The SortedList ADT

search() function:

Time > The search function in SortedList performs a binary search with comparison as the main performance. The code in the while loop takes constant time, that is, O(1) for every iteration and the size of input is the number of items in the list (n). After each iteration the length of the list is divided by 2 which means that after some amount of iterations, k, the size of the list is $n / 2^k$. Once there have been k

iterations, the size of the list reduces to 1 and we get which becomes then applying log base 2 on both ends gives resulting in . The best-case scenario is that the element to be found is the middle index and will only take one iteration giving a complexity of O(1). The average and worst-case complexities are the same; that is O(logn).

Space > There is no additional space used in this function, so the space complexity is constant – O(1).

Constructors , destructors : SAME AS THE LIST ADT

getOrder() :

Time >The function getOrder simply returns the boolean value of order , so it is constant//

Space > The only memory required in the function is the return statement , so the space complexity is constant

append() , insert() (both appending an item or a list) :

Time > These functions require a shift of all items in the list , so the time complexity is linear.

Space >The space complexity of the function is constant as it only requires a constant amount of memory even if the expand function is called.

remove():

Time > The time complexity of the function is linear.If all is set to true, the function would count the number of occurrences of the value to be removed, then shift the elements of the list down by the number of occurrences , otherwise , the function would simply call the remove function by index, which also has a time complexity of O(n).

Space > The space complexity is constant only countable number of temporary variables are created in the function.The shrink function will have limited impact on the remove function since it is called only if the size of the list is smaller than the capacity by a certain threshold.

Operator = , - , + :

Time > The time complexity is linear every item in the list is visited once.

Space >The space complexity is linear because a new array was allocated to contain the item in the 'obj' object.

Move assignment operator :

Time > The time complexity is constant since it is only swapping the content of the two objects which can be done in constant time.

The space complexity is also constant because no (new)excess memory has been allocated.

Operator —(postfix):

The time complexity and space complexity is constant because it is doing a constant operation to the object