CSSE230: Stacks and Queues

# Name(s):

Cooper Anderson

Joseph Peters

# Analysis

**Table 1:** Big-Theta runtimes of enqueue and dequeue for 4 implementations of the Queue ADT:

|  |  |  |
| --- | --- | --- |
| **Implementation** | **Enqueue runtime** | **Dequeue runtime** |
| LinkedList | O(1) | O(1) |
| ArrayList | O(N) | O(N) |
| Two stacks | O(N) | O(N) |
| Growable circular array | O(N) | O(1) |

# Part 2: Discussion

Justify each of the runtimes in Table 1, as described in the specification:

**LinkedList**

enqueue:

Assuming we have a tail pointer, we need only to append after this node.

dequeue:

Set new head equal to new node containing new item, then set old head to next.

**ArrayList**

enqueue:

If adding another item would exceed the array’s max size, it must traverse the entire array adding to a new larger array.

dequeue:

Remove the item at the beginning of the array, then move all other items back one.

**Two** **stacks**

enqueue:

Push all items in the back array to the front and push the new item to the back.

dequeue:

Push all items in the front array to the back and push the new item to the front.

**Growable circular array**

enqueue:

If size would become greater than the array’s max size, it must traverse the entire array adding to a new larger array.

dequeue:

Pop the item in the position (front + size) % length.