**Big O of the Sorts**

I decided to make graphs only individually and not cumulatively because when all the sort graph data is being placed into one graph, the behavior for each of the sorts is not visible as some of the sorts vary quite a bit in their input sizes and timing.

**Bubble Sort:**

|  |  |
| --- | --- |
| Input Size | Bubble Sort Timing |
| 1000 | 0.093704 |
| 5000 | 2.228790 |
| 10000 | 8.973262 |
| 15000 | 21.075831 |
| 20000 | 38.954409 |
| 25000 | 61.677316 |
| 30000 | 89.3539513 |
| 35000 | 121.6138002 |

The bubble sort Big O is O(n^2). This is shown clearly in the graph as it increases exponentially with more input size. The exponential increase is also shown in the table of data showing the rapid increase in timing with input size increase.

Insertion Sort

|  |  |
| --- | --- |
| Input Size | insertion sort timing |
| 1000 | 0.027466 |
| 5000 | 0.680552 |
| 10000 | 2.752495 |
| 15000 | 6.513295 |
| 20000 | 11.649338 |
| 25000 | 17.969997 |
| 30000 | 25.9856661 |
| 35000 | 35.1974663 |

The insertion sort Big O is O(n^2). This is shown clearly in the graph as it increases exponentially with more input size. The exponential increase is also shown in the table of data showing the rapid increase in timing with input size increase.

Merge Sort:

|  |  |
| --- | --- |
| Input Size | merge sort timing |
| 1000 | 0.003000 |
| 5000 | 0.014766 |
| 10000 | 0.032844 |
| 15000 | 0.053555 |
| 20000 | 0.073277 |
| 25000 | 0.091993 |
| 30000 | 0.1122222 |
| 35000 | 0.1340037 |

The merge sort big O is O(n log n). This is shown clearly on the graph as it does increase nearly linearly with more input size. The near linear increase is also shown in the table of data showing the near proportionate increase in timing with input size increase.

Iterative Merge Sort:

|  |  |
| --- | --- |
| Row Labels | Iterative\_merge sort timing |
| 100000 | 0.3769911 |
| 1000000 | 5.0380997 |
| 2000000 | 11.3076263 |
| 3000000 | 18.6034597 |
| 4000000 | 25.4461141 |
| 5000000 | 34.6231913 |
| 6000000 | 43.0106017 |
| 7000000 | 53.0724751 |

The iterative merge sort big O is O(n log n). This is shown clearly on the graph as it does increase nearly linearly with more input size. The near linear increase is also shown in the table of data showing the near proportionate increase in timing with input size increase.

Quick Sort:

|  |  |
| --- | --- |
| Input Size | quick sort timing |
| 1000 | 0.000887 |
| 5000 | 0.005387 |
| 10000 | 0.012436 |
| 15000 | 0.019746 |
| 20000 | 0.025473 |
| 25000 | 0.034001 |
| 30000 | 0.0426193 |
| 35000 | 0.052194 |

The quick sort average Big O is O(n log n). Very rarely in worst case it becomes O(n^2). This graph depicts the average case nature of quick sort being nearly linear aka O(nlogn). The graph does show near linear increase as the input size increases. The near linear increase is also shown in the table of data showing the near proportionate increase in timing with input size increase.

Shell Sort:

|  |  |
| --- | --- |
| Row Labels | Shell Sort timing |
| 1000 | 0.0016685 |
| 5000 | 0.0111484 |
| 10000 | 0.0256805 |
| 15000 | 0.0433935 |
| 20000 | 0.0584216 |
| 25000 | 0.0757828 |
| 30000 | 0.0941604 |
| 35000 | 0.1135352 |
| 40000 | 0.8749791 |
| 45000 | 0.936137 |
| 50000 | 1.304069 |

The shell sort big O is an average case of about O(n^3/2), but a worst case of O(n^2). We can see in the graph how the shell sort increases originally by its average case O(n^3/2) with smaller input sizes, being closer to linear than exponential due to a very minor exponential increase. It then rapidly increases later on by almost O(n^2) showing its worst-case scenario with much larger input sizes. This change from being closer to linear to a rapid exponential increase is also shown in the table of data.