When picking a sorting algorithm, the 2 main priorities are run time and memory required. While there are many good options to choose from, LSD Radix sort is going to be the best bet for a spreadsheet sorting application. Radix sort is a non comparison based sort and has the fastest run time compared to other sorting methods because of this. Since it is non comparison based, its run time is determined instead by the number of items in the collection and the length of the items. Radix sort is also a stable sorting algorithm, meaning in regards to sorting duplicates, the entries will retain their relative position. While it does have the fastest run time of all sorts, one drawback is that it requires extra overhead memory to sort the data. While this could be a problem for extremely large data sets with extremely large values, it is practically insignificant. A sort that would not stand a chance against radix sort would be bubble sort. Bubble sort is a comparison based sort and has a quadratic run time. This means the more entries in the spreadsheet, the longer it will take to sort the collection. In comparison to radix sort, unless the length of the entries is equal to the number of entries, radix sort will always outperform bubble sort. The only minor advantage bubble sort has is its in place characteristic, meaning it requires no additional memory overhead when performing the sort. However the horrendous sorting time compared to radix sort makes it unusable. Another comparison based sort that is better than bubble sort is merge sort. Merge sorts average run time is faster than bubble sort, but not as fast as radix sort. In comparison to radix sort, as long as the size of the entries in the spreadsheet stay relatively small, radix sort will always outperform merge sort. Merge sort, just like radix sort, does require memory overhead when sorting so this is a drawback. By using radix sort you will be getting the fastest run times you can compared to any other sorting algorithm. You will also be able to keep duplicates in their relative position because of radix sorts stable nature. This is why radix sort is a great choice for this spreadsheet application.

Radix sort has a run time of O(n*w) where n is the number of entries and w is the length of the entries. This is compared to bubble sort which has a terrible run time of O(n^2). As stated before the only way bubble sort could be better is if w=n, in which case the length of the entries equals the number of entries which is very unlikely in a simple spreadsheet application. Merge sort has an average run time of O(n log n). By graphing these 3 functions you can clearly see how their run time is affected by the number of entries. Bubble sort's n^2 run time is so bad it's not even worth looking at. By changing the w in Radix sort's run time you can see how it compares to merge sort, and while for a limited number of entries it may be faster, as the number increases radix sort will ultimately become faster. If w is less than 6 as long as there are at least 100,000 entries, radix sort will always be faster, and the more entries it has, the faster it will be in relation to merge sort. In regards to memory overhead, bubble sort wins that aspect as it is an in place sort. Radix and Merge on the other hand require overhead, with merge sort having a space complexity of O(n) and radix having a space complexity of O(n+w). For a spreadsheet application these space complexities should not pose too much of a problem as the number of entries should be relatively low. By relatively I mean no more than a couple million, if that.