Part 2dv

Two important reasons we want to sort our data is readability for the user and algorithm efficiency. The first of these seems obvious enough, it is much easier for a person to read a sorted list, alphabetical or numeric, versus an unsorted list. In terms of algorithm efficiency, let's discuss the difference between a merge sort and a bubble sort algorithm, and then how a binary search works. Merge sort takes a divide-and-conquer approach to sorting elements. This method is O (n log n), a combination of linear and logarithmic complexity. Bubble sort, on the other hand, is O (n^2). This quadratic complexity is since a bubble sort runs through list for each element, checking to see if it is greater than or less than the next element. As we can see sorting can be an expensive operation, so we would want to do it upfront, consider our data preprocessing, before we need to use the sorted data to ensure our analysis is occurring as quickly as possible. A great example of this is the binary search, which is O (log n), or logarithmic complexity. Given a presorted list, a binary search can quickly cut a list in half, then half again, and then again to quickly find what it is looking for. If the data were not already presorted this operation would not be possible (Sam, 2014).