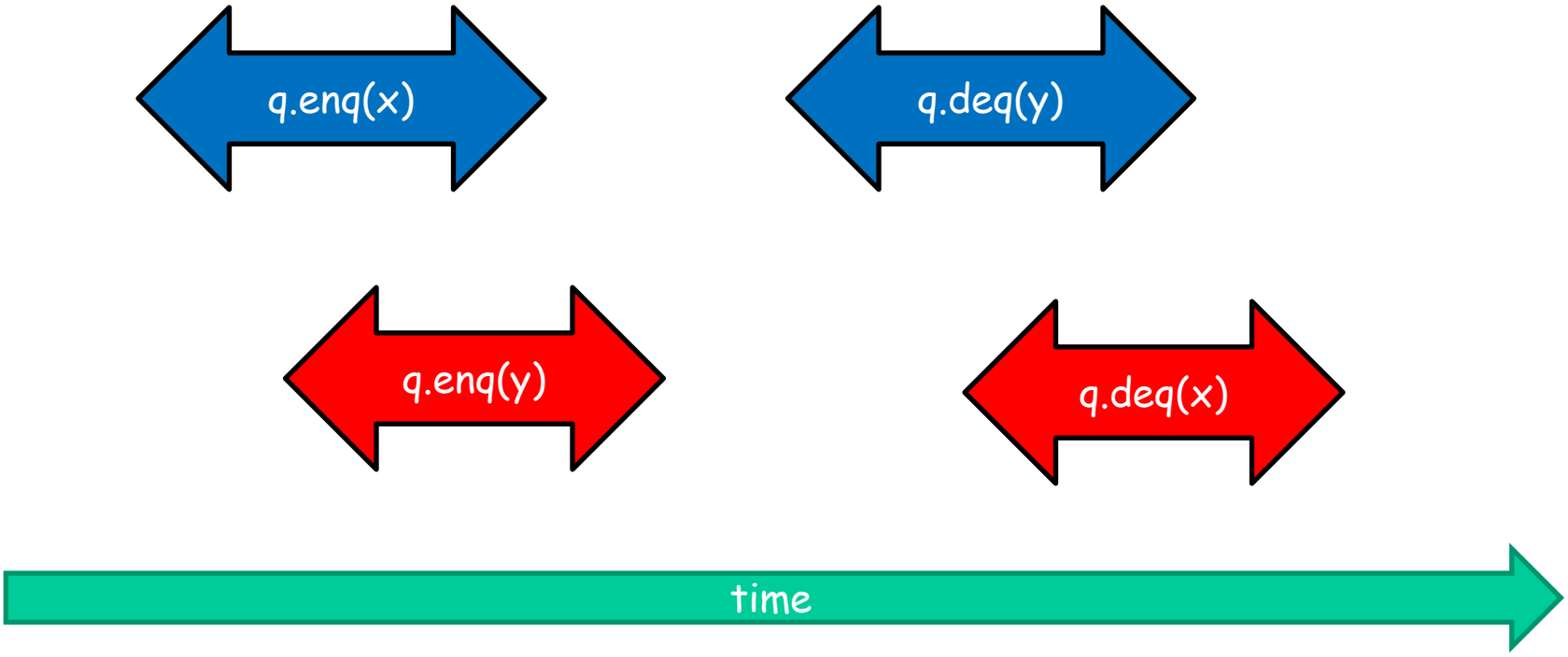
  Method calls separated by a period of quiescence should appear to take effect in real-time order.

# Sequential Consistency

- Principle 3

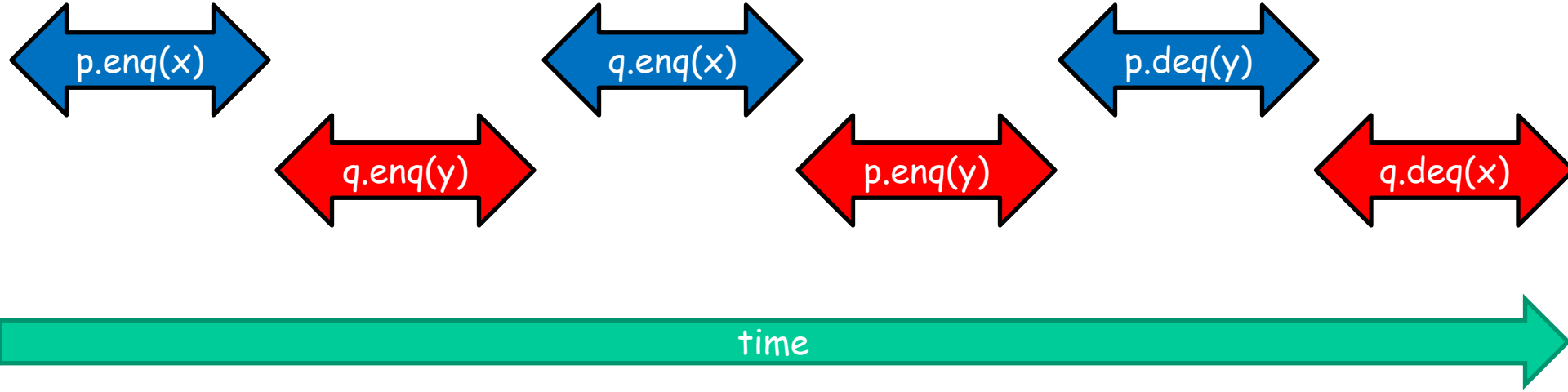
  Method calls should appear to take effect in program order

# Example




What orders are possible?

# Sequential Consistency




Is it SC with respect to "p"? What about "q"? What about "p and q"?

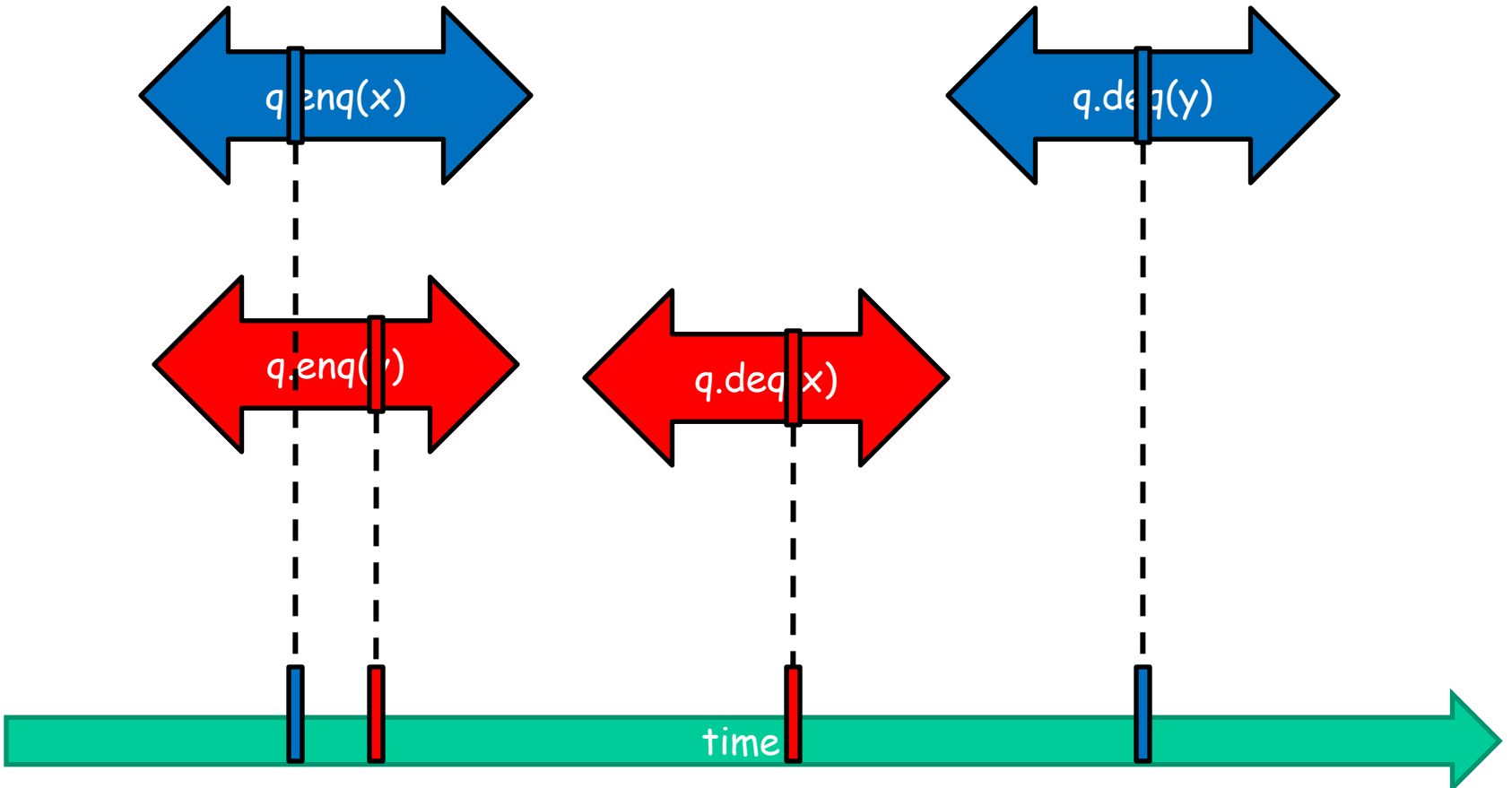
# Linearizability

- Principle 4

  Each method call should appear to be instantaneous at some moment between its invocation and response.

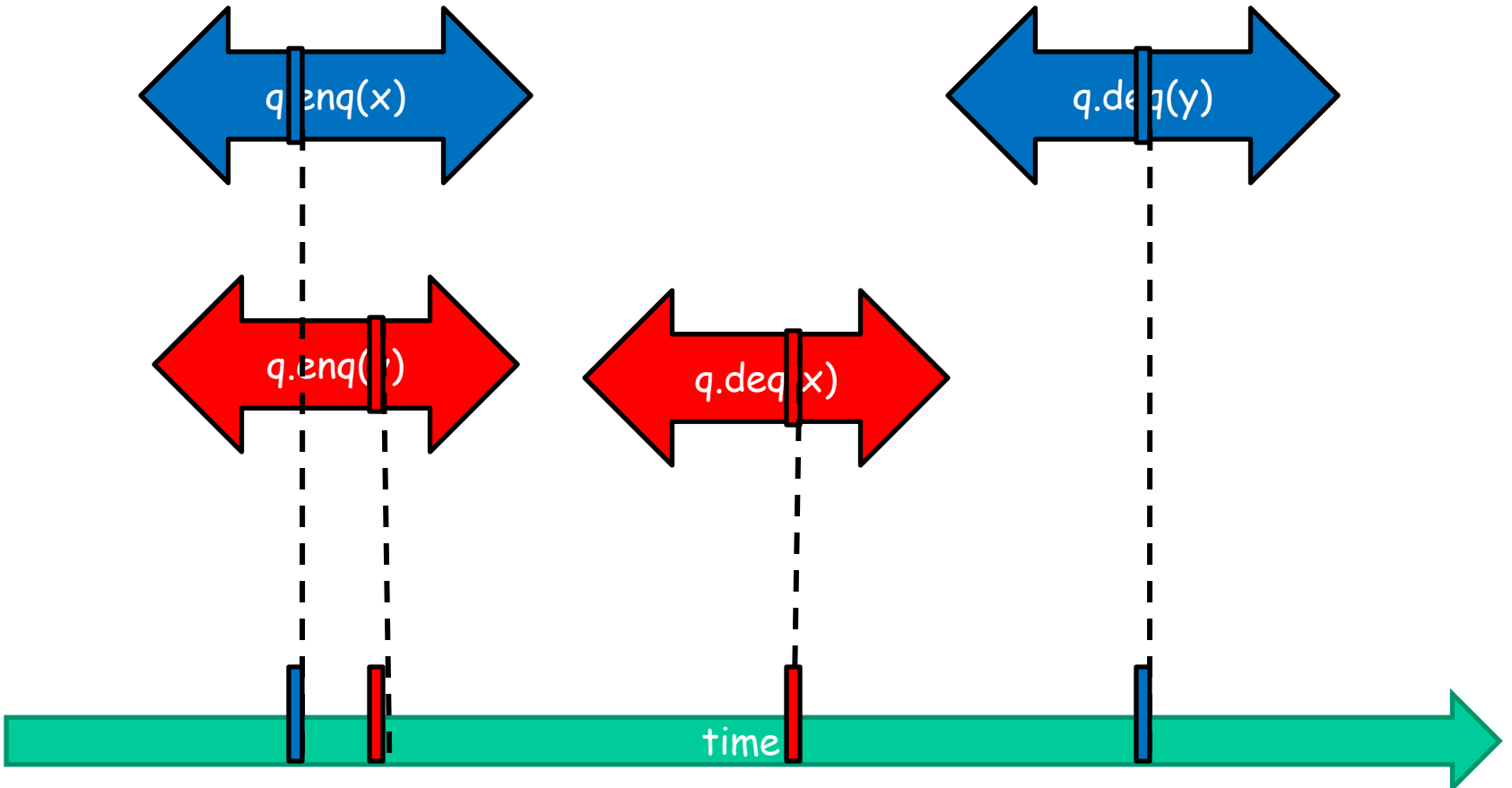
# Linearizablity

Is this linearizable? **Yes!**



# Linearizability

What if we choose other points of linearizability?



# Linearizability

Is this linearizable? **No!**