CS 2302

Fall 2019

Lab Report #3

John Rodriguez

Due: October 4, 2019

Professor: Olac Fuentes

TA: Anindita Nath

Introduction

For lab 3 we were given the task to create a sorted list class which is used to create a linked list. We were also told to create ten different functions that are used to manipulate the linked list or return a specific item from the list. This lab’s purpose is to become familiar with using linked list in python and comparing the big O time complexity of the functions we created for the sorted linked list to a normal linked list.

Proposed Solution Design and Implementation

I approached this lab by first creating a Node class and a SortedList class. The Node class is used to create a node that can be used in a linked list and the SortedList class is used to create a linked list, however it specifically is used to create a linked list with all the elements being in ascending order. After I created these two classes I moved on to creating the ten specified functions we needed to create for the lab.

The first function I focused on was the Print one which is used to print all the contents of the list. This function was easy to integrate because it is the same as crating a print function for a normal linked list. I first gave the variable t the reference to the list head and then used a while loop to print the data of the current node and move one to the next one in the list continuing to print the data until the end of the list is reached.

The second function Insert was different than the print function because unlike a normal list where you would add the new node to the head of the list in a sorted list the node needs to be placed in the correct position to keep the list’s elements in ascending order. This function has two parameters which are the sorted list and I which holds a number to be inserted into the list. The first thing I did was create a node which holds the value of the variable i. I then created a condition which checks if the list is empty which would then make the new node the head of the list. The next condition checks if the head of the list element is larger than the new node’s and then makes the new node the head of the list and it’s refence the old head of the list. The last conditional is a else condition which goes threw the list until one of the nodes data holds a number that is larger than the data in the new node and the new node is given the reference to that node and the new node is then added to the list.

The third function is called Delete which has two parameters, the first is the sorted list and the other is i which holds a number that is to be deleted from the list if one of the list’s elements has the same number. I first gave the variable t the refence to the list’s head and created three conditionals. The firs condition is if the list is empty which will do noting and exit the function. The next condition is if the head of the list holds the value to be deleted which will make the head of the list the next refence in the list or make the list empty if the head is the only element in the list. The last condition is an else which has a variable called prev which holds the reference to the previous node in the list. I then created a while loop which goes threw the list and if the element to be deleted is found the reference to the previous node is given the next of the current node removing that node from the list.

The fourth function is called Merge which is used to combine two sorted linked list into one list. The Merge function has two parameters, one being the first list and the second being the second list. I was unable to complete this function because when I tried to create a variable with a reference to the head of the second list it would say that that reference doesn’t hold a data value even though it did. I commented out the code so it could still be included in my code but note crash the entire program.

The fifth function is called IndexOf which is used to return the index of the element in the list that holds the value that the variable i is given. The first thing I did was create a variable called count which holds the value of the current index of the list and I gave the variable t the reference the list head. I then created two conditions which the first being if the list is empty which would then return the value -1. The next condition is if the list isn’t empty which then goes threw the list using a while loop and as the list is moved threw the count value is increased. When the value if the variable i is found in one of the elements of the list the value of the variable count is returned which holds the value of the index of that element in the list and if the value of i isn’t found -1 is returned to show that it wasn’t found in the list.

The sixed function is called Clear which is used to remove all elements from the list making it empty. I added one condition to check if the list is already empty which exits the function without doing anything and if it’s not empty, I make the next reference from the head of the list None and then make the head None which clears the entire list.

The seventh function is called Min which is used to return the smallest element in the list. I first check is the list is empty and if it is, I return math.inf. If the list isn’t empty, I return the data stored in the head of the list because the list is sorted in ascending order the head always holds the smallest element.

The eighth function is called Max which is used to return the largest element in the list. I first give the variable t the reference to the head of the list. I then check if the list is empty which will return math.inf if it is empty. The next condition checks if the list only contains one element which then returns the data found in the head of the list. If neither of these conditions are met, then I used a while loop to go threw the list until the last element of the list is reach then I return the data found inside it which hold the largest number in the list.

The ninth function is called HasDuplicates which checks if the list contains duplicate numbers inside its nodes. I first give the variable t the reference to the head of the list. I then check if the list is empty or if it only contains one element which then returns false. If this condition isn’t met, then I used a while loop to go threw the list and checked if any of the references to the next node contain the same number which if a duplicate if found it returns true but if it doesn’t it returns false.

The tenth function is called Select which has two parameters which one being the list and the second being the variable k which holds the value of a kth smallest index that you wish to return from the list. I first give the variable t the reference to the head of the list and create a variable called count which holds the current index of the list. I then made a condition that checks if the list is empty and if it is returns math.inf. If this condition isn’t met, then I used a while loop to go threw the list and if the value of count is equal to the value of k the data of the current node is returned. If the index k isn’t found in the list, then math.inf is returned.

Experimental Results

**Print Function Tests**

To test the Print function, I just used the other functions tests because the Print function was used when testing every function.

**Insert Function Tests**

To test the Insert function, I started by creating a new sorted list called llist and then used the Insert function to add new items to the list. I also tested adding a duplicate number to see if the function performed correctly and placed all the numbers in ascending order.

A close up of a logo

Description automatically generated

**Delete Function Tests**

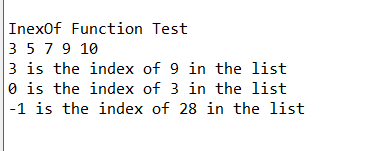
To test the Delete function I used the list I created using the Insert function then I used the Delete function to remove the number 1 from the list, then the number 7 to see how then function behaves when removing a duplicate number from the list.

A close up of a logo

Description automatically generated

**IndexOF Function Tests**

To test the IndexOf function I searched the index of 9 in the list which is 3 and then the index of 3 in the list which was 0. I then searched for a number that wasn’t in the list and -1 was returned to show that that number isn’t found in the list.



**Clear Function Tests**

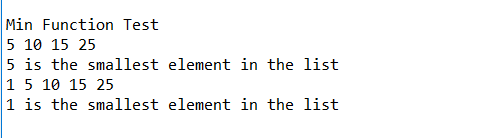
To test the Clear function, I used the llist I created and then used the Clear function to remove all items from the list. I then added 25 to the empty list and then used the Clear function again to test how the function behaves with one item in the list. With both test the list was cleared of all items.

A close up of a logo

Description automatically generated

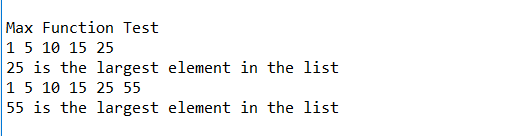
**Min Function Tests**

To test the Min function, I used the Min function on the list llist and 5 was returned which was the smallest element in the list. I then added the number 1 to the list and used the Min function again and this time the number 1 was returned which was the new smallest number in the list.



**Max Function Tests**

To test the Max function, I used the Max function on the list llist and 25 was returned which was the largest item in the list. I then added the number 55 to the list and used the Max function again and this time 55 was returned as the largest element in the list.



**HasDuplicate Function Tests**

To test the HasDuplicate function I used the function on a list that didn’t have any duplicate numbers in it and false was returned. I then added the number 25 to the list making it a list that contained duplicate numbers the used the HasDuplicare function and this time true was returned.

A close up of a logo

Description automatically generated

**Select Function Tests**

To test the Select function I gave k the value of 3 to find the 4th smallest element in the list and when I did it returned 15 as the 4th smallest element in the list which was correct. I then tested is k was given a value that was larger then the number of items in the list. I gave k the value of 25 and when I used the function inf was returned which was the correct response.

A screenshot of a cell phone

Description automatically generated

**Empty List Test**

For this test I tried using all the functions except for the Insert function on an empty list to see if all the function could handle the case of an empty list and all the functions behaved as expected.

A close up of a logo

Description automatically generated

|  |  |  |
| --- | --- | --- |
| **Function** | **SortedList** | **List** |
| Print() | O(n) | O(n) |
| Insert(i) | O(n) | O(1) |
| Delete(i) | O(n) | O(n) |
| Merge(M) | O(n2) | O(n) |
| IndexOf(i) | O(n) | O(n) |
| Clear(i) | O(1) | O(1) |
| Min() | O(1) | O(n) |
| Max() | O(n) | O(n) |
| HasDuplicates() | O(n) | O(n2) |
| Select(K) | O(n) | O(n2) |

Conclusion

This lab helped me become more comfortable when using linked list in python. I also found that a sorted linked list has benefits when trying to find specific things in the list when compared a normal unordered linked list. I also have a better understanding on how to create a function for a linked list like almost always creating a conditional that’s used for checking if the list is empty first. I also now feel much more comfortable using python now.

Appendix

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class