Jeff DiPaola
CS 214

Assignment 2 Readme File

This readme file focuses on each of the different functions found in "sorted-list.c" and is broken up according to those functions.

SLCreate:

SLCreate uses the data structure "SortedListPtr" to create a new element, for use at the start of running the program. It takes up only the memory necessary for the new object it's creating and incorporates no loops giving it a runtime of O(1).

SLDestroy:

SLDestroy is used to remove the above created data structure from memory when it is no longer needed. It traverses through the length of the list that is passed to it, using the given destructor function to remove the void* element of the node and then uses free to remove the node itself. Once there are no nodes left, our temporary variable used as a place marker is freed and the list itself is freed. This removes the list from memory and has no overhead memory of itself. Due to the while loop traversing the length of the array it has O(n) runtime where n is the number of elements in the list.

SLInsert:

SLInsert is passed the list and an element and determines where to place that element in the list to keep the order correct and then places it in the proper location. It mallocs three nodes of its own, ptr1, ptr2, and put. These are used as pointers to the needed parts of the list and as a result are never freed. This function also contains a while loop, that in the worst case scenario (the number is the smallest) would traverse through the whole list, again giving us a runtime of O(n).

SLRemove:

Passed the list and an element, SLRemove finds the element in the list if it exists and removes it appropriately for its position in the list. It mallocs two pointers that again, due to their referencing of points in the list, are not freed. Additionally, it contains a while loop that given the worst case scenario (the element is the last one in the list) would traverse through the entire array giving us a runtime of O(n).

SLCreateIterator:

This is passed the list and uses the defined SortedListIteratorPtr data structure from the header file to create an iterator to help us step through the list. This performs one main malloc to create the iter and two others that define the size of iter's structures. None of this is freed as it is needed, and there is no loop run in this function giving a runtime of O(1).

SLDestroyIterator:

SLDestroy iterator only runs a free function before returning so it does not take up any memory and has a runtime of O(1).

SLGetItem:

Passed a current SortedListIteratorPtr, SLGetItem returns the void* value that the pointer holds or 0 if it is null. This takes O(1) time.

SLNextItem:

This does the actual traversing through the iterator that was created in SLCreateIterator. There is no memory defined here so it does not take up any more memory space. It does, however, have a while loop that will execute in the event that someone had removed an element from the array between calls of SLNextItem. If this while loop does execute, the worst case scenario would be that SLNextItem was at the second to last object before this edit happened, causing the loop to go from the start of the list to the second to last object. This would take O(n-1) time.