A Short Report on Hybrid Sort Performance (using matplotlib)

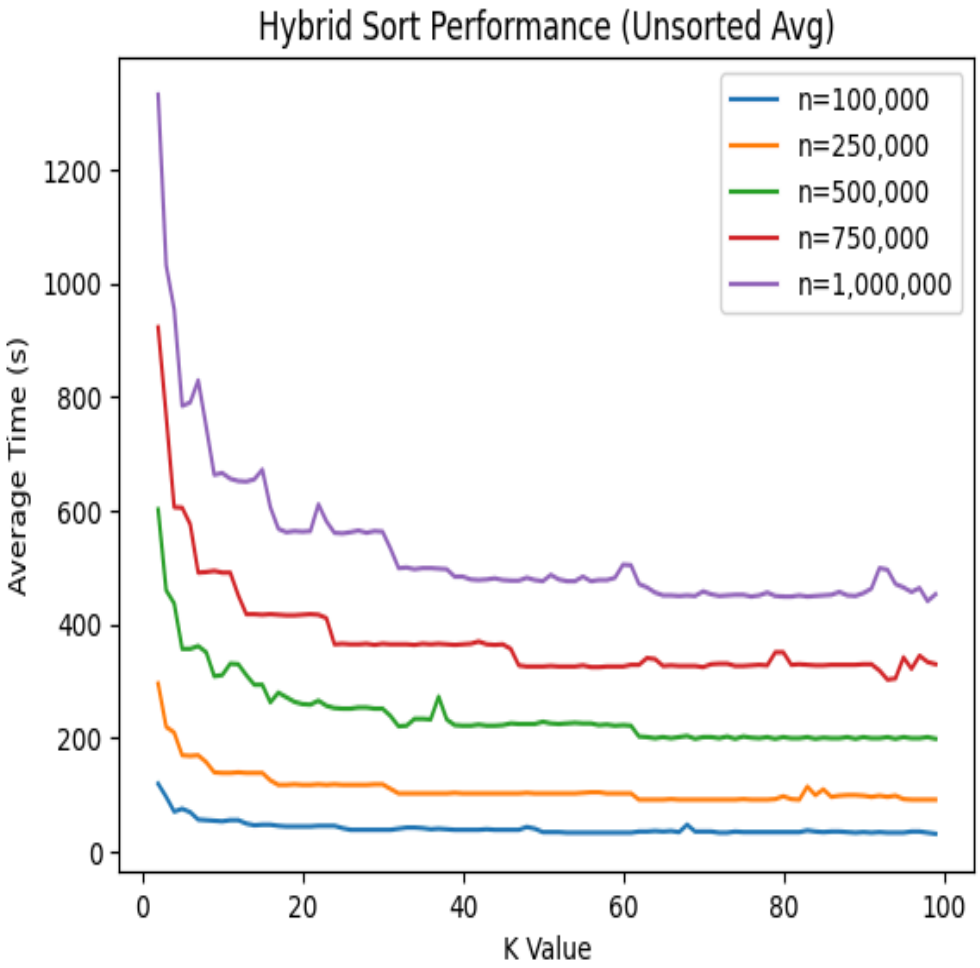
by Jacob W.

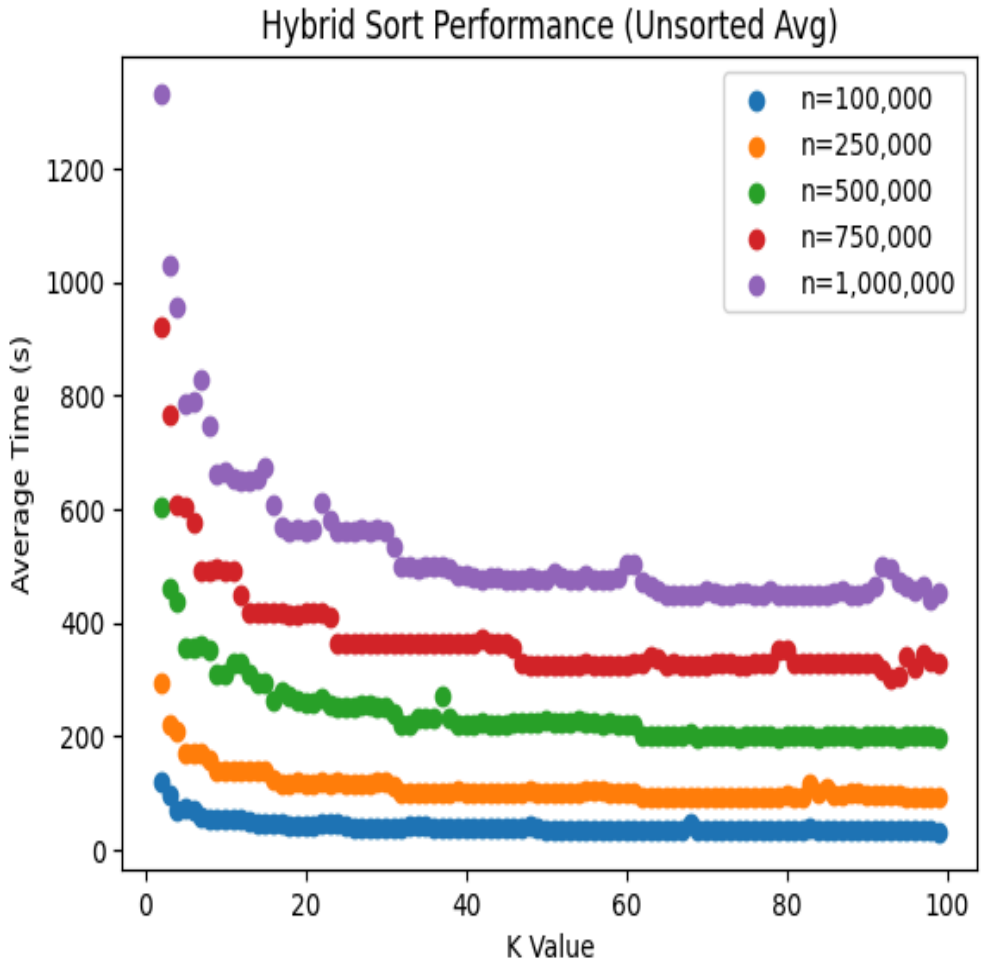
This report is split into 3 sections each answering a single project deliverable.

- Deliverable 1.2
- Deliverable 1.3
- Deliverable 1.4

Deliverable 1.2

A plot showing the average run time of your algorithm as a function of K, with a separate trace for at least 5 representative values of n. (20 points)





We met this expectation by finding the average runtime of the hybrid sort for k-values 2 through 99 for 5 different arrays of length n. The lengths of n we tested our hybrid sort algorithm on include 100,000, 250,000, 500,000, 750,000, and 1 million. Testing on these relatively large values of n give us a good idea about how the algorithm operates on large data sets.

Our program calculates the average by computing the runtime of each k-value 5 times on an array of length n and averaging out to find the average runtime. This did make testing take longer, but it gives accurate test results. See the plots below...

Each of the plotted lines follows a similar trajectory. At first when the value of k is small, the average runtime is at its highest. Following the line as the value of k is increased, the average runtime decreases (until a certain point).

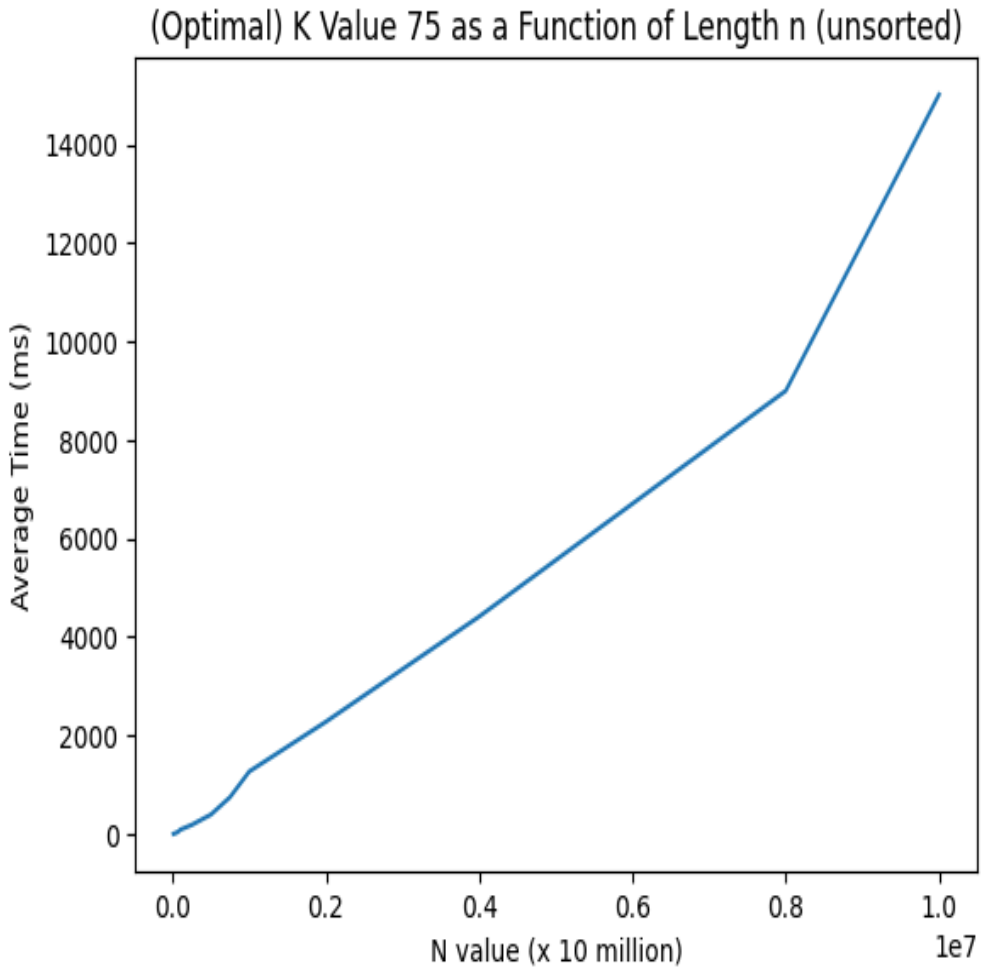
It is interesting to note that the larger the dataset is either the more outliers appear or the greater the variance between the outlier and the average line is. Notice on the yellow line (250k) there are less random peaks appearing than compared to the purple line (1M) and the

variance is greater. The same follows when comparing the blue line to the yellow line (100k vs 250k).

From the two plots, the optimal value of K seems to lie in the range of 60 to 80. I chose 75 as the optimal value because on each line plotted there are no outliers at that point, and the time has stabilized in a global valley.

Deliverable 1.3

A plot showing the optimal value of K as a function of array length n. Explain why you think the relationship between n and optimal K is the way that it is.(30 points)



Interestingly the line plot of the optimal value of K as a function of length n is linear from 1 million to 8 million length arrays. When testing the optimal value k on relatively small array sizes the results are not much different compared to if you were to use a different value of k. When the size of the array that you are testing on increases the time the algorithm takes to run also increases. The optimal value of k and this graph shows us that at the k value 75

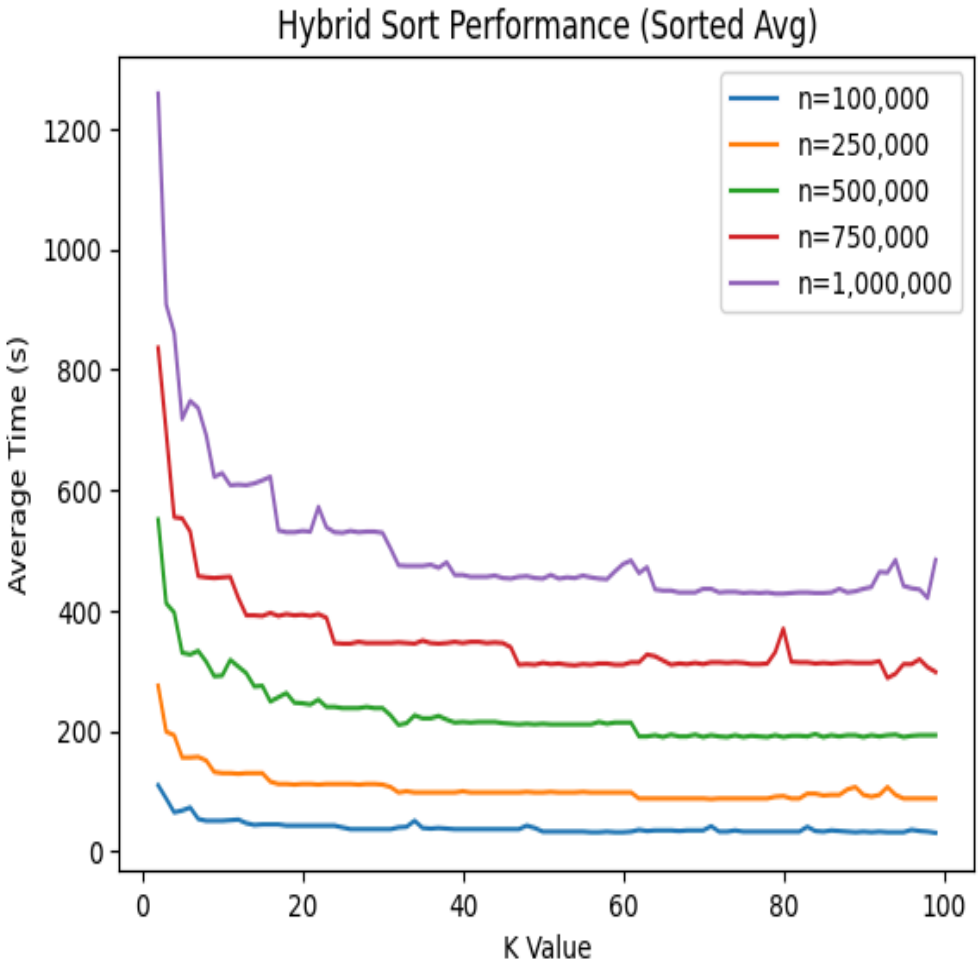
when you double the size of the arrays (1 million <= n <= 8 million) the average runtime doubles.

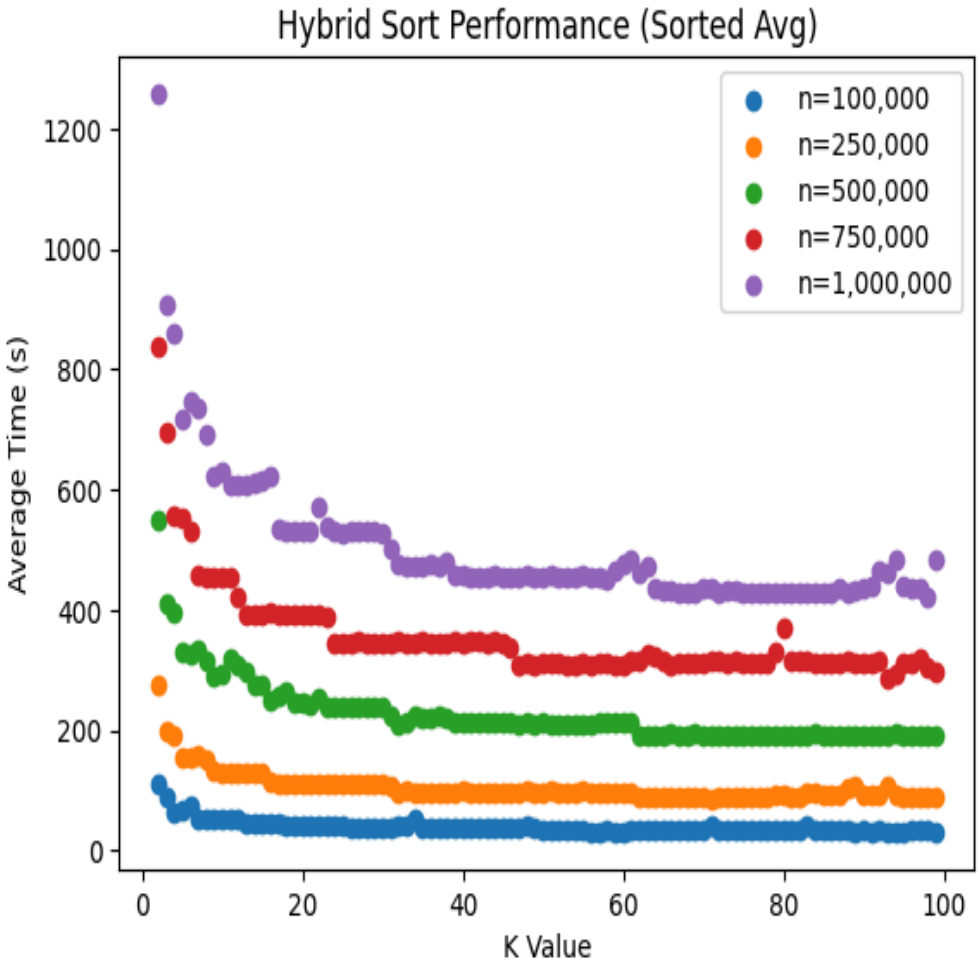
Deliverable 1.4

Deliverable 1.4: Deliverable 1.4: Your observations and findings from Task 4. (30 points)

Task 4: Repeat deliverables 2 and 3; however, this time, test your algorithm only on sorted arrays. How do the results differ from what you reported in Deliverables 2 and 3? Explain these differences to the best of your ability.

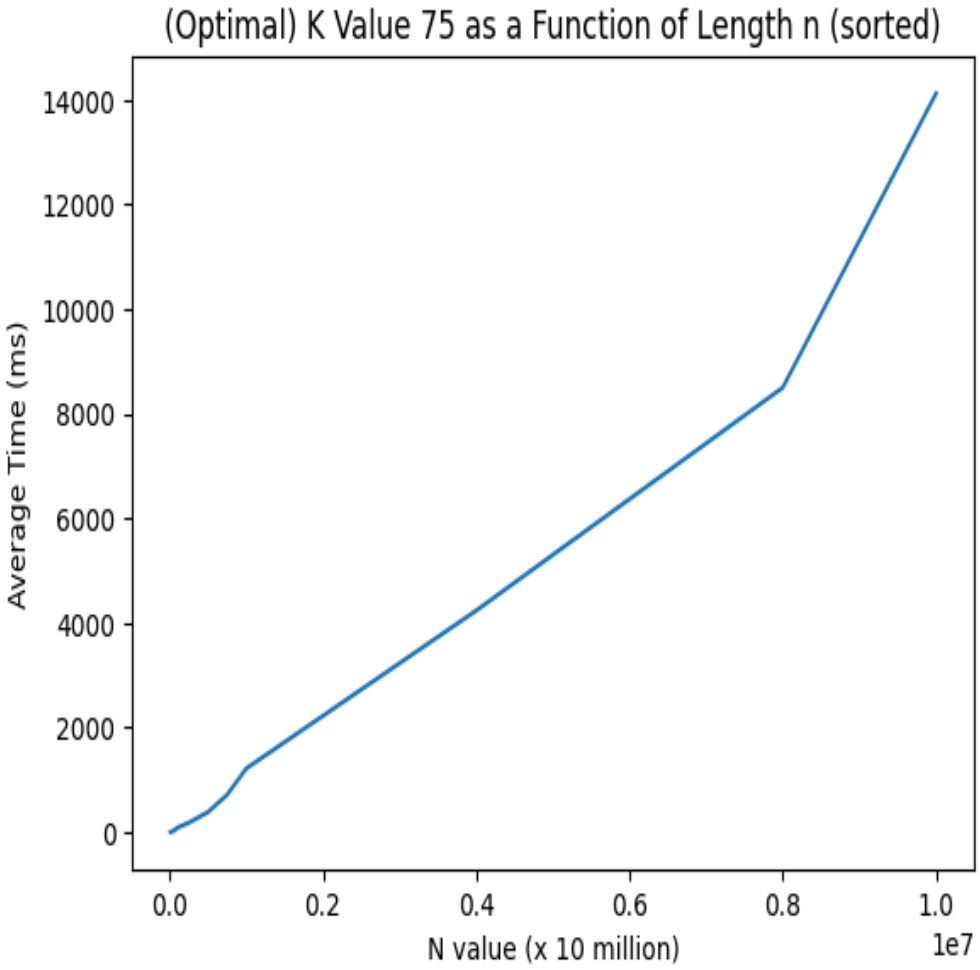
Deliverable 1.2 (sorted)





From the two plots, the optimal value of K seems to again lie in the range of 60 to 80. Again, a K value of 75 seems to be the optimal value, the graph shows that at the value 75 no outliers exist on any of the 5 graphs and the average runtime has stabilized in a global valley.

Deliverable 1.3 (sorted)



The graph of the sorted array tests is extremely similar to that of the graph of the unsorted tests.

The graphs are essentially identical. While it is true that it takes less time to use the hybrid sort algorithm to sort pre sorted arrays, but generally the time difference is less than 100 ms. It makes sense why this is true, less operations are having to occur over the algorithm's duration resulting in a slightly faster runtime, but since this is seen true for these cases, the difference still ends up being linear.