Linearizability for concurrent objects: The definition of linearizability for arbitrary concurrent objects extends the notion of linearizabiilty we have discussed for shared read/write registers.

We assume that each object has a sequential specification, which specifies the expected behavior if operations are performed sequentially (i.e., no overlapping operations).

For instance, a concurrent queue's sequential specification will be as follows:

- q.dequeue (or q.deq, for short) is dequeue operation performed on queue q.

  q.dequeue() operation performed on an empty queue q returns null.

  q.dequeue() performed on a non-empty queue returns the item at the head of the queue (i.e., the first item in the queue), and removes that item from the queue.

  In drawing execution timelines, for brevity, we may denote q.dequeue() operation that returns item x as q.dequeue(x) or q.deq(x).

- q.enqueue(x) (or q.enq(x), for short) operation on queue q adds item x at the tail of the queue.

In an execution, we may assume that a concurrent object is initialized suitably. For instance, we may assume that the queue is initialized as empty; alternatively, we may assume that the queue is initialized to contain some items, as desired. This is analogous to assuming that a read/write register is initialized to, say, 1.

An execution is linearizable if the following statement is true.

> *There exists a permutation of all the operations in the execution such that the following two conditions hold:*
>
> - *The permutation respects real-time order of the operations (and hence also respects the program order of each process). Recall that each process performs its operations sequentially, but operations performed by different processes may overlap in time.*
>
> - *If we consider the operations **on any single object** in their order in the above permutation, then the responses of these operations are consistent with the object's sequential specification (i.e., the responses of these operations on a single object are identical to the corresponding responses when the operations are performed sequentially on that object in their order in the above permutation).*

The lecture on this topic will consider some example executions. Please watch the lecture video for this discussion.

Intuitively speaking, in a linearizable execution, each operation can be viewed as ``taking effect'' at an instant of time between its invocation and its response.