**Lab # 2 Report**

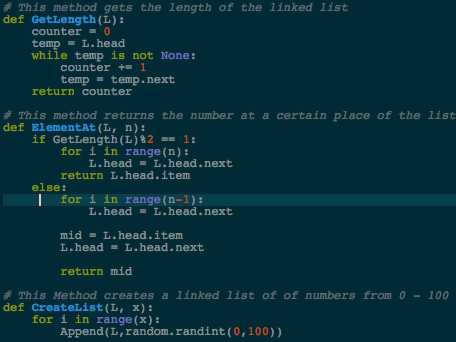
**Introduction:**

The main purpose of this assignment was to practice problems that use linked lists as data structures. ﻿For this lab assignment we were asked to create linked lists with random numbers from 0 to 100 and sort them with different methods. In this case we have to sort them using bubble sort, merge sort, and quick sort method. Once we have the lists sorted we have to find the median of each list. To complete this task, we had to also have a method that return the length (*GetLength(L)*) of the list, one that returns an specific item of a list (*ElementAt(L*)), and another one that create lists with random numbers from 0 to 100.

**Pre-methods:**

For *GetLength(L)* where L is the list it receives.First, I initialized a counter that add 1 each time the list is not empty and made a loop that stops whenever the element is empty. Finally it just returns the counter. For *ElementAt(L, n)* L is the list it receives and n is the place of the item that we want to return. If the reminder of length of the list is equals to 1 then open a for loop that at the end will return the head of the list and if this is not the case then it will return the item after the head. Finally, I made the *CreateList(L, x)* method where L is the List and x is the number of random numbers the list will have. It just appends a x numbers into the list.

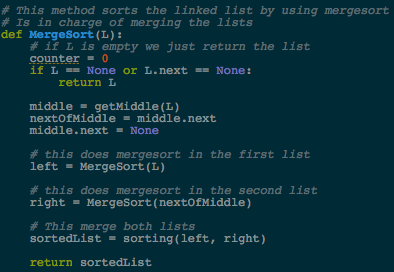
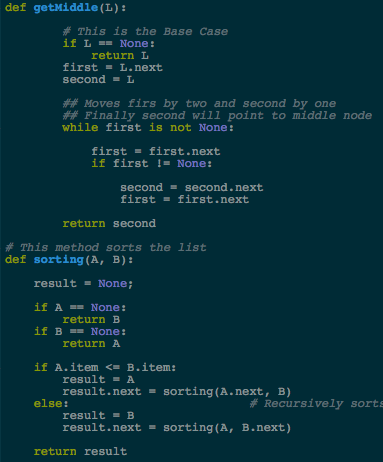
|  |
| --- |
| GetLength(L) = O(n)  ElementAt(L, n) = O(n)  CreateList(L, x) = O(n) |

****

**Merge sort:**

From my point of view, this was the hardest method because I needed to have 3 methods.

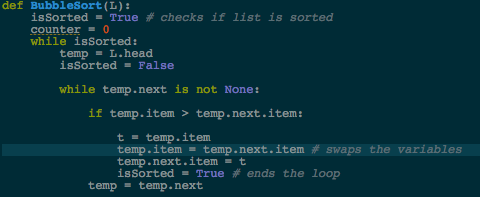
The first one is *getMiddle(L)* where L is the list it receives and it return half of the list. Secondly, I created the *sorting(A, B)* method that in this case A and B are two new lists. Its job is to sort both of them. Finally, we have *MergeSort(L)* where L is the list and its job is to merge the to lists sorted in *sorting(A, B)*

****

MergeSort(L) = O(n log(n))

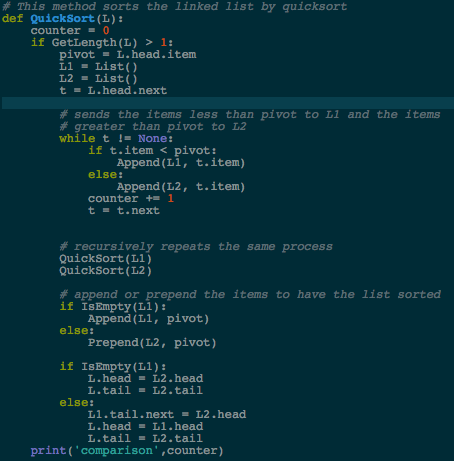
**Bubble Sort:**

In this one I created a method *BubbleSort(L)* where L is the list it receives. For this method I created a boolean variable which will stop the loop that change the variables of the items if the first item is greater than the second item. At the end the boolean variable will change to false to True to stop the loop.

 BubbleSort(L) = O(n^2)

**Quick sort:**

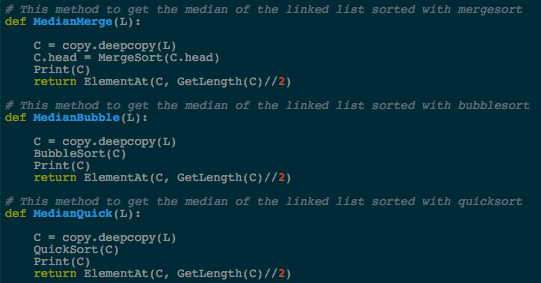
For this one, I created *QuickSort(L)* where L is the list it receives. First, if the length of the list is greater than 1 then it will create 2 lists in which we will be appending the elemtens from the original list with the purpose of having a list with the elements that are less than the pivot and the other one with the elements greater than pivot. After that, recursively, we will do the same process for both lists to have the elements separated. Once the length is less than 1 we put together bothe lists by appending the first list if it is empty and if it is not the prepending the second list, after that, is when we start checking which element is less or greater to sort them.



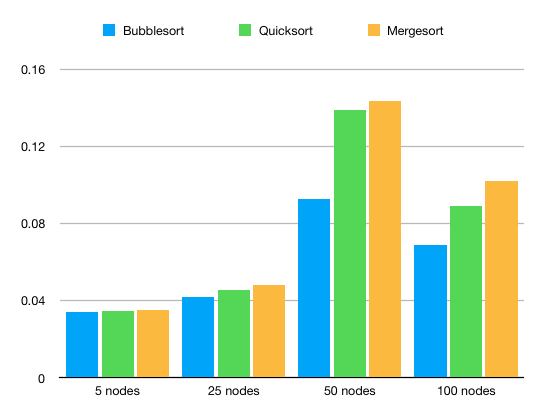
QuickSort(L) = O(n^2)

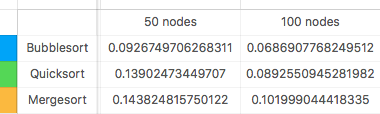
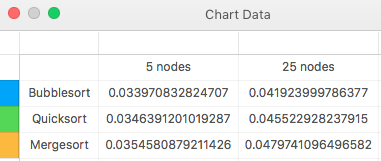
**Median:**

The final task was to create a method that returns the median of the list. This method was given in the assignment but what I did was that I created a method for each sort type. I did this with the purpose of having the outputs separated, this way I could compute different inputs and see the results. Also, to compute the comparisons I just had to create a counter that will add 1 each time it checks for the median.



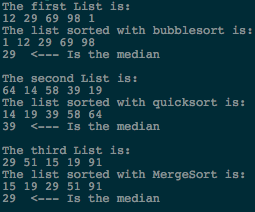
The running times for each method are the same as each specific method that is called inside.

**Running Times:**

   
**Conclusion:**

I think this lab assignment was really helpful because we had the opportunity to play and know how linked list work in a different programming language. Thanks at this assignment a know how to solve problems whenever we use linked lists.

This is my output:



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.

* David A. Davis