Matthew Bolda

ECE 368

9/19/19

Project 1: Milestone 1

For this project we will be learning how about Shell Sort and how it improves Insertion Sort. In order to fully comprehend how this effects performance we will be implementing our own versions of Shell Sort as well as Improved Sort that uses Bubble Sort.

As I understand it, Shell Sort takes in a larger array and "shells" it into many much smaller arrays. These small arrays are then sorted using a different sort method, in our case we will be using Insertion Sort to sort the smaller arrays. Once small arrays are sorted more "shells" are created that are even smaller and the sorting continues. Finally, the shells will compare just two elements and it will complete.

That leaves us with an improved Bubble Sort. The original Bubble Sort compares 2 numbers, it takes the larger number and compares it to the next number until the largest number has "bubbled" all the way to the last spot in the array. There are a few things you can do to help increase the speed of bubble sort. The first thing I can think of is to make it that if the first run through happens to be sorted then it will not need to run through any more times and will just return the already sorted array. Another idea that could help is to make sure to not compare between the last few elements. This is because every time you run through the largest element should go to the last spot, in that case we do not need to compare with that location ever again because it is already the largest element.

Creating the original sequence could prove to be quite difficult. From the research I've done it looks like taking the prime factors of the elements as we go seems like the proper way to do this. Doing the sequence in reverse seems to be the same as doing it forward. For every number see if it is a smooth number, that is divisible by two and three until it is one. If it is one then it's a smooth number but if it is not then it is not a smooth number and we just decrease by one until it is or we have reached one.

Let's look at the functions we must create. The first two are the easiest as Load_File is just file with the unsorted array and the size of that array. This is something similar to what we've done in ECE 264 so I do not think this will be a problem. Similarly Save_File is also like things we've done in 264 so I do not see a problem there. These functions will help us do Save_Seq1 and Save_Seq2 which will help the grader. The only other functions are the most important ones. The Shell Sort I feel will be much more difficult compared to the Bubble Sort. Like I explained in the previous paragraphs, Bubble Sort is quite simple and the process does not really change much to sort. Shell Sort also does not change the process but it does change the amount of elements that are compared at a time and that can be harder to understand. Because of this I will have to do more research on how to properly implement and improve the Shell Sort method.