CSC212 Data Structures

Exercise 5 – Ordered Linkied List

Consider the following Node class and Ordered Linked List class definitions, followed by the instructions in the application program. There are 10 instructions in the application program (numbered 1 to 10). Draw diagrams (by tracing the code) to show the content of the Ordered Linked List after each instruction or indicate the output of the print instructions.

class Node:

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

self.data = initdata

self.next = None

def getData(self):

return self.data

def getNext(self):

return self.next

def setData(self,newdata):

self.data = newdata

def setNext(self,newnext):

self.next = newnext

class OrderedList:

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

self.head = None

def search(self,item):

current = self.head

found = False

stop = False

while current != None and not found and not stop:

if current.getData() == item:

found = True

else:

if current.getData() > item:

stop = True

else:

current = current.getNext()

return found

def add(self,item):

current = self.head

previous = None

stop = False

while current != None and not stop:

if current.getData() > item:

stop = True

else:

previous = current

current = current.getNext()

temp = Node(item)

if previous == None:

temp.setNext(self.head)

self.head = temp

else:

temp.setNext(current)

previous.setNext(temp)

def isEmpty(self):

return self.head == None

def size(self):

current = self.head

count = 0

while current != None:

count = count + 1

current = current.getNext()

return count

mylist = OrderedList() #1

mylist.add(31) #2

mylist.add(77) #3

mylist.add(17) #4

mylist.add(93) #5

mylist.add(26) #6

mylist.add(54) #7

print(mylist.size()) #8

print(mylist.search(93)) #9

print(mylist.search(100)) #10

Consider the *add* method in the Unordered Linked List class definition in Exercise 4 and the *add* method in this Exercise. In your own words, compare and contrast the work in the two *add* methods relative to the action of adding the new node to the list in each scenario. If you are to analysis the complexity, what is the Big O notation of each *add* method?

The major difference is the location of the added number. In unordered list The number is just placed in slot 0 of a list for the unordered list and everything is moved forward. In the case of ordered lists the number walks down the row of numbers until it hits a number bigger than it, than it stays there. If there is no bigger number, the added number becomes the last item on the list. The unordered lists there is a bigger Big 0, because there is more data that takes longer to sort. Ordered lists sort as they are created making them have a smaller Big 0.