ECE Programming Assignment 1

Eric Murphy

February 18, 2014

Time and Space Complexity

The time and space complexities for my program are given as follows:

$$T_{time}(n) \in \Omega(n^2) \qquad T_{space}(n) \in \Omega(n^2)$$

Space Complexity

The space complexity of my program is n^2 because the number of rows in the pyramid created for the sequence is n, the size of the data set, divided by three. From this, one can obtain the total number of elements in by using the equation:

$$\#\ of\ items = \frac{n(n+1)}{2},\ where\ n\ =\ the\ data\ set\ divided\ by\ 3$$

From this, one can obtain that the space complexity. The size of the data set divided by three is also equal to the number of rows that will be in the complete pyramid.

Time Complexity

Similar to the space complexity, the time complexity of my program is n^2. My time complexity of my program is as such because the number of elements in my sequence (n^2) are iterated over. Just as my sequence took up n^2 spaces, my time complexity is the same because n^2 elements are placed in to my malloc'ed array.

Run-Time Analysis

Running my code on a data set of 10000000 elements, I obtain a run-time analysis of:

Number of comparisons: 2.541659e+07
Number of moves: 1.899581e+08
I/O time: 1.000000e+00
Sorting time: 8.412430e+05

Running my code on a data set of 1000000 elements, I obtain a run-time analysis of:

Number of comparisons: 1.944636e+06
Number of moves: 1.364907e+07
I/O time: 0.000000e+00
Sorting time: 5.848500e+04

As one would notice, the number of comparisions and moves are each increased by around a factor of 10. Knowing this, I can deduce that my time complexity of n^2 is correct.

Additional Space Complexity

The space complexity of the rest of my program is:

$$T_{loading} \in \Omega(n)$$

The reason for the space complexity of the rest of my program being n is that the data set size is loaded from the input file and stored in an array. each item from the input file, aside from the first item which specifies the data set size, is stored.

Selection sort works so inefficiently with shell sort is because in selection sort, each element is checked to determine which element in the array is the largest. Since each element has to be checked, it can require a large amount of time when parsing large data sets. Selections sort needs to be optimized in order to allow for large quantities of data to be used in a shell sort with an inner selection sort.