Assignment 2 - Sorted List
CS. 214
Likhitha Aineri, Krishna Yellayi

# Readme - Big Oh Analysis

Implementation : We used a linked list for our implementation of a sorted list. As we iterate through the list, we keep track of the changes which are made to the list. The value would be stored when we create the iterator. For the node, it is destroyed once and it won't have any pointers to it. Once the node is removed from list, the reference count is decremented.

**SLCreate :** The SLCreate creates a new empty list. It allocates the memory for the SortedList and then returns the pointer to it.

```
thing = (SortedListPtr)malloc(sizeof(struct SortedList));
```

O(1) runtime.

**SLDestroy:** The SLDestroy iterates through the list and it frees each node, until it frees itself. So of there were n ( number of elements) in a SortedList. Since it has to iterate through each Node Struct which has n operations. The other functions within SLDestroy run in a constant time.
O(n) runtime.

```
while(list->head != NULL){

            list->DestructFuncT(list->head->data);
            pointer = list->head;
            list->head = list->head->next;
```

**checkinsert :** Before inserting the new object into the list, the check insert function does error checking. It makes sure the list and new object is not empty.
O(1) runtime.

**SLInsert:** SLInsert has to iterate through the list and find the appropriate place for the new node. It allocates the memory for a node. The allocating of memory is constant. But the iteration takes linear time.
O(n) runtime.

**SLRemove:** SLInsert had to iterate through the list and find a matching node. It then has to remove all the links and decrement the refrence counter. Once that is done, the node is destroyed. The interation through the list takes linear time.
O(n) Runtime.

**SLCreateIterator:**
SortedListIteratorPtr iter = (SortedListIteratorPtr)malloc(sizeof(struct
This function creates an iterator. This function takes constant time.
O(1) Runtime.

**SLDestroyIterator:**
SLDestroyIterator frees the iterator. This function takes constant time.
O(1) Runtime.

**SLGetItem:**
SLGetItem returns the pointer data. This function takes constant time.
O(1) Runtime.

**SLNextItem:**
This function takes constant time. It deletes the node if the reference count is 0.
O(1) Runtime.