

Homework-5

Ali Valiyev

December 2021

1 Homework solution

Task 1

Question R-6.8: Suppose an initially empty queue Q has executed a total of 32 enqueue operations, 10 first operations, and 15 dequeue operations, 5 of which raised Empty errors that were caught and ignored. What is the current size of Q?

Solution:

Since 32 enqueue operations are done, initially we have a size 32. 10 first operation does not influence to the queue size. Afterwards, 15 dequeue operations are done, 5 of which raised Empty errors. Therefore, 10 dequeue operations influencing to the size of queue. Hence, $32 - 10 = 22$. The size of Q is 22.

Question R-6.9: Had the queue of the previous problem been an instance of ArrayQueue that used an initial array of capacity 30, and had its size never been greater than 30, what would be the final value of the front instance variable?

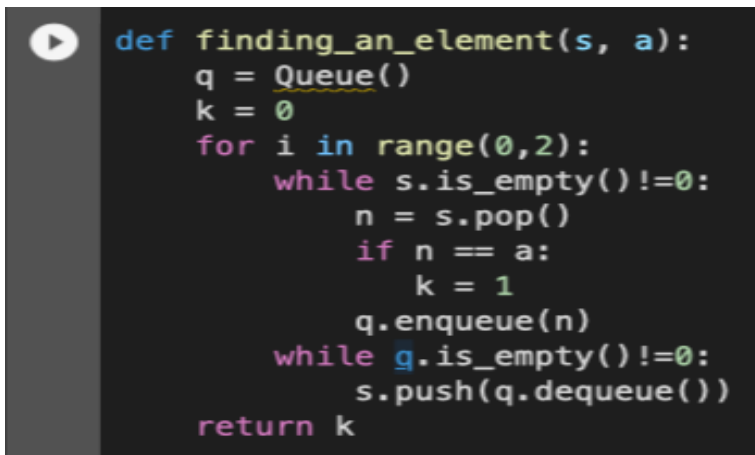
Solution:

In this question, all we need to do is count the number of Dequeues, as this is the only operation that affects the front pointer. Since 15 operations can change where front points and 5 of them would not have changed the position of front, so front would have moved a total of 10 times. Assuming the array is zero-indexed, front will be pointing at array index 10.

Task 2

Question C-6.27: Suppose you have a stack S containing n elements and a queue Q that is initially empty. Describe how you can use Q to scan S to see if it contains a certain element x, with the additional constraint that your algorithm must return the elements back to S in their original order. You may only use S, Q, and a constant number of other variables.

Solution:



```
def finding_an_element(s, a):
    q = Queue()
    k = 0
    for i in range(0,2):
        while s.is_empty()!=0:
            n = s.pop()
            if n == a:
                k = 1
            q.enqueue(n)
        while q.is_empty()!=0:
            s.push(q.dequeue())
    return k
```

colab link: <https://colab.research.google.com/drive/1eu1sbFRdgtYWApcKi476jAatFWxh3gtDscrollTo=-7KjsSH1vzT>

I am using "while" loop twice, but not as nested loops. So, its complexity is n(size of stack).

Task 3

Question: Assume that there is a queue class, namely WeirdQueue, that implements queue ADT and internally uses two stack objects, namely S1 and S2 instead of an array. (Please recall that we have implemented queue ADT by making use of an array which is internally maintained.) Provide two separate pseudo codes for the enqueue(e) and dequeue() operations of WeirdQueue.

Solution:

enqueue:

```
def enqueue(self, x):  
    self.s1.push(x)
```

dequeue:

```
def dequeue(self):  
    if len(self.s1) > 0 and len(self.s2) == 0:  
        while len(self.s1) != 0:  
            element = self.s1.pop()  
            self.s2.push(element)  
        return self.s2.pop()  
    else:  
        return self.s2.pop()
```