Nancy Hernandez

Olac Fuentez

Lab 2 Report

**Introduction**

The objective of this lab is to implement several algorithms for finding the median of a list of integers, using objects of the List class described in class, and compare their running times for various list lengths.

**Proposed Solution**

To achieve getting the medians I first figured to create the methods ElementAt(), GetLength(), and randomInts() as they are very important for the four other methods that need to be created for each type of sorting.

Now, that all those methods necessary were completed, I began with the bubbleSort() method. This method I thought should begin with an if statement that makes sure that if the list is less than two then it would just return. An else follows the if, and this in this else is where all the work begins. I created a for loop that goes through every element in the list. Inside this loop I made a while loop that should compare two elements at a time.

Then, I moved onto mergeSort(). For this method I thought about starting it with an if statement just like in the bubbleSort() method. The if statement should check if the list is less than two and return the list as it is already sorted. The else following this if statement should take care of the rest of the comparisons.

Next, the method quicksort(). Unlike the past two functions I thought about making a separate method, pivot(), that would do all the comparisons and then have the method quicksort() do the two recursive calls that will do the rest of the sorting. In the method pivot() I do the comparisons of every element in order to determine which values are smaller or larger that the pivot and have a variable remember where in the list these numbers are located. This way, they can be arranges in the right order.

Finally, for each of these sorting methods I created a method that would calculate the median given the list. And once calling these methods will return the list sorted correctly as well as returning the median of that specific list.

**Setup**

To complete this lab I used an HP Pavilion x360 Convertible with a 2.71 GHz Intel® Core(TM) i5 processor.

**Results**

n = 5 n = 10

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generated

n = 20 n = 50

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generated

**Time Analysis**

|  |  |  |  |
| --- | --- | --- | --- |
|  | O(n^2) | n log(n) | n log(n) |
| n | Bubble Sort | Merge Sort | Quick Sort |
| 5 | 25 | 3 | 3 |
| 10 | 100 | 10 | 10 |
| 20 | 400 | 26 | 26 |
| 50 | 2,500 | 84 | 84 |

**Conclusion**

From this lab I learned how python functions when using different types of sorting methods. I found that quick sort was fastest, merge sort was second fastest, and bubble sort was the slowest and most inefficient of them all. I also learned how to implement these different sorts into python.

**Appendix**

**//finds medium**

def Median(L):

C = Copy(L)Sort(C)

return ElementAt(C,GetLength(C)//2)

//**list class**

#Node Functions

class Node(object):

# Constructor

def \_\_init\_\_(self, item, next=None):

self.item = item

self.next = next

def PrintNodes(N):

if N != None:

print(N.item, end=' ')

PrintNodes(N.next)

def PrintNodesReverse(N):

if N != None:

PrintNodesReverse(N.next)

print(N.item, end=' ')

#List Functions

class List(object):

# Constructor

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

self.head = None

self.tail = None

def IsEmpty(L):

return L.head == None

def Append(L,x):

# Inserts x at end of list L

if IsEmpty(L):

L.head = Node(x)

L.tail = L.head

else:

L.tail.next = Node(x)

L.tail = L.tail.next

def Print(L):

# Prints list L's items in order using a loop

temp = L.head

while temp is not None:

print(temp.item, end=' ')

temp = temp.next

print() # New line

def PrintRec(L):

# Prints list L's items in order using recursion

PrintNodes(L.head)

print()

def Remove(L,x):

# Removes x from list L

# It does nothing if x is not in L

if L.head==None:

return

if L.head.item == x:

if L.head == L.tail: # x is the only element in list

L.head = None

L.tail = None

else:

L.head = L.head.next

else:

# Find x

temp = L.head

while temp.next != None and temp.next.item !=x:

temp = temp.next

if temp.next != None: # x was found

if temp.next == L.tail: # x is the last node

L.tail = temp

L.tail.next = None

else:

temp.next = temp.next.next

def PrintReverse(L):

# Prints list L's items in reverse order

PrintNodesReverse(L.head)

print()

L = List()

print(IsEmpty(L))

for i in range(5):

Append(L,i)

Print(L)

PrintRec(L)

PrintReverse(L)

Remove(L,2)

Print(L)

Remove(L,20)

Print(L)

Remove(L,0)

Print(L)

Remove(L,4)

Print(L)

print(L.head.item)

print(L.tail.item)

**Academic Dishonesty Statement**

I, Nancy Hernandez, was not involved in any copying from or providing information to another student, possessing unauthorized materials during a test, or falsifying data in laboratory reports. Neither did I participate in any type of collusion involving collaboration with another person to commit an academically dishonest act.