Unit 7 Algorithmic Enhancements

**Corey Crooks**

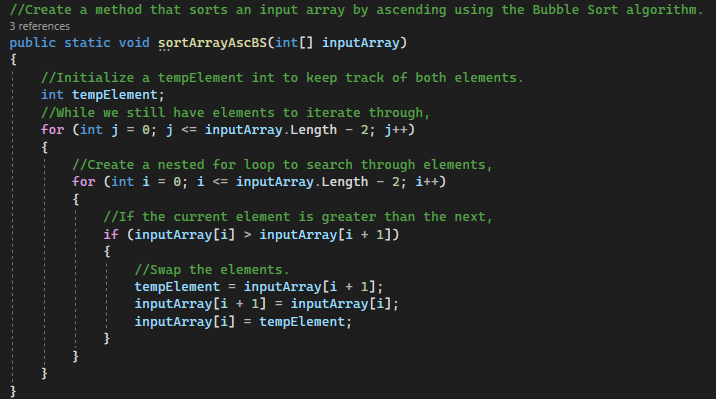
**Purdue University Global**

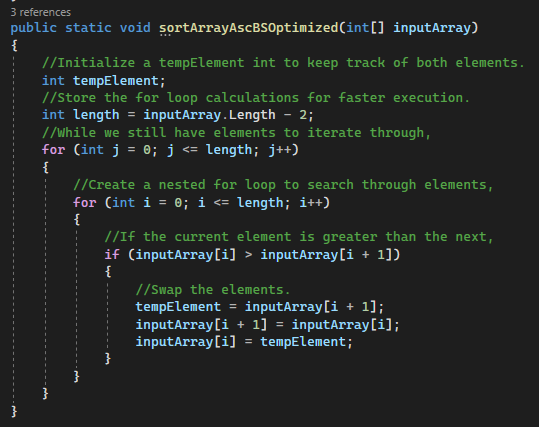
**IT481 – Advanced Software Development**

**Ahmad Kassem**

**June 7th, 2023**

Interestingly, I have found mixed results for my algorithm. I have implemented a bubble sorting algorithm that sorts an input integer in ascending order from smallest to greatest values. In my original algorithm, I had two for loops that used an upper-bound of the inputArray.Length - 2. In my optimized sorting method, I instead stored inputArray.Length - 2 in its own int variable called “length”, and then used the variable for the upper-bound in the for loop. Theoretically, this would aide the processor in not needing to perform this calculation every single iteration of each of the for loops, and would instead use a single value that could just simply be read with no calculation required.





Using the large data set as a testing ground, I implemented an array of 10,000 random integers and sorted using both methods while starting, stopping, and resetting a stopwatch to time the execution in fractions of seconds. The large data set proved that my calculation move was successful at reducing the execution time by a substantial amount. Specifically, the large data set could be sorted through in .3728855 seconds with the original algorithm, while the optimized method takes only .2285779 seconds. This is only 61% of the total execution time of the original algorithm to make a nearly 40% reduction in execution time. To think that something as simple as making a single calculation into a variable could benefit performance to such a staggering degree is outstanding. Although given that almost all of the sorting method’s code is stored in those two for loops, it does make sense. By changing the arguments of the for loop, I have interacted with a large portion of the code’s overall execution.

The small data set of just 10 integers proves a different story. The execution time of the original algorithm was .002030 seconds while the new algorithm took .0013439 seconds. I presume the operating efficiency was not worth needing to declare space in the computer’s memory just for the sake of storing a variable. The medium dataset of 1000 does actually provide minimal improvements, however. The original algorithm makes quick work of sorting in just .0029766 seconds. But the “optimized” sorting method makes quicker work in only .0020573 seconds.

