CSC111 Winter 2023 Assignment 1: Linked Lists and Blockchain

TODO: FILL IN YOUR NAME HERE

January 29, 2023

Part 1: Linked List Debugging Exercise

1. (a) The issue is with links of sequences being broken. After assert statement, we can assume curr is last node in LinkedList. In parallel assignment, self._first is set to be 'curr', however, we assumed curr was last node in LinkedList, and its next is None. Therefore, self._first becomes last node, and its next becomes None. As a result, making LinkedList's first node become last node will lead LinkedList to only contain 1 last item in LinkedList. For example, LinkedList input [1, 2, 3] will become [3] when converted to list. This means, in parallel assignment, when curr is set to be self._first, it would not affect anything to the LinkedList as LinkedList's first node is set to be the last node and its next is None. So, reaching to curr won't be possible since link to curr is broken. In other words, since "curr" is not reachable, it won't be effective and since sequences(second node and so on..) are not connected to first node, but instead self._first's next is None, the sequences will end after the first node; not continuing to "curr". Thus, leading to contain only one last item in linked list, which is not what we're expecting for elements of 2 or more items.

   (b) Complete your work for part (b) in `a1_part1.py`.

2. (a) Function does not handle the case where an item is fewer than 2 in the linked list; meaning case handling is not implemented. If the item is less than 2, the function shouldn't run the rest of the code as mentioned in the function description. Let's look more deeper on why the function raised an error. Specifically, the function causes errors when the linked list is empty, i.e., LinkedList([]) for example. For an empty linked list, self._first and curr are automatically set to None (due to "Optional[_Node] = None" in class _Node), and so they won't have the next node. So when the while loop tries to check whether curr.next is None or not, since curr doesn't have the next attribute, it raises an AttributeError. Following the same reasoning, assert code will also have the same issue with not being able to access the next attribute, since it doesn't exist. In other words, error is caused since Professor Liu did not implement the case when item is less than 2; if item is less than 2, function shouldn't run rest of code in the first place and should not try to access next attribute of curr to avoid errors. Thus, when linked list is empty, function causes an error, since Professor Liu did not handle cases where item is fewer than 2.

   (b) Complete your work for part (b) in `a1_part1.py`.

3. (a) Complete your work for part (a) in `a1_part1.py`.

   (b) When linked list have only 1 item, it doesn't raise error. Items for self._first and curr are the same node with the same id because while loop only runs when curr.next is NOT None. Since curr.next is None because there's only 1 element, we do not enter the while loop, so we do not make curr to be curr.next, thus, curr stays as self._first. As a result, they both have same item, so when there's parallel assignment, linked list will still contain 1 identical element. When there's only 1 node, there's no need for curr to be reachable as it already contains only 1 identical element as self._first and curr is the same. Thus, this avoids the error that I pointed out in first question(i.e. link being broken). So then question is why linked list with 1 element avoids error I mentioned in question 2? Well since, self._first and curr(they're same) is not None, but instead it is class _Node, with having next attribute: None, when there's while loop and assert code trying to access next attribute it does not raise an AttributeError, since attribute exist. Thus this avoids error I mentioned in question 2(i.e. not having next attribute). So now, we can conclude that linked list with 1 element avoids 2 errors that I've pointed out in previous questions.

4. Complete your work for this question in `a1_part1.py`.

Part 2: Blockchain and Cryptocurrencies

Complete this part in the provided `a1_part2.py` starter file. Do **not** include your solutions in this file.