|  |  |  |
| --- | --- | --- |
| **Selection Sort** | | |
| **List Size** | **Comparisons** | **Time (seconds)** |
| **1,000 (observed)** | 4999500 | .04 |
| **2,000 (observed)** | 1999000 | .13 |
| **4,000 (observed)** | 7998000 | .48 |
| **8,000 (observed)** | 31996000 | 1.9 |
| **16,000 (observed)** | 127992000 | 8.2 |
| **32,000 (observed)** | 511984000 | 34.36507248878479 |
| **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 | 0.043473243713378906 |
| **2,000 (observed)** | 1018717 | 0.13045167922973633 |
| **4,000 (observed)** | 3995264 | 0.4594452381134033 |
| **8,000 (observed)** | 16112194 | 1.997429609298706 |
| **16,000 (observed)** | 64667449 | 8.247373104095459 |
| **32,000 (observed)** | 257507119 | 39.929078102111816 |
| **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?

Insertion. Though better in my runs (no idea why), it has fewer comparisons over time. One of the advantages of insertion sort is the best case O(n) complexity vs the best O(n^2) complexity of selection sort. Insertion sort sorts to the final location of an object compared to selection sort, which switches comparatively. One of the advantages is that insertion sort’s list becomes sorted which makes it faster than the unsorted nature of the selection sort.

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

Insertion sort, it does a better job because it doesn’t swap as soon as it finds an item, it waits. This makes it better for nearly sorted lists.

Insertion sort because it sorts objects into their final place. This means it will have to do far less swaps over time compared to the selection sort which comparatively sorts the list. One interesting advantage of this (I assume) would be in large datasets, where when an item is inserted it can be moved to it’s final location quicker.

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.)

My times are pretty similar. This is because insertion sort has to do more list traversal, to find the proper place for the item. It has to move backwards through the list to find the location where the item should be stored, for each item which consumes time.