Tyler Baxter

CPE-202

05/02/2022

Lab 6

Selection Sort		
List Size	**Comparisons**	**Time (seconds)**
1,000 (observed)	499500	0.03651094436645508
2,000 (observed)	1999000	0.11082077026367188
4,000 (observed)	7998000	0.40855908393859863
8,000 (observed)	31996000	1.5979969501495361
16,000 (observed)	127992000	6.250909805297852
32,000 (observed)	511984000	25.34986972808838
100,000 (estimated)	4999950000	253.184
500,000 (estimated)	124999750000	6542.488
1,000,000 (estimated)	499999500000	26546.533
10,000,000 (estimated)	49999995000000	2.784 x 10^6

Insertion Sort		
List Size	**Comparisons**	**Time (seconds)**
1,000 (observed)	247986	0.04208827018737793
2,000 (observed)	1018717	0.13818693161010742
4,000 (observed)	3995264	0.5060708522796631
8,000 (observed)	16112194	1.9940118789672852
16,000 (observed)	64667449	7.884877920150757
32,000 (observed)	257507119	31.641596794128418
100,000 (estimated)	2499975000	310.885
500,000 (estimated)	62499875000	7844.676
1,000,000 (estimated)	249999750000	31505.215
10,000,000 (estimated)	24999997500000	3.193 x 10^6

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

I believe that Selection Sort is better than Insertion Sort. I think this because, although insertion sort (on average) makes less comparisons, it took much longer to execute when the list size get very large.

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

It is better to use an Insertion Sort when it comes to sorting a list that is already, or mostly already, sorted. This statement is true because in the best case scenario, which is an already sorted list, it takes much less time (o[n]) for insertion sort to execute rather than selection sort [n^2].

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.)

The amount of comparisons done by insertion sort was roughly half of the amount done by selection because in the average case, insertion sort did n(n-1)(1/4) comparison while selection sort did n(n-1)(1/2) comparisons. This is because in insertion sort, once the value knows there are no smaller values left with to compare, it goes into the next unsorted number rather than continuing through the entire list. However, the times for insertion sort are not half of what they are for selection sort because insertion sort must do more swaps than selection sort. Selection sort only swaps the smallest with its correct position, while insertion swaps almost every time it compares.