Alan Lai

11/30/2020

CSS 342: Program 4 Charts

*Table 1: Comparison of Time of Unique Sorts.*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Size:** | 10 | 100 | 1000 | 2500 | 5000 | 7500 | 10000 | 25000 | 50000 | 100000 | 250000 |
| Bubble | 0 | 0 | 31 | 219 | 891 | 2031 | 3516 | 21312 | 83218 | 269188 | 1321625 |
| Insertion | 0 | 0 | 16 | 63 | 282 | 641 | 1109 | 6516 | 24750 | 62656 | 178172 |
| Quick | 0 | 0 | 0 | 0 | 16 | 16 | 16 | 31 | 62 | 110 | 359 |
| Merge | 0 | 0 | 0 | 32 | 47 | 78 | 94 | 250 | 390 | 813 | 2141 |
| Iter.Merge | 0 | 0 | 0 | 0 | 0 | 15 | 16 | 16 | 62 | 109 | 201 |
| Shell | 0 | 0 | 16 | 0 | 16 | 16 | 0 | 32 | 63 | 141 | 344 |

*Table 2: Sorting Time by Milliseconds by Array Size*

*Table 3: Sorting Time by Milliseconds by Array Size (Without Bubble/Insertion)*

*Table 4: Big O Complexity of the Sorts*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Bubble** | **Insertion** | **Merge** | **Iter.Merge** | **Quick** | **Shell** |
| **Worst Case** | n^2 | n^2 | nlog(n) | nlog(n) | n^2 | n(log(n))^2 |
| **Avg. Case** | n^2 | n^2 | nlog(n) | nlog(n) | nlog(n) | n(log(n))^2 |
| **Best Case** | n | n | nlog(n) | nlog(n) | nlog(n) | nlog(n) |

**Takeaways:**

* The BubbleSort and InsertionSort are the two slowest sorting algorithms.
* IterativeMergeSort and ShellSort are the fastest sorting algorithms.
* QuickSort, MergeSort, IterativeMergeSort, ShellSort in the 100-10000 range is the most efficient
* As array sizes reach 100000 or more, MergeSort is less efficient than QuickSort and ShellSort.
* IterativeMergeSort is faster than MergeSort and ShellSort in the 100,000-250,000 array sorting range