# CS 0445 Spring 2026
# Recitation Exercise 5

**Introduction:**

Now that we have been discussing asymptotic analysis it is useful to try an empirical study to confirm (or possibly not confirm) what asymptotic analysis tells us about some implementations.

Recall that the String class in Java is an immutable class – meaning that once a String object is created it cannot be changed. Thus, modifications to Strings in Java must be achieved by creating new objects, copying all of the relevant characters from the old String into the new one.

For example, consider the statements below:

```
String S = new String("");
for (int i = 1; i <= N; i++)
     S = S + 'A';
```

At first glance it appears that this code should have a run-time of O(N), since we are adding N characters to the end of the initially empty String. However, since Strings are immutable, in fact what occurs is the following:

| | |
|---|---|
| Iteration 1: New String created with 'A' added to it -> | 1 total assignment |
| Iteration 2: New String created; copy 1 'A' and add a new 'A' -> | 2 total assignments |
| Iteration 3: New String created; copy 2 'A's and add a new 'A' -> | 3 total assignments |
| … | |
| Iteration N: New String created; copy (N-1) 'A's and add a new 'A' -> | N total assignments |

Total assignments required: $1 + 2 + \ldots + N = N(N+1)/2 \rightarrow O(N^2)$

This is a very poor asymptotic run-time, which is why we have mutable Strings such as the StringBuilder (and your Assignment 2 class, MyStringBuilder).

Similarly, removing a character from a String would require a new String to be generated containing the remaining characters. Consider the special case of removing the last character (character N-1) in a String. This would require a new String to be created which contains characters 0 to N-2. This can be done using String methods but the time for each operation should be proportional to the size of the resulting new String. Thus, if we systematically remove the last character from the end of a String until the String is empty, the analysis would be as follows:

| | |
|---|---|
| Iteration 1: New String created which is substring 0 to N-2 -> | N-1 assignments |
| Iteration 2: New String created which is substring 0 to N-3 -> | N-2 assignments |
| Iteration 3: New String created which is substring 0 to N-4 -> | N-3 assignments |

Total assignments required: $(N-1) + (N-2) + \ldots + 1 = (N-1)(N)/2 \rightarrow O(N^2)$

In order to actually delete the characters in the way described, you will need to look up some methods within the String class. Browse the class and figure out the method calls that would be appropriate for this operation. See: https://docs.oracle.com/en/java/javase/20/docs/api/java.base/java/lang/String.html

In this exercise you will empirically verify the run-times of the processes above in the following way:

Start with an empty String variable
Start the timer
Add N characters to the end of the String, one character at a time, within a loop using the + operator
Stop the timer and note the total result and result per add

Start the timer
Remove all of the characters from the String, one at a time, within a loop, always removing the last character
Stop the timer and note the total result and result per remove

To see how the time changes with the problem size, you should have different values for N. Use the following values for N: {50000, 100000, 200000, 400000, 800000}. Run the above code segments for each value of N and see how the run-times change as N increases. Based on the asymptotic analysis from above the times for all N operations should grow at $O(N^2)$ and thus the average times per single operation should grow at O(N). Since N is doubling with each new test, an O(N) time would see the run-time also double (approximately) and an $O(N^2)$ time would see the run-time increase by a factor of $2^2$ or by 4 times. See if your results are consistent with these asymptotic times.

For help with this exercise, see the Assig2B.java program from Assignment 2. This program is testing your MyStringBuilder but the approach used and the timing for this exercise will be similar.

As usual, if you are able, please volunteer to demonstrate and explain your solution to the rest of your class at the end of recitation. If you are unable to complete the exercise, you may view my solution after it is posted onto Canvas.