Spreadsheet applications are one of the most used pieces of technology when collecting, computing and displaying data. These applications sort through the data using different sorting algorithms to efficiently calculate data. When choosing a sorting algorithm for the spreadsheet application, I believe that LSD Radix sort is the most suited algorithm for a spreadsheet application. LSD Radix sort is the best suited sorting algorithm for a spreadsheet application because it is superior in terms of how fast the algorithm sorts the data and requires the least amount of comparisons when compared with bubble sort and selection sort.

Since spreadsheet applications use large amounts of unsorted data majority of the time, LSD radix sort is a better sorting algorithm because it can sort through the data faster than both bubble sort and selection sort. Both bubble sort and selection sort would work slower majority of the time because both sorting algorithms are better suited for smaller sets of data with only ten to twenty elements. In addition, bubble sort only works when the data is almost completely sorted or is already sorted which rarely occurs with spreadsheet data. To sort the data, bubble sort must iterate through the data many times, which is not time efficient. Similarly, selection sort would also take longer because it also iterates through the data many times to find the smallest value and swap it with the first cell whereas LSD radix sort sorts each digit in every cell from rightmost to leftmost until the data is sorted. LSD radix sort is faster to use especially when external memory is not an issue and is suited for spreadsheet applications.

Compared to bubble sort and selection sort which are both comparison sorts, LSD radix sort is a non-comparison sort which is inherently faster. This means that LSD radix sort does not compare the data in each cell to each other but sorts the data by grouping the individual digits that share the same position and value. This indicates that LSD radix sort does not need to compare the values making it more efficient to use for a spreadsheet application.

When looking at the runtime of all three sorting algorithms, LSD radix sort is better suited for a spreadsheet application than both selection sort and bubble sort. LSD radix sort has an average runtime of Θ(nk) where k represents the highest number of digits in a cell whereas both selection and bubble sort have an average runtime of Θ(n2). Especially for sufficiently large n, which occurs often with spreadsheet data, LSD radix sort is better suited because when n becomes large, k is presented as a constant. When faced with sufficiently large data (larger than ten to twenty elements), selection sort is greatly outperformed by Θ(n log n). Similarly, bubble sort also does not work well with larger sets of data because its efficiency decreases dramatically on lists greater than a small amount of elements. The only advantage that bubble sort has over LSD radix sort is where it sorts through an already sorted list where it has a runtime of O(n); however this seldom occurs.

LSD radix sort is more efficient because it does not need to compare the values to one another unlike both bubble sort and selection sort. Because both are comparison sorts, bubble sort and selection must iterate through the data many times. Bubble sort swaps adjacent values and “bubbles” the larger value until it reaches the N-1 item. The number of swaps can be calculated where the expected number of inversions are for a permutation with length n where P(i,j) is an inversion. Similarly, selection sort must scan n elements, where there are n-1 comparisons, and swap it with the value in the first position where the total number of comparisons are . In addition, because both are comparison sorts, they are inherently slower because they need to execute a comparator where each call gives the sort one bit of information. Because spreadsheet applications usually already have a requirement that the data in each cell must fall within a certain range of constant length, LSD radix sort is better suited for the application and does not need to worry about the lack of generality of the input which can be a problem for non-comparison sorts. LSD radix sort sorts by enqueuing the data into arrays of queues and remapping it to a different index location to be sorted. This also denotes that if space is not considered an issue LSD radix sort can outperform both selection sort and bubble sort when used in a spreadsheet application.

LSD radix sort is a better sorting algorithm than bubble and selection sort for a spreadsheet application because of how spreadsheet applications require data to have a certain range in each cell and knowing the typical size of the data used for spreadsheets. Because spreadsheets usually use larger sets of data, LSD radix sort is better suited especially when memory usage is not an issue and drastically outperforms bubble sort and selection sort. Because LSD radix sort is not a comparison sorting algorithm, it only needs to iterate through the data k times which is significantly less than the number of comparisons/swaps that selection sort and bubble sort use. In addition, because spreadsheet applications require each cell to be of certain length, this makes it easier for non-comparison sorts like LSD radix sort, since non-comparison sorts require all input data to be within a range of constant length to ensure linear time complexity. As a result, this allows the spreadsheet application to run more efficiently.