Matthew Bolda (PUID 0028979907)

ECE 368

10/1/19

Project 1 Report

In this project we were to sort an array using two different methods: Shell Insertion Sort or Improved Bubble Sort. To begin let's talk about the Shell Insertion Sort. Shell sort implements gaps to sort smaller arrays and combine that into the full array. To find the proper gaps we used 3-smooth numbers or $(2^p)*(3^q)$. We had to create this sequence on our own using the size of the array that needed to be sorted. I implemented a power(a,b) function that helped in this process. I believe the time complexity of creating Seq1 to have a best case of O(1) for an array the size of 1, and a worst case of $O(n^2)$. I believe it is $O(n^2)$ because I had to first get the size of the sequence, malloc for that size, and then insert the 3-smooth numbers into the array using nested for loops. For the second sequence we had to create we were looking at an array that every element is the floor of the previous element divided by (1.3). This sequence was far easier to implement and seems to have a best case of O(1), again if the array is size 1, and a worst case of O(n). The reason for O(n) is because there are no nested loops, so the heaviest task is a single for loop that initializes the array.

I have attached a table comparing time, number of moves, and number of comparisons for the few sample tests we were given. Immediately you can see that Bubble Sort uses comparisons more, but Insertion makes a significantly larger number of moves. The way I implemented Insertion Sort was two nested while loops with multiple for loops, because of this it would seem to have a time complexity of $O(n^3)$ as the depth of nested loops never reaches more than three. For Bubble Sort, I did not stray away from the usual Bubble Sort implementation and allowed the gap sequences to do most of the work. This implementation required two loops, but only one of them nested, so the time complexity is $O(n^2)$. After my two loops I still had a few unsorted elements, so I ran through the array one more time with a gap of one to smooth out the unsorted elements.

Type of Sort	# of elements	Time (s)	# Moves	# Comparisons
Bubble	1000	0	13197	23702
Insertion		0	71178	9264
Bubble	10,000	0	186408	326725
Insertion		0	1226764	125338
Bubble	100,000	0.01	2438928	4066741
Insertion		0.09	19141156	1930330
Bubble	1,000,000	0.17	30144183	49666762
Insertion		1.44	273030330	25056024