Report

The Doubly Linked List was difficult, but nothing out of the ballpark of understanding. I made some switches to the structure of the list compared to the Singly Linked List.

Instead of adding lots of functions such as getNext(), getPrevious(), setNext(), setPrevious(), I opted to just use a self.value, next, and prev in the __init__ of my Node constructor. It changed the ways I had to go about setting up the DLL constructor, but made it more simple and efficient in my opinion.

Inside of the DLL constructor, I created five methods. add2head, add2tail (like the single list). These were simple transitions to point each direction. With add2tail, I tried to not have the runtime be O(n), but couldn't figure it out sadly. I wanted to reference self.tail to not use the while loop, but ended up getting errors and couldn't properly debug it. However, it does work.

For the insert method, I ended up breaking it up into two. I struggled to make a positional insert in the doubly linked list, so ended up using an insert before and after. This finds the value as the position, then sets the node before or after the given value.

The delete method is similar to insert as well. I used value instead of position. Along with a good amount of if-cases, once the node is equal to the inputted number, the method will crunch the nodes down and essentially delete the requested node.

The last section of the DLL constructor, display, works in forward and reverse. To make this possible, I had a 'count' in a sense for the original forward traverser. Once that traverser got to the end, I saved that node with 'last', and had it reverse to show the previous order.

I had a lot of fun creating this! A lot of errors, but the experience making it was great.