Ben Haase

CS 3130

Project 1

**Part A3**

For each of these sorting algorithms the predicted average time complexity is around O(n2). Bubble Sort and Cocktail Sort are very similar and we expect to see a best case of O(n) and worst/average case of O(n2) which is what the test data indicates. Using T(2n)/T(n) for the times 1000/500 and 5000/2500 one would expect to see a value around 4 since (2n)2/(n)2 = 4n2/n2=4. Bubble Sort's ratios are fairly close to this at 3.7 for the 1000/500 sample and 4.03 for the 5000/2500 sample. The 3.7 ratio can be explained by the smaller sample size, but still falls in the range of O(n)-O(n2) so it is not wildly inaccurate. Cocktail Sort's ratios are dead on at 4 for both values. As expected Cocktail Sort's times came in lower than Bubble Sort's due to sweeping both ways allowing it to quickly adjust elements located far away from their ordered positions.

For Shell Sort we expect to see a best case around O(nlogn) with an average/worst case around O(n2). Given this the results seem reasonable. For Shell Sort with a gap of n/2 and decreasing each time by a factor of 1/2 results for 3.83 for the 1000/500 ratio and 3.994 for the 5000/2500 ratio. Shell Sort with a gap of n/2.2 decreasing by a factor of 1/2.2 and adding one to even results showed a ratio of 4 for 1000/500 and 4.02 for 5000/2500, which is completely reasonable. Shell Sort with n/2.2 odd had outperformed Shell Sort n/2 for larger values of n as was expected. Interestingly Cocktail Sort and Shell Sort n/2.2 odd had almost identical run times. As shown in the graphs all of these algorithms show a quadratic curve in their graphs, which is again what we would expect.

**Part B3**

Timing both of these search algorithms was a fairly difficult task since the execution times on such small input sizes where at the highest xE-5. To be able to register passage of time with the timer, each search was executed 50,000 times between the timer start and stop. The resulting time was then divided by 50,000. This does not seem to have affected the integrity of the data in a significant way

For Linear Search we would expect the time complexity at best to be O(1) and at worst to be O(n) with an average somewhere around n/2 leaving us with O(n). With that the predicted value of the ratio T(2n)/T(n)=2n/n=2 which is exactly what the data indicates with a ratio of 2.18 for 1000/500 input size and 2.07 for 5000/2500. The graph of Linear Search shows a straight line which is what we would expect to see.

For Binary Search the expected time complexity is at best O(1) and a worst/average case of O(logn). Our predicted ratio is T(2n)/T(n)=log(2n)/log(n) this function approaches 1 as the input size increases, which is what the sample data shows. For 1000/500 a ratio of 1.2 was obtained and for 5000/2500 a ratio of 1.06 was obtained indicating that it is indeed going to 1 as expected. The graph for Binary Search shows the slope decreasing as the input size is increased which is again what we would expect to see.