Asher Kirshtein

MaxQueue Write up

Why Naive implementation doesn't work:

A Naive implementation wouldn't work because if you'd need to search through the whole queue every time you popped off your max you would get a worst-case scenario of O(n) and not O(1). Another issue could be if you were to count each value in the list every time you wanted the size it would be O(n).

My Implementation:

My implementation for size is I set a variable equal to 0. If I enqueue a value I add 1 to the variable and if I dequeue a value I subtract 1. When I want to get that size I return the number.

Getting the max is a little bit tougher. I store all of the potential values that could become a max value in an array. So when we insert the very first value it becomes the max value so we put it in the array and move to the next slot. After that when we add another value we check to see if it is larger than the value to its left. While it is larger than the value on the left we remove all of the values until we either get to the beginning of the array or we have a value that is larger than it on its left. Once we hit a value on its left which is larger we put it in the slot to the right of that value. When I want to get the max I have an index to the max value in the array. When we start the index is at 0. If I dequeue I check to see if we removed our max. If we did I just move my index over one to the right.

Why its O(1) or amortized O(1):

Both my size() and max() methods are O(1) since I pay for everything in my other methods and simply return my max and size which are already stored.

My Enqueue is amortized O(1). The parts that slow it down is when I would have to double the array in the queue which takes O(n) time but since it happens less and less often as we store more values it will average out to O(1) time making it amortized O(1). Another thing that would slow it down would be inserting my max into my max values array. In the worst case, we would be inserting a new max where we would have to wipe the whole array. That will not happen so often but if it does happen often it won't be wiping so many values and if it does have to wipe many values it won't be taking a large number of values out for a while so it will be averaging out to O(1) making it amortized O(1).

My Dequeue method is O(1) since all I do is pop off the value, check if it's our max. If it is our max I move my index over one and that's all.