Luis Gomez

CPE 202

Lab 6, Sorting

NOTE: I implemented my sorting algorithms using recursion and couldn’t figure out how to safely increase my recursion depth so I adjusted my test cases for lists under 1000 elements long

|  |  |  |
| --- | --- | --- |
| **Selection Sort** | | |
| **List Size** | **Comparisons** | **Time (seconds)** |
| **100 (observed)** | 4950 | 0.0003590583801269531 |
| **200 (observed)** | 19900 | 0.003576993942260742 |
| **400 (observed)** | 79800 | 0.007686614990234375 |
| **800 (observed)** | 319600 | 0.027962923049926758 |
| **16,000 (observed)** |  |  |
| **32,000 (observed)** |  |  |
| **100,000 (estimated)** |  |  |
| **500,000 (estimated)** |  |  |
| **1,000,000 (estimated)** |  |  |
| **10,000,000 (estimated)** |  |  |

|  |  |  |
| --- | --- | --- |
| **Insertion Sort** | | |
| **List Size** | **Comparisons** | **Time (seconds)** |
| **100 (observed)** | 2721 | 0.00024771690368652344 |
| **200 (observed)** | 10127 | 0.0009021759033203125 |
| **400 (observed)** | 40873 | 0.0025763511657714844 |
| **800 (observed)** | 158444 | 0.011184215545654297 |
| **16,000 (observed)** |  |  |
| **32,000 (observed)** |  |  |
| **100,000 (estimated)** |  |  |
| **500,000 (estimated)** |  |  |
| **1,000,000 (estimated)** |  |  |
| **10,000,000 (estimated)** |  |  |

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

Insertion sort is better, because it doesn’t iterate through the entire unsorted portion of the list before assigning the next element in the sorted list. Instead, the insertion process iterates through the sorted portion only, inserting the new element at the first True result of a comparison.

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

When sorting a list that is already/mostly sorted, it would be best to use insertion sort as the sorted portions of the data set will merely append to the temp list used in the process. It is only when unsorted data occurs in the list that an insertion operation is needed.

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 half as many comparisons as Selection sort because the insertion process can be altogether avoided if it is found that the next element to be sorted is greater than the last element in the sorted portion of the list. It is only when this comparison fails that the list must be iterated through to execute an insertion operation.

I would reason that the times for Insertion sort (IS) are not half that of Selection sort because IS is an n\*log(n) operation.