

kuznecpl★ 75Last Edit: September 12, 2019 2:31 AM1.2K VIEWS

11

The idea is to use two semaphores, pushing and pulling, to maintain the invariants of the queue. Initially, no thread can dequeue (`self.pulling.acquire()`) until a thread has enqueued (`self.pulling.release()`). When capacity elements have been enqueued, `self.pushing.acquire()` will block the thread until a dequeue releases the semaphore again. Additionally, use a Lock so only a single thread can modify the actual queue at once.

```
import threading

class BoundedBlockingQueue(object):
    def __init__(self, capacity: int):
        self.capacity = capacity

        self.pushing = threading.Semaphore(capacity)
        self.pulling = threading.Semaphore(0)
        self.editing = threading.Lock()

        self.queue = collections.deque()

    def enqueue(self, element: int) -> None:
        self.pushing.acquire()
        self.editing.acquire()

        self.queue.append(element)

        self.editing.release()
        self.pulling.release()

    def dequeue(self) -> int:
        self.pulling.acquire()
        self.editing.acquire()


        res = self.queue.popleft()

        self.editing.release()
        self.pushing.release()
        return res


    def size(self) -> int:
        return len(self.queue)
```

Type comment here... (Markdown is supported)

Post

- NightDriveDrones★ 22September 12, 2019 4:47 AM
- Since `deque` s are threadsafe, is the `editing` semaphore necessary here?
- 3

Show 1 reply

Reply
- goodstudy123★ 24January 25, 2020 5:10 AM
- Nice solution! I swapped the sequence of `self.pushing.acquire()`, `self.editing.acquire()` and got "Time Limit Exceeded". Can you let me know why this sequence matters ? Thanks!
- 0

Show 1 reply

Reply