Geoffrey Cramer
ECE368 – Summer 2017

Programming Assignment 1

Generating the Sequence:

The number of elements in a $2^p 3^q$ tree for an input of size n is:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = O(n^2)$$

This means the space complexity of the sequence generation is $O(n^2)$. To generate the sequence, each level should be iterated and each element within the given level should be checked. This resulted in a double for loop which indicates an $O(n^2)$ time complexity.

Results:

Insertion Sort

Size	# of Comparisons	# of Moves	I/O Time	Sorting Time
15	90	165	0	0
1,000	35,498	66,453	0	0
10,000	621,820	1,172,531	0	0
100,000	9,610,236	18,215,650	.01	.03
1,000,000	137,466,600	261,453,700	.02	.45

Selection Sort

Size	# of Comparisons	# of Moves	I/O Time	Sorting Time
15	233	225	0	0
1,000	1,474,965	92,865	0	0
10,000	149,603,100	1,652,133	0	0.15
100,000	14,994,190,000	25,816,230	0	15.74
1,000,000	1,499,920,000,000	371,961,500	.02	1684.320

It seems that the insertion sort has a time complexity of $O(n)$ because it is best for "small" cases. Even a size of 1,000,000 seems to still be "small" for this particular algorithm as it is highly optimized (taken directly from class notes). However, the selection sort time complexity is clearly $O(n^2)$ (multiplying n by 10 multiplies the total time by 100) and takes significantly longer, primarily because it is not optimized. Both algorithms have $O(1)$ space complexity as they both need a standard set of tracking variables independent of the size of n.