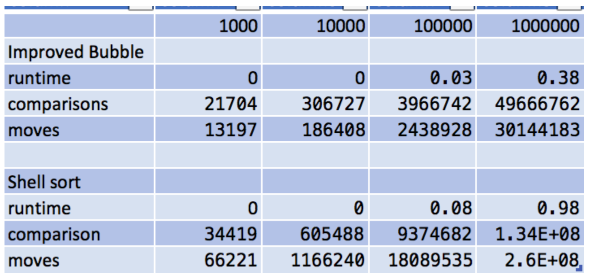
ECE 368
Bryan Settles

My algorithm works with two helper functions. One swaps two values, and the other is to get the correct gap value for the improved bubble sort. The shell sort first calculates an array of the gap values, and then performs the sort using those values. The improved bubble sort gets the gap values while it is in the process of performing the sort. I optimized the swap helper function by using XOR instead of just the traditional pointer. I also used the built-in function 'qsort' when needed to sort the gap values. The time-complexity, for the worst-case scenario, of the improved bubble sort is O(n^2). However, the average case is O(n^2/(2^p)), where p is the number of increments. The shell sort, with the best-known gap sequence of 2^p*3^p, has a worst-case runtime of O(n*log^2(n)).

My code to generate the first sequence had a loop within a loop to find all the necessary numbers for the gap sequence, a call to 'qsort' to organize them, and a final loop to print the sequence out. The inner loop is dependent on the outer loop for deciding how many times it will run. In total, together they run in O(log(n)^2) time, where n is the number of input values, which greatly outweighs the number in the gap sequence.

My code to generate the second sequence had one loop to go through the input and find the gap values, a call to 'qsort' to organize them, and a final loop to print out all the values. The process of looping through all the input values, each time dividing the gap value by 1.3, brings the runtime to approximately O(log(n)^2), with n being the number of input values. Because the number of input greatly outweighs the number in the gap sequence, this code is defined by O(log(n)^2). The space complexity of both sequence functions is O(n) because of the array that must be allocated in order to store all of the gap values.

The runtimes for the improved bubble sort and shell sort are listed in the table below. The improved bubble sort ended up being significantly faster than the shell sort. This is likely because of the increased time that it takes to calculate the sequence values in the shell sort. The space complexity for the improved bubble sort is O(1). The shell sort, with the best-known 2^p*3^p gap sequence, has a worst-case space complexity of O(n), because of the process of breaking the array down into subarrays. This could be another reason for the increased runtime of the sort.



| | 1000 | 10000 | 100000 | 1000000 |
|---|---|---|---|---|
| Improved Bubble | | | | |
| runtime | 0 | 0 | 0.03 | 0.38 |
| comparisons | 21704 | 306727 | 3966742 | 49666762 |
| moves | 13197 | 186408 | 2438928 | 30144183 |
| | | | | |
| Shell sort | | | | |
| runtime | 0 | 0 | 0.08 | 0.98 |
| comparison | 34419 | 605488 | 9374682 | 1.34E+08 |
| moves | 66221 | 1166240 | 18089535 | 2.6E+08 |