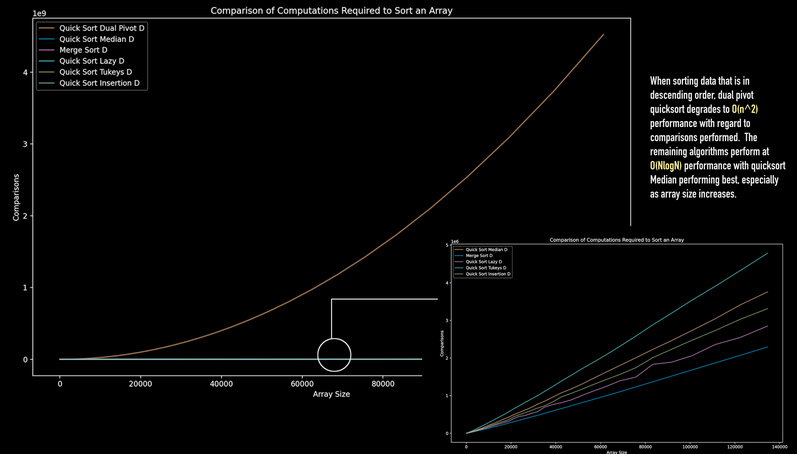
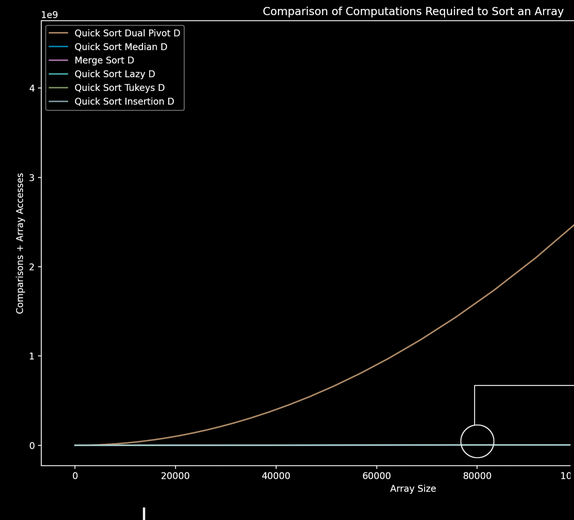
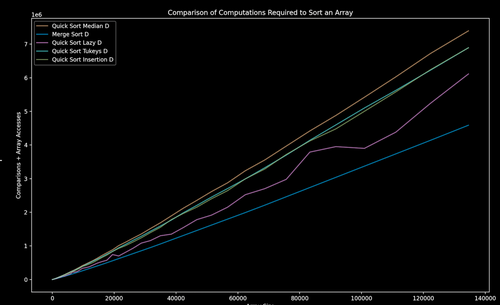
Recursive Sort Algorithm Performance on Arrays with Descending Data

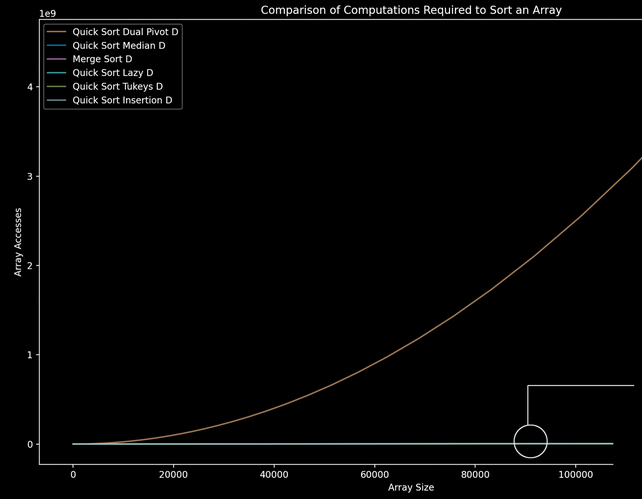


When sorting data that is in descending order, dual pivot quicksort degrades to O(n^2) performance with regard to comparisons performed. The remaining algorithms perform at O(NlogN) performance with quicksort Median performing best, especially as array size increases.

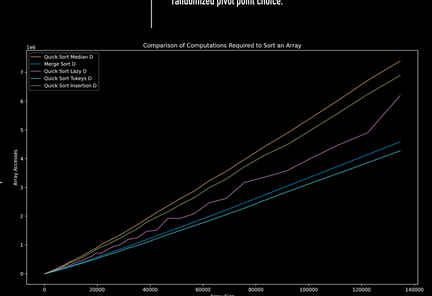


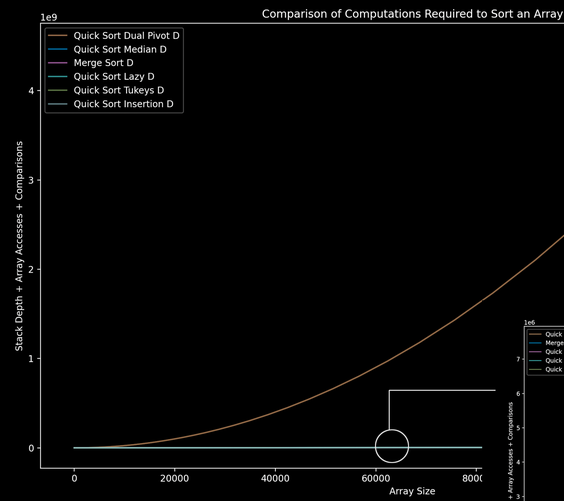
When looking at combined array accessess and comparisons, Merge Sort comes out as the best performer, especially as array size grows. Interestingly, Lazy sort (with randomized pivot value) outperforms all others after Merge Sort.



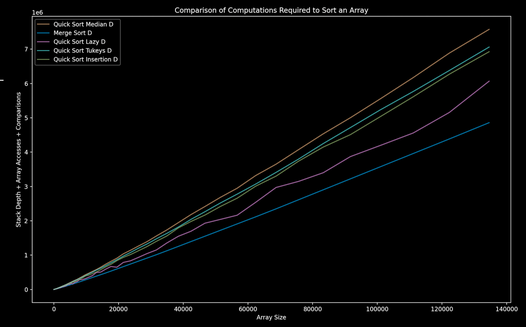


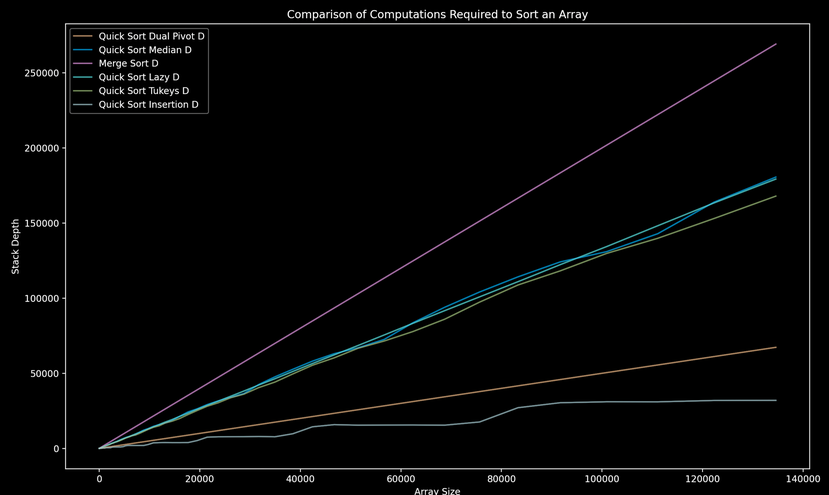
Again, when sorting data that is in descending order, dual pivot quicksort degrades to O(n^2) performance with regard to array accesses performed. The remaining algorithms perform at O(NlogN) performance, this time with Lazy pviot performing the best, implemented with a randomized pivot point choice.





When combining time and spacial complexity, Merge and Lazy come out on top again with regard to sorting data in descending order, and dual sort continuing to degrade to O(n^2) performance.





As with the random data arrays, spacial complexity is is O(NlogN) for all algorithms. Quicksort Insertion outperforms all, which makes sense as this algorithm attempts to cut down on recursive calls when array sizes are very small (< 15 items), saving stack memory in the process. Merge sort performs the worst, as it will always recurse an array to its individual elements, creating array sizes of 1 for every item in an array.