Jack Micher

9/17/20

CS/IT 200

Lab 2 Part I

1. Since we need interact with the last node, a doubly linked list will help us be able to jump one Node back and forth to accomplish that.
2. Having a circular linked list would probably cause more problems for this type of implementation, as going removing from the end of a singly linked circular list would require the program to wrap all the way around to the 2nd to last node. A doubly linked circular list could possibly work, but it would probably require more special cases (such as when list is empty, or when list has 1 node).
3. The addition of a tail reference alongside the head reference is the first step to creating deque. With references to both the start and end of the linked list we can quickly call methods to delete objects at the start and end without having to move through the list. There may be special cases if we only use the head and tail references, which is why…
4. …We should add sentinel nodes, as head and tail, to the linked list. By using head and tail sentinel nodes we would remove the need to create more special cases for adding or removing from lists with 0 or 1 elements. These sentinel nodes would probably streamline a lot of the code in the class methods.