|  |  |  |
| --- | --- | --- |
| **Selection Sort** | | |
| **List Size** | **Comparisons** | **Time (seconds)** |
| **1,000 (observed)** | 499500 | 0.07480931282043457 |
| **2,000 (observed)** | 1999000 | 0.29620885848999023 |
| **4,000 (observed)** | 7998000 | 1.1729109287261963 |
| **8,000 (observed)** | 31996000 | 4.710436820983887 |
| **16,000 (observed)** | 127992000 | 18.664280891418457 |
| **32,000 (observed)** | 511984000 | 78.42795205116272 |
| **100,000 (estimated)** | 4999843750 | 765.89796925 |
| **500,000 (estimated)** | 124996093750 | 19147.4492312 |
| **1,000,000 (estimated)** | 499984375000 | 76589.796925 |
| **10,000,000 (estimated)** | 4.9998438e+13 | 7658979.6925 |

|  |  |  |
| --- | --- | --- |
| **Insertion Sort** | | |
| **List Size** | **Comparisons** | **Time (seconds)** |
| **1,000 (observed)** | 247986 | 0.0608367919921875 |
| **2,000 (observed)** | 1018717 | 0.27550816535949707 |
| **4,000 (observed)** | 3995264 | 1.035517692565918 |
| **8,000 (observed)** | 16112194 | 4.033452033996582 |
| **16,000 (observed)** | 64667449 | 16.525542736053467 |
| **32,000 (observed)** | 257507119 | 68.4173698425293 |
| **100,000 (estimated)** | 2514717958.98 | 668.138377368 |
| **500,000 (estimated)** | 62867948974.6 | 16703.4594342 |
| **1,000,000 (estimated)** | 251471795898 | 66813.8377368 |
| **10,000,000 (estimated)** | 2.514718e+13 | 2.514718e+13 |

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

I think that insertion sort is better because on most of the cases it tends to be faster. Selection sort always has a n(n-1)/2 time complexity while in insertion sort, the worst case scenario is that it has a n(n-1)/2 time complexity. Essentially, Insertion sort has the chance to run faster most of the time so I believe it is the better sorting system.

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

Insertion sort is definitely better because unlike selection sort which always has to check all of the unsorted elements, when insertion sort runs, it does not need to iterate over every element and thus making less comparison which ensures a faster runtime.

1. 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.)

For insertion sort comparisons, on the best case they make 1 comparison while on the worst case it makes the same amount as selection sort. Thus, on average, it makes about half the amount of comparisons as selection sort. As for the times, the times were not exactly half because insertion sort has a lot of swaps which take up more time than selection sort. Thus, despite making about half the comparisons, the time is not half.