Lab 03 Report

In this lab we are asked to implement functions using a sorted linked list. The class obtains the following functions: Print, insert, delete, merge, index, clear, min, max, has duplicates. I implement all of the functions by using a regular linked list class that is provided to us except that the list is sorted the entire time.

The first function is the print function. The first thing that is done to implement the function is to set a temp to the head of the list. the next step is to create a while loop to iterate through every node in the list. the first line in the while loop prints the node and the next sets the temp to t.next so that it could get to the next element.

The second function asked to implement is the insert function which was a little different regarding that it needs to be sorted and efficient. The first step is to check if the head is none, if so, we return the list. we then use an elif statement to check if the head is greater than or equal to the the element i which is being inserted. If so self.head set equal to i. we then use another else statement. We set a temp to the head of the list and we iterate with a while loop through the list until current.next is not none and the next data is less than i. we use the temp to get to the next node and after the loop I = current.next since all other cases have passed.

The next function is the delete function which deletes the element I from the list. the first check makes sure that the head is none and if it is the node does not exist. We then set a temp node to an empty temp list and another temp to the head of the list. we will then use while loop to iterate until the head is none. in the loop the first check is if the data is equal to the node being deleted. If it is we will set that nodes pointer to the next.next.next node. If it is not equal we need to get tot the each of the next nodes so we set the data to the next nodes.

The next function is the index function. Which returns the of I in the list. the first is to check if the list is empty. if it is, we will return -1. We then set a temp to the head of the list and a counter. We will use the counter to find the index of I. we will then use a while loop to iterate through the list. inside the loop we will have a check to see if the nodes data is equal to i. if it is we return count. Following this we will count up one and set t = t.next to get to each index. After the loop we return count to finish the function.

The next function is clear which is relatively simple. the only step that needs to be done in this function is to set self.head = None which deletes the head of the list clearing the list.

The next function is min. this function is a bit simple regarding that the list is constantly sorted. The only step that needs to be done is to return the head of the list which will always be the head of the list.

The max function is almost the same as the min function. It is also just as simple because it is also a sorted list. so we can always return the tail of the list since it will always be the last one in the list, which is the tail.

The next function is the select function. Which returns the kth smallest element in the list. if the list is empty, we will return -1. The next step is to create a counter set to 0, and also create a temp that is pointed to the head of the list. we will then create a while loop while iterate through the list until it is none. in the loop if the first check is to see if count is equal to k in that was passed in. if it is we will print the data of the node. And in the loop we will count up plus one. And also set t to t.next to get to the next node in the list.

Function Sorted List

Print O(n) O(n)

Insert O(n) O(1)

Index O(n) O(n)

Clear O(1) O(1)

Min O(1) O(n)

Max O(1) O(n)

Select O(n) O(n)

Screenshots of the output of code



