|  |  |  |
| --- | --- | --- |
| **Selection Sort** | | |
| **List Size** | **Comparisons** | **Time (seconds)** |
| **1,000 (observed)** | 499500 | 0.04357767105102539 |
| **2,000 (observed)** | 1999000 | 0.16928935050964355 |
| **4,000 (observed)** | 7998000 | 0.6321229934692383 |
| **8,000 (observed)** | 31996000 | 2.7773544788360596 |
| **16,000 (observed)** | 127992000 | 11.388050079345703 |
| **32,000 (observed)** | 511984000 | 53.04303431510925 |
| **100,000 (estimated)** | 4999843750 | 517.998 |
| **500,000 (estimated)** | 1.2499 x 10^11 | 12949.95117 |
| **1,000,000 (estimated)** | 4.9998 x 10^11 | 51799.80468 |
| **10,000,000 (estimated)** | 4.9998 x 10^13 | 5179980.468 |

|  |  |  |
| --- | --- | --- |
| **Insertion Sort** | | |
| **List Size** | **Comparisons** | **Time (seconds)** |
| **1,000 (observed)** | 247986 | 0.03899073600769043 |
| **2,000 (observed)** | 1018717 | 0.20920014381408691 |
| **4,000 (observed)** | 3995264 | 0.7443361282348633 |
| **8,000 (observed)** | 16112194 | 3.058337926864624 |
| **16,000 (observed)** | 64667449 | 13.77522087097168 |
| **32,000 (observed)** | 257507119 | 56.407695293426514 |
| **100,000 (estimated)** | 2514717959 | 550.856 |
| **500,000 (estimated)** | 6.286 x 10^10 | 13771.40991 |
| **1,000,000 (estimated)** | 2.514 x 10^11 | 55085.63965 |
| **10,000,000 (estimated)** | 2.514 x 10^13 | 5508563.965 |

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

Although on average, both Selection Sort and Insertion Sort has a big O notation of O(N^2), therefore on average, they would both perform the same. However, on special occasions when a data is sorted, Insertion Sort would be better since it has a big O notation of O(N) while selection sort still has a big O notation of O(N) on time complexity.

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

Insertion Sort is better when sorting a list that is already sorted (or mostly sorted) because it would only make one comparison with the number behind it every time, cutting back on time complexity, which will maintain a linear time O(N). With Selection Sort, it would still maintain a big O(N^2) since it will always make N^2 comparison even with sorted list.

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.)

Insertion sort had about half as many comparisons as selection sort but not half the times because on occasions where the values is the lowest value in the list, insertion sort would need to make comparisons with all the values on the left, swapping each as it goes every time, which increases the time complexity.

|  |  |  |
| --- | --- | --- |
|  | **Number of Quicksort Comparisons** | |
| **Starting List** | pivot = first | pivot = median of 3 |
| Ordered, ascending |  |  |
| n = 100 | 4950 | 1763 |
| n = 200 | 19900 | 6862 |
| n = 400 | 79800 | 27061 |
| n = 800 | 319600 | 107460 |
| Random |  |  |
| n = 100 (average 10 runs) | 682 | 602 |
| n = 200 (average 10 runs) | 1476 | 1551 |
| n = 400 (average 10 runs) | 3773 | 3533 |
| n = 800 (average 10 runs) | 8215 | 8254 |
|  |  |  |
| Observed Big O() behavior, ordered with pivot = first : O(N^2) | | |
| Observed Big O() behavior, ordered with pivot = median of 3 : O(nlog(n)) | | |
| Observed Big O() behavior, random with pivot = first : O(nlog(n)) | | |
| Observed Big O() behavior, random with pivot = median of 3 : O(nlog(n)) | | |
| For random list, observation regarding using first vs. median of 3 : It is still O(nlog(n)) with random list regardless of using first or median of 3. | | |