MaxQueue Write up

Benjy Katz

October 18, 2021

1 The Naive Implementation

Simply using a naive implementation of a standard queue would generally hit all the requirements if it wasn't for the max() requirement. It might seem simple to track the max element as the elements are en-queued, however once the max is de-queued finding the max becomes an $O(n)$ task of traversing the entire queue searching for the new max element. This doesn't seem so bad since only when the max is de-queued does the de-queue become $O(n)$ however in the worse case where the elements are en-queued in descending order does every call to dequeue() is $O(n)$.

2 My Implementation

2.1 The Array Based Queue

I implemented The MaxQueue by using two arrays, one of which represents the actual queue, and a second to be used as a list of Maxes. Both arrays start small and double when they reaches capacity. A reference to the start of the queue and a reference to the back is saved. When an element is en-queued it is placed in the back of the queue while when an element is de-queued it is removed from the front.

Simultaneously, a queue like data structure(For simplicity will be referred to as queue of maxes) of next maxes is stored along side the standard queue and allows for constant time for max() even after the max element is just de-queued. Every time an element is en-queued into the standard queue it is also added to the back of the queue of maxes and moves from the back of the queue to the front replacing values smaller than it.(See figure for clear demonstration) Once the max element is de-queued it is also de-queued from the queue of maxes and the current max element of the list is at the start of the queue of maxes.

2.2 Time Complexity

2.2.1 Dequeue: $O(1)$

Dequeue simply returns the element of the array that the start pointer is pointing to, advances the start pointer, and, if that element equals the first element in the queue of maxes, advances the start pointer of the queue of Maxes. Since this is three distinct operations irrespective of the size of the array each call to deque is $O(1)$

2.2.2 Max: $O(1)$

Max simply returns the element that the start pointer is pointing to in the Queue of Maxes. It makes no comparisons and is therefore $O(1)$

2.2.3 Size: $O(1)$

A variable constantly keeps track of the size of the queue adding one with an enqueue and subtracting one with a dequeue. It is therefor $O(1)$

2.2.4 Enqueue: amortized $O(1)$

When an element is enqueued there are a lot of moving parts. Firstly, it is added onto the back of the queue(constant time). Secondly, it doubles the array if the array fills up(amortized $O(1)$ since it does n operations every n times and $\frac{n}{n} = 1$). Thirdly, it adds the element to the back of the queue of maxes and iterates through the maxes replacing all elements less than that element and stopping when it encounters one larger. The two "best cases" are one: when the numbers come in descending order and every call to enqueue adds the element to the back of the array makes one comparison and stops. Two: when the elements come in descending order and every call to enqueue replaces the only element in the queue of maxes making only one comparison. However looking at the data structure from an amortized perspective every element can only be looked at by an element larger than it one time and then it is deleted. When every element is enqueued it comes in with two "coins" one to make the comparison right away to the element before it and one to pay for the comparison for when it is eventually replaced by a larger element(See diagram for visual). Therefore Enqueue maintains amortized $O(1)$

2.3 Space Complexity $O(n)$

Since the array only doubles when the array is full and the end of the queue utilizes the empty beginning of the array, the size of the data structure is always proportional to n.

enqueue 10, 15, 3, 17, 22, 100, 5

Standard 10 15 3 17 22 100 5

q of maxes 100 5

enque 4, 1

Standard 10 15 3 17 22 100 5 4 1

q of maxes 100 5 4 1

enque 7

Standard 10 15 3 17 22 100 5 4 1 7

q of maxes 100 7

max() = 100

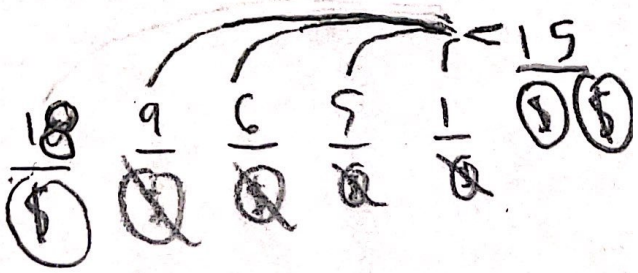
dequeue()x6

max() = 7

q of marks

18 9 6 5 1

enqueue(15)



18 > 15

18 15