Systems Programming
Project #2 - Sorted List

David Lambropoulos
Demetrios Lambropoulos

Professor Brian Russell
February 25, 2015

Rutgers University
Department of Computer Science
School of Arts and Sciences

1 How to Use

Given pieces of data inputted by the user and a compare and destroy function the program forms a sorted linked list contating the information.

2 Analysis

2.1 SLCreate

This function runs withing constant time, $\boxed{O(1)}$

If an the CompareFuncT or DestructFuncT functions that are passed to the SLCreate function is NULL, returns NULL. Allocate the memory required for the SortedList pointer size of the SortedList structure. Set the Comparator function to the inputed CompareFuncT. Set the Destructor function of the SortedList struct to the inputed DestructFuncT function. Set the item count in the list to 0. And finally, return the SortedListPtr to user.

2.2 SLDestroy

This function has one loop that iterates through the list and therefore has a running time of $\boxed{O(n)}$

First, check that a proper list was passed. Check that head is not null. If it is, free the memory dynamically allocated for the SortedListPtr. If the head is not null, iterate through the list freeing all dynamic allocations. Once finished free the memory allocated for head. Free the memory dynamically allocated for the head pointer.

2.3 SLInsert

This function has one loop that iterates through the list and therefore has a running time of $\boxed{O(n)}$

Check if the list or the newObj entered into the SLInsert method was NULL. Check that the head node in the SortedList is empty. If so, this will insert the given data newObj to the head of the list. Begin the infinite while loop which is used to traverse the list while still being ignorant of the size. Check if the data in the next node is smaller than the data in the node n that is passed to the function. If so replace the node.

2.4 SLCreateIterator

This function runs withing constant time, $\boxed{O(1)}$

Check if the list insert is NULL. Create an iterator

2.5 SLDestroyIterator

This function runs withing constant time, $\boxed{O(1)}$

Check if the list insert is NULL. Destroys an iterator

2.6 SLGetItem

This function runs withing constant time, $\boxed{O(1)}$

2.7 SLNextItem

This function runs withing constant time, $\boxed{O(1)}$

2.8 PrintList

This function has one loop that iterates through the list and therefore has a running time of $\boxed{O(n)}$

2.9 SLRemove

This function has one loop that iterates through the list and therefore has a running time of $\boxed{O(n)}$