Task 2.3 – Explanation

Name: Er Tian Ru

Student ID: 30881668

Graphs

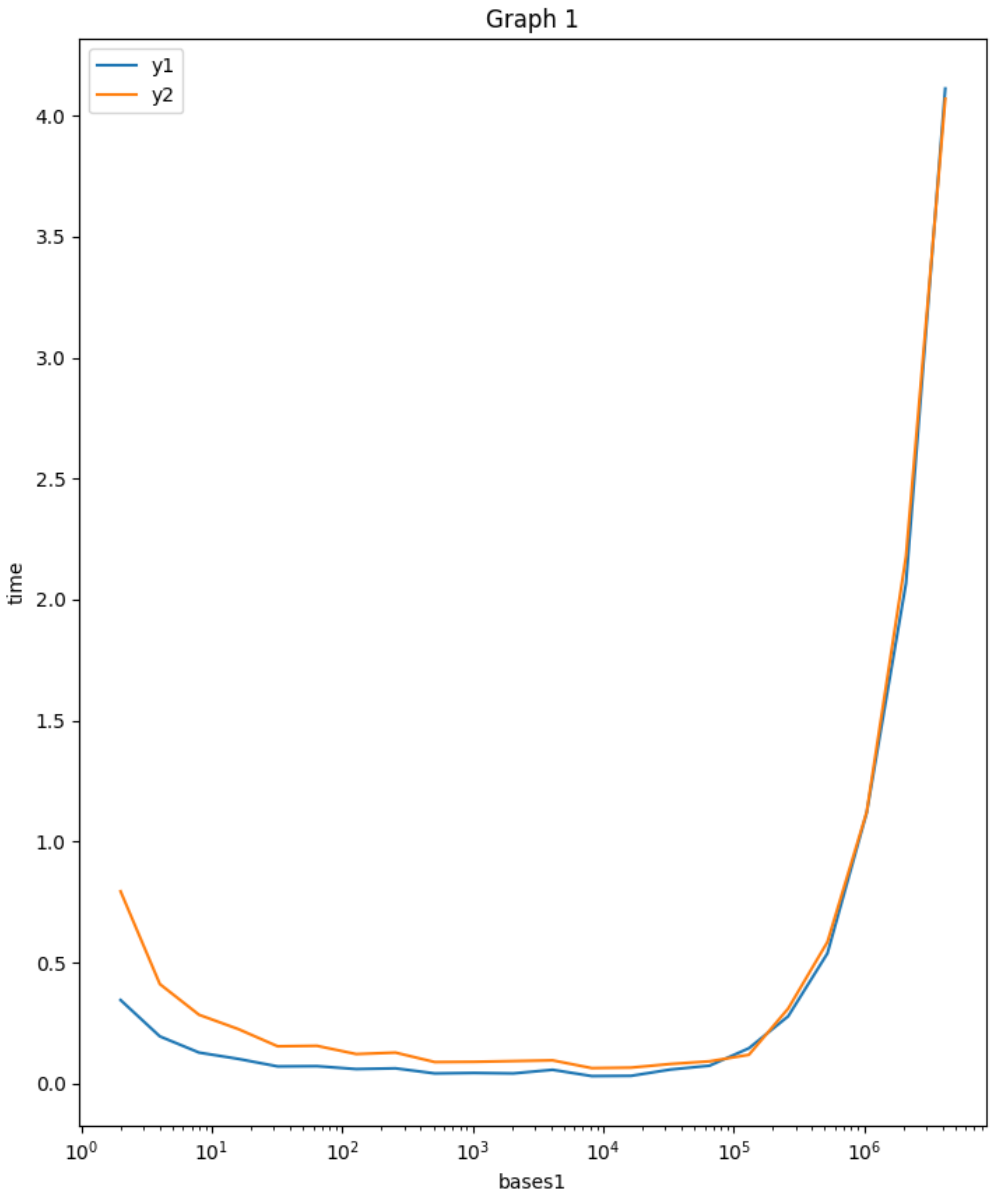

Figure 1 - comparing y1 and y2, a logarithmic x scale

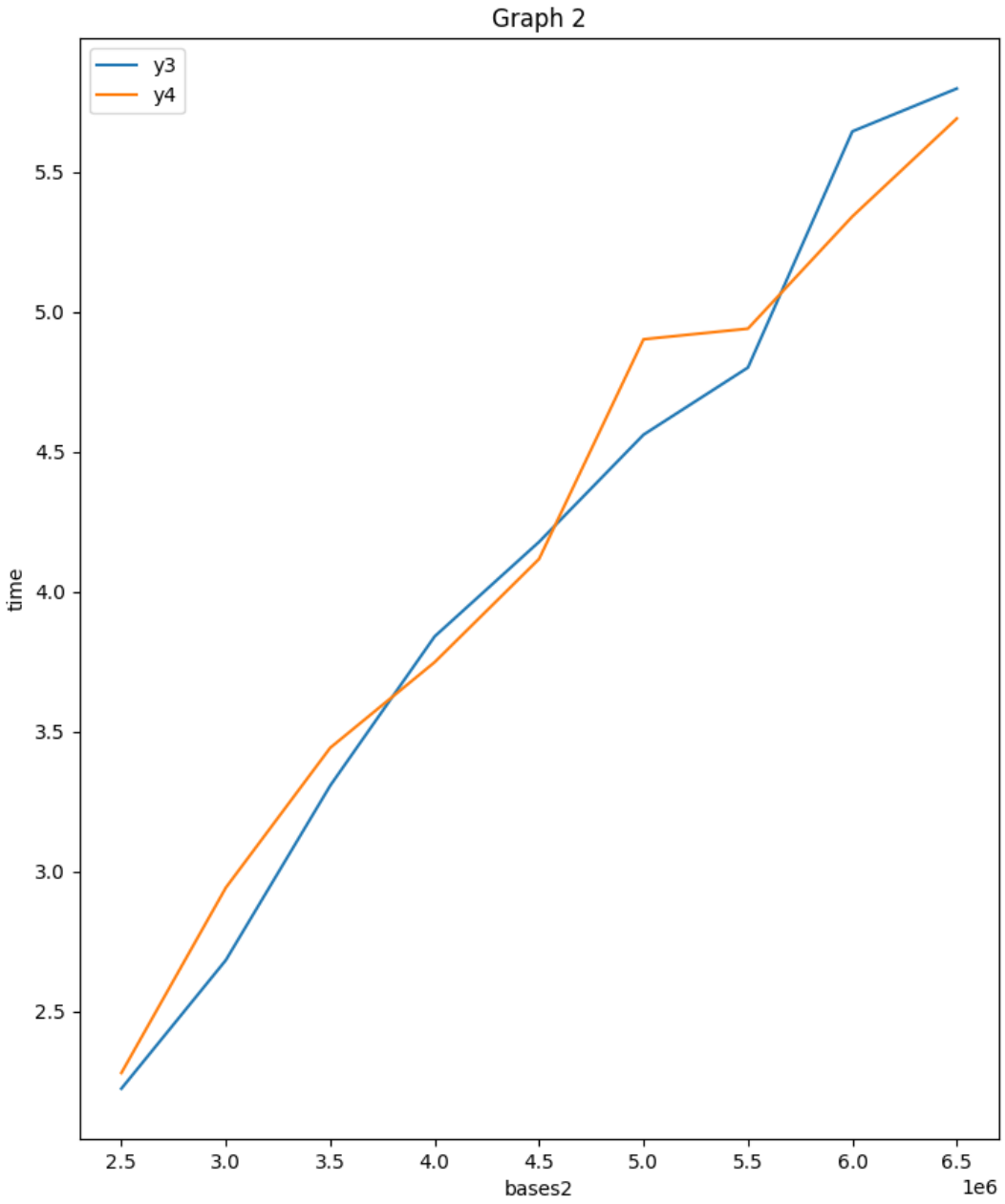

Figure 2 - comparing y3 and y4, linear x scale

Questions

1. **Why do the base/time curves for the first two graphs show a U shape? In other words, why are the times high when the base is low and when the base is high, but low when the base is in between? Justify your answer using the complexity of radix sort.**
   The runtimes are high when the base is low or high is because of the complexity of radix sort $O((n+b) \times log_b M)$. Having a lower base would mean having to sort through a high number of columns while having a high base would reduce the number of columns, it would increase the number of cells needed in count array. This can be seen in the complexity and in the graph when the base is high, it seems to have an exponential growth. The U curve tells us that there is a balance to be maintained with the number of columns and size of count array and a base that's in between would be the most optimal.

2. **Why are the times for y2 about twice as long for y1 when the base is low? Include a mathematical argument based on the complexity of radix sort in your answer.**
   The data list for y2 is almost double compared to the data list for y1, which means it has a larger n. When the base is low, we can see the base as negligible, in other words $O((n) \times log_b M)$. Since M does not make a lot of difference between y1 and y2 (the data for y1 and y2 are selected from the same range), we can view the time complexity as $O(n)$, so increasing the n for one dataset would make a noticeable difference in comparison.

3. **Why are the times for y2 not twice as long for y1 (and in fact are very close) when the base is high? Include a mathematical argument based on the complexity of radix sort in your answer.**
   When the base is high, it is not negligible and it will contribute a lot to the time complexity of $O((n+b) \times log_b M)$. Because of this an increase in n wouldn't affect the complexity by a large margin.

4. **Why are the times for y3 and y4 almost the same, despite data2 having twice as many elements as data1? Include a mathematical argument based on the complexity of radix sort in your answer.**
   The times for y3 and y4 are similar despite the data having twice as many elements is because doubling the size of n would not make a difference in terms of big O complexity. $O(2n)$ can be written as $O(n)$. This is because for large datasets, the constant of multiplying 2 would not make much of a difference in the runtime of the algorithm.

5. **Why do the graphs for y3 and y4 show an almost linear shape? Include a mathematical argument based on the complexity of radix sort in your answer.**
   The graph for y3 and y4 show an almost linear shape because the complexity of $O((n+b) \times log_b M)$ matches the linear formula of $y = mx + b$, where y is the runtime, x is the base and b and m are constants.