CS/IT 200

Lab 2: Linked Lists

Recommended Due Date: Thursday, September 24, 11:59pm

Submission

- Submit all code and answers to questions in a single zip file to the Lab 2 assignment folder on Kodiak.

Goal: Implement a variation of the List ADT using a linked list.

Part I

A deque (pronounced "deck") is a special kind of list where you can only add and remove from the beginning and end of the list. Answer the questions below, defending your answers with running times and other information from class. Note that your answers may be inter-related, so read all questions before answering any of them. Your goal should be to answer the questions in a way that leads to adding and removing being $O(1)$ at the front and back of the list.

1. Would it be better to implement a deque as a singly linked list or a doubly linked list? Why?
2. Would it be better to implement a deque as a circularly linked list or not? Why or why not?
3. Would an implementation of a deque benefit from maintaining a tail reference? Why or why not? If yes, would it be in addition to or instead of a head reference?
4. Would an implementation of a deque benefit from having sentinel (dummy) nodes at either the head or tail (or both)? Why or why not?

Part II

Implement a `Deque` class with the methods below. Place the class in a module called `deque.py`. Your implementation should be a linked structure (that is, it should use `Nodes`). Use your answers from Part I to guide your implementation. All methods (except `__str__`) should run in $O(1)$ time. Be sure to include comments for each function.

- **`Constructor`** – Creates an empty deque.
- **`add_to_front(x)`** – Adds `x` to the beginning of the deque.
- **`add_to_end(x)`** – Adds `x` to the end of the deque.
- **`remove_from_front()`** – Removes and returns the item at the beginning of the deque. Raise an `IndexError` if the deque is empty.
- **`remove_from_end()`** – Removes and returns the item at the end of the deque. Raise an `IndexError` if the deque is empty.
- **`__len__()`**
- **`__str__()`** – Strongly recommend copying the `__str__()` method from `LinkedList` and modifying it if necessary.

You may include other public or private methods if you wish. However, there must not be any methods that allow insertion or removal from anywhere other than the beginning or end of the list.

In `lab2.py`, write code that creates an instance of a `Deque` and tests the methods listed above.

What to Submit

Submit a zip file containing:

- Your code in `deque.py` and `lab2.py`
- A Word, text file, or PDF containing your answers from Part I.

Rubric

Grade	Overall	Part I	Part II
4	4 in both parts.	Answers are accurate and would lead to O(1) operations.	Code contains, at most, minor bugs. All add/remove run in O(1) time. Solution reflects answers from Part I. Well commented.
3	At least a 2 in Part I and a 3 in Part II.	All answers are accurate, but do not lead to O(1) operations. -OR- 3 of 4 answers are accurate and lead to O(1) operations.	Fails to address edge case in 1 of the add/remove methods. -OR- Missing 1-2 comments or working code that differs from Part I answers.
2	At least a 1 in Part I and a 2 in Part II.	At least 2 answers are accurate and lead to O(1) operations.	Significant bugs or slower runtime in 1 function or missing edge case in 2 of the add/remove methods. May lack comments on some functions.
1	At least a 1 in both parts.	At least 1 answer is accurate.	Missing edge cases in all functions OR significant bugs or slower runtime in no more than 2 functions.
0		Answers are inaccurate or not given.	Fails to meet the standards for 1.