Ian Chadwick
U90596416

Programming Question 5a)

I used a modified counting sort algorithm because we know that the input must be one of 26 lowercase letters.

1. I created a comparison string of the alphabet and an integer "count" array of size 26 initialized with 0s to hold the counts of each letter. O(1)
2. Then I looped through the input string once for each letter of the alphabet and stored the count in the count array this ends up being 26 * 0(n).
3. Then I looped through the count array and pick the smallest value to put into the output string, with alphabetical order being the tiebreaker. O(1)
    a. I used a while loop to keep adding letters until the count at that index is 0. O(1)
4. Once the loop checked the entire count array, exit the loop and return the string with the sorted letters in the correct order. O(1)

Time complexity: 26* O(n) + 4*O(1) = O(n)
Space complexity: n + 1 array size 26 + 1 string size 26 = O(n)

Since we must traverse the entire input string at least once because there is no other way to guess the input based on the given information, that automatically makes the best run time 1* O(n). Since we're looping through input string n 26 times that also makes the algorithm in O(n). The space used is a fair tradeoff for the speed with only an additional string of size n, a string of size 26, and an integer count array of size 26 added. This means that the space complexity is O(n).

In order to reduce the space, you could sort it in place, but that would come at the cost of increasing the runtime significantly because you would have to do a comparison sort, the best of which runs in O(nlogn).