Lab 6

Selection Sort

List Size	Comparisons	Time (seconds)
1,000 (observed) -	499500	0.06634879112243652
2,000 (observed) -	1999000	0.20564985275268555
4,000 (observed) -	7998000	0.7249789237976074
8,000 (observed) -	31996000	2.8059098720550537
16,000 (observed) -	127992000	11.138371706008911
32,000 (observed) -	511984000	45.069738149642944
100,000 (estimated) -	4095872000	306
500,000 (estimated) -	102396800000	7650
1,000,000 (estimated) -	409587200000	30600
10,000,000 (estimated) -	4095872000000	3060000

Insertion Sort

List Size	Comparisons	Time (seconds)
1,000 (observed) -	247986	0.05938577651977539
2,000 (observed) -	1018717	0.19080901145935059
4,000 (observed) -	3995264	0.6690669059753418
8,000 (observed) -	16112194	2.6035640239715576
16,000 (observed) -	64667449	10.494043111801147
32,000 (observed) -	257507119	42.00761413574219
100,000 (estimated) -	100000000	128
500,000 (estimated) -	400000000	512
1,000,000 (estimated) -	1600000000	2048
10,000,000 (estimated) -	6400000000	8192

1. Which sort do you think is better? Why?

I think insertion sort seems to be better. One of its advantages is the best case O(n) versus the best case O(n^2) for selection sort. It also sorts the the correct location rather than using a comparison method like selection sort.

2. Which sort is better when sorting a list that is already sorted (or mostly sorted)? Why?

Insertion sort is better for sorting a list that is already sorted, because it waits to make a swap unlike selection sort. It will have to do far less swaps over time.

3. You probably found that insertion sort had about half as many comparisons as selection sort. Why? Why are the times for insertion sort not half what they are for selection sort? (For part of the answer, think about what insertion sort has to do more of compared to selection sort.)

Insertion sort has to has to move backward through the list to find the spot the item is supposed to go. It also has to do more list transversal. This takes more time.