Daily Coding Problem #180

Problem

This problem was asked by Google.

Given a stack of N elements, interleave the first half of the stack with the second half reversed using only one other queue. This should be done in-place.

Recall that you can only push or pop from a stack, and enqueue or dequeue from a queue.

For example, if the stack is [1, 2, 3, 4, 5], it should become [1, 5, 2, 4, 3]. If the stack is [1, 2, 3, 4], it should become [1, 4, 2, 3].

Hint: Try working backwards from the end state.

Solution

It's a bit hard to see how we could transform our stack directly to the desired state. So let's consider going backwards from the desired state.

- Given [1, 2, 3, 4, 5] we want [1, 5, 2, 4, 3].
- We can see this is just a pairing of a queue with (5, 4) and a stack of [3, 2, 1] where we first pop off stack and then get from the queue.
- At this point, we can get to the above from a queue of (3, 2, 1, 5, 4)
- Which is just a rotation of (5, 4, 3, 2, 1)

Now let's go forward with these insights.

1. Put all elements into the queue and get (5, 4, 3, 2, 1)
2. Rotate len(stack) / 2 elements by taking them off the queue (5, 4) and putting them back to get (3, 2, 1, 5, 4)

3. Put ceil(len(stack) / 2) elements into the stack. The queue is now (5, 4) and stack is [3, 2, 1]
4. Pair them up len(stack) / 2 times. If stack is still not empty, pop one more time
5. Move all elements from the queue to the stack

```python
from Queue import Queue
import math

def interleave(stack):
    size = len(stack)
    queue = Queue()
    # Step 1.
    while stack:
        queue.put(stack.pop())
    # Step 2.
    for _ in range(size / 2):
        queue.put(queue.get())
    # Step 3.
    for _ in range(int(math.ceil(size / 2.0))):
        stack.append(queue.get())

    # Step 4.
    for _ in range(size / 2):
        queue.put(stack.pop())
        queue.put(queue.get())
    if stack:
        queue.put(stack.pop())
    # Step 5.
    while not queue.empty():
        stack.append(queue.get())
    return stack
```

Since each step is at most O(N), this runs in O(N) time, and since we use an extra queue, it takes up O(N) space.