Colin Weil

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Algorithms |  |
|  | Bubble | Insertion | Merge |
| Data Types  (10) | Random | 2.666e-06 | 6.25e-07 | 1.0292e-05 |
| Reversed | 1.103e-06 | 6.07e-07 | 1.168e-06 |
| Unique | 1.16e-06 | 4.8e-07 | 1.074e-06 |
| Partial | 8.84e-07 | 3.91e-07 | 1.017e-06 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Algorithms |  |
|  | Bubble | Insertion | Merge |
| Data Types  (1000) | Random | 0.00739677 | 0.00178841 | 0.000163701 |
| Reversed | 0.00777365 | 0.00345273 | 0.000106094 |
| Unique | 0.00759355 | 0.00180927 | 0.000158939 |
| Partial | 0.0053441 | 0.000964694 | 0.000128501 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Algorithms |  |
|  | Bubble | Insertion | Merge |
| Data Types  (10,000) | Random | 0.701264 | 0.170851 | 0.00202447 |
| Reversed | 0.64887 | 0.360134 | 0.00125385 |
| Unique | 0.64244 | 0.177239 | 0.00194645 |
| Partial | 0.422891 | 0.0985599 | 0.00149257 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Algorithms |  |
|  | Bubble | Insertion | Merge |
| Data Types  (100,000) | Random | 66.5909 | 18.4924 | 0.0213185 |
| Reversed | 64.7779 | 34.6524 | 0.0125085 |
| Unique | 66.0008 | 17.327 | 0.0199621 |
| Partial | 42.3972 | 9.55998 | 0.0169787 |

Results:

For the small data sizes, all of the sorting algorithms were so fast that the difference between them was so unrecognizable. The only differences that can be seen is for the random data set where insertion did the best, then bubble, then merge. For all other data set sizes, no matter what the data type was, merge sort was significantly the best. This can be seen especially in the bigger data sets when merge sort never goes higher than a second, but the other two sorting methods go over a minute. This is because merge sort should never be longer than nlogn while insertion and bubble sort can go up to n^2, where is the size of the data set while. Also the reversed data set was the worst for both insertion and bubble because the way that the algorithms are made, every single element is looked at every single pass of the sort.