Gray Houston

ghousto

Project 1 Analysis Questions

Part 1

1.)

a.) The worst case running time/computation for the enqueue operation when no resizing of the array occurs is **O(c)** where “c” is an arbitrary constant value. This operation occurs in constant time because the array is never traversed. The only operations that occur in this function (without resizing) are placing the patient into the array at a known index, modularly incrementing the tail index, and incrementing the counter for the patients. All of these operations run in O(c) time. Therefore, the enqueue operation runs a worst case of O(c) time.

b.) The worst case running time for the dequeue operation without resizing the array is **O(c)** where “c” is an arbitrary constant. Similar to the enqueue operation, dequeue runs in constant time because the array is never traversed and the internal computations/assignments occur in O(c) time (see above for examples of internal operations running in constant time).

c.) The worst case running time for the size() operation regardless of the array resizing or not is **O(c)** where “c” is an arbitrary constant. First, the array is never traversed. Second, this function only returns the value of a counter variable, which runs in O(c) time. Third, the increment/decrement of this counter runs in O(c) time and occurs during the enqueue/dequeue operation. Therefore, the worst case running time of the size operation is O(c).

Furthermore, because only a variable is returned during a call to the size() function and the counter variable is updated before any potential array resizing in the enqueue and dequeue operations (in fact, the counter variable is used as a loop bound in array resizing), the running time of the size operation/function is independent of array resizing. Therefore, the worst case running time in all situations for the size operation is O(c).

2.)

a.) The worst case running time for the enqueue operation with array resizing is O(n) where “n” is the number of elements in the queue. During array resizing, all of the elements in the old array must be copied to the new array, resulting in “n” number of assignment operations. Because the memory allocation and the computations in the array resizing run in O(c) and the other computations of the enqueue method run in O(c) time (as shown in part “a” above), the “n” assignments used to copy the elements becomes the determining factor. Because the worst case running time for copying the array runs in O(n), the overall worst case running time for the enqueue operation with array resizing is O(n).