CS 111 Algorithms Design and Analysis

Programming Assignment

ANALYSIS:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Insertion Sort  (seconds) | Merge Sort  (seconds) | Quick Sort  (seconds) | Heap Sort  (seconds) |
| 10 | 0.0 | 0.0 | 0.0 | 0.0 |
| 100 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1000 | 0.0 | 0.0 | 0.0 | 0.0 |
| 10000 | 0.288 | 0.04 | 0.0 | 0.4 |

Table 1: Average running time for an input array that is random

Conclusion: All four sorting algorithms performed all equally fast in sorting an array of random integers that has a size of 1000 and below. When given an array of 10000 integers, Insertion Sort took 0.2s to sort the array. Merge Sort and Heap Sort both took 0.4s to sort the array probably the two runs on O(n log n). Quick Sort showed fast performance.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Insertion Sort  (seconds) | Merge Sort  (seconds) | Quick Sort  (seconds) | Heap Sort  (seconds) |
| 10 | 0.0 | 0.0 | 0.0 | 0.0 |
| 100 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1000 | 0.0 | 0.0 | 0.0 | 0.0 |
| 10000 | 0.0 | 0.04 | 1.75 | 0.04 |

Table 2: Average running time for an input array that is sorted

Conclusion: Insertion Sort performed really well with the sorted arrays. Insertion Sort only did comparisons and because it is already sorted swapping is not necessary. Both Merge Sort and Heap Sort showed consistency in terms of performance. Its performance is not affected by the input size. Quick Sort has the longest sorting time among the four. While Insertion Sort only used comparisons on sorted arrays, Quick Sort has to compare all elements(if the pivot is the highest/lowest value in the partition) to pivot and then put the pivot in its sorted place.