Linked list

**Space complexity: O(n)**

**Time complexity: O(n)**

Python code:

class Node:

def \_\_init\_\_(self):

self.next = None

self.data = data

class LinkedList:

def \_\_init\_\_(self):

self.head = None

def push(self, val):

if self.head == None:

self.head = Node(val)

else:

curr = self.head

while curr == None:

curr = curr.next

curr = Node(val)

def show\_llist(self):

curr = self.head

llist = []

while curr != None:

llist.append(curr.data)

curr = curr.next

return llist

def length\_llist(self):

curr = self.head

count = 0

while curr != None:

count += 1

curr = curr.next

return count

def get\_element\_pos(self, element):

curr = self.head

count = 0

while curr == None:

if curr.data == element:

return count

curr = curr.next

count += 1

def reverse(self):

prev, curr = None, self.head

while curr != None:

nxt = curr.next

curr.next = prev

prev = curr

curr = nxt

self.head = prev

def swap\_two\_vals(self, val1, val2):

pos1 = self.get\_element\_pos(val1)

pos2 = self.get\_element\_pos(val2)

curr = self.head

count = 0

while curr != None:

if count == pos1:

curr.data = val2

elif count == pos2:

curr.data = val1

curr = curr.next

count += 1