Jonathan Tsai

Programming Methodology II

Midterm A

Problem 2

1. My remove() function achieves O(1) time because the only thing we are doing is adjusting pointers, which takes constant time. The highlighted code shows where we are adjusting pointers.

/\*

\* Remove the value according to the FIFO rule and return it. Print the

\* removed value to the console.

\*/

**public** **int** remove() {

// if empty queue

**if** (first == **null**) {

System.***out***.println("Can't remove from an empty queue!");

**return** 0;

}

**int** data = first.value;

// if more than one value in the queue

**if** (first.next != **null**) {

// update first

first = first.next;

} **else** {

// if only one value in the queue

first = **null**;

last = **null**;

}

System.***out***.println("Removed value: " + data);

**return** data;

}

My getValue(index I) achieves O(1) time in the best case and O(N) time in the worst case, where N is the number of elements in the queue. This is because in the best case, we simply return the first element in the queue, which takes O(1) time. In the worst case, we must retrieve the very last element in the queue, which requires traversing through the entire queue and takes O(N) time.

\* retrieve the i-th element in the data structure, where i=0 corresponds to

\* the first element in the queue

\*/

**public** **int** getValue(**int** index) {

**int** i = 0;

Node ptr = first;

**while** (ptr != **null**) {

**if** (index == i) {

**return** ptr.value;

}

ptr = ptr.next;

i++;

}

**return** -1;

}

2. My implementation achieves O(1) time for add(), and it does so by keeping track of both the first and last element in the queue; as a result, in order to add an integer to the queue, all we have to do is adjust the pointer of the previous last element.

// adds an integer 'a' to the end of the queue

**public** **void** add(**int** a) {

// if empty queue

**if** (first == **null**) {

first = **new** Node(a, **null**);

last = first;

**return**;

}

Node oldlast = last;

last = **new** Node(a, **null**);

oldlast.next = last;

}