Runtime Analysis

SLCreate runs in O(1) time; always initializing an empty list takes constant time.

SLDestroy runs in O(n) time, n being the number of elements in the list. The function traverses the list once and calls the destruct function on the data of each node as it traverses.

SLInsert runs in O(n) time; worst case is if the node is to be inserted at the end of the list, meaning every element is to be traversed. One comparison is made for each node over the traversal, and a few pointer assignments to actually traverse, resulting in O(n) runtime.

SLRemove runs in O(n) time, the worst case being if the last node from the sorted list is to be removed, such that every element in the list must be traversed before removing the last element.

SLCreateIterator runs in O(1) time. The iterator that is made always points to the headnode, so input size does not affect SLCreateIterator.

SLDestroyIterator runs in O(1) time. There are a few checks made to see if decrementing the node the iterator points to necessitates destruction of the node, but input size does not affect the function.

SLGetItem runs in O(1) time. The function returns data encapsulated in the node structure, irrespective of input size.

SLNextItem runs in O(1) time. The function looks only at two nodes, regardless of input size. If decrementing the reference count of the node the iterator previously pointed to