Data Structures

Homework Assignment 5 - Linked List - SLL

The total score for this assignment is 50 Points. The other 50 points are awarded for solving the assignment LinkedList - DLL, released after the midterm exam.

Problem 1 – Remove Duplicates Sorted List - 12.5 Points
Problem 2 – Linked List Cycle - 12.5 Points
Problem 3 – Sort List - 12.5 Points
Problem 4 – Palindrome Linked List - 12.5 Points

25% of Gradescope Autograder test cases are hidden for this assignment.

Spring 2024 - Revision 1 - *October*

Problem 1 – Remove Duplicates Sorted List - 12.5 Points

Implement the member function *deleteDuplicates()*, which removes all duplicated elements from the sorted SinglyLinkedList such that each element appears only once.

Example

```
ls1 = SingleLinkedList()
ls1.insertAtFirst(4)
ls1.insertAtFirst(2)
ls1.insertAtFirst(2)
ls1.insertAtFirst(1)

print(ls1)  # Should print: Head-->1-->2-->2-->4-->None
ls1.deleteDuplicates()
print(ls1)  # Should print: Head-->1-->2-->4-->None
```

Requirements

- Your function has to be in O(n) time complexity.
- Your function has to be in O(1) space complexity.
- Your function has to work in place.

Problem 2 – Linked List Cycle - 12.5 Points

Implement the member function *hasCycle()*, which detects whether a SinglyLinkedList has a cycle.

Example

```
ls1 = SingleLinkedList()
ls1.insertAtFirst(4)
last = ls1._head
ls1.insertAtFirst(3)
ls1.insertAtFirst(2)
ls1.insertAtFirst(1)
last._next = ls1._head

print(ls1.hasCycle())  # Should print: True
```

Requirements

- Your function has to be in O(n) time complexity.
- Your function has to be in O(1) space complexity.

Hint

- Don't try to print our lists to avoid endless loops. Use the debugger instead.

Problem 3 – Sort List - 12.5 Points

Implement the member function *sortList()*, which sorts a SinglyLinkedList in ascending order.

Example

```
ls1 = SingleLinkedList()
ls1.insertAtFirst(3)
ls1.insertAtFirst(2)
ls1.insertAtFirst(6)
ls1.insertAtFirst(5)

print(ls1)  # Should print: Head-->5-->6-->2-->3-->None
ls1.sortList()
print(ls1)  # Should print: Head-->2-->3-->5-->6-->None
```

Requirements

- The list has to be sorted in ascending order.
- Your function has to be in O(n log n) time complexity.
- Your function has to be in O(1) space complexity.
- Your function has to work in place.

Problem 4 – Palindrome Linked List - 12.5 Points

Implement the member function *isPalindrome()*, which checks whether a SinglyLinkedList elements form a palindrome.

Example

```
ls1 = SingleLinkedList()
ls1.insertAtFirst(1)
ls1.insertAtFirst(2)
ls1.insertAtFirst(2)
ls1.insertAtFirst(1)

print(ls1)  # Should print: Head-->1-->2-->2-->1-->None
print(ls1.isPalindrome())  # Should print: True
```

Requirements

- Your function has to be in O(n log n) time complexity.
- Your function has to be in O(1) space complexity.