Switch to Radix Sort

When dealing with spreadsheets, keeping it alphabetized can be a mundane, repetitive task. Especially when dealing with large quantities of words in a single document. That's where Radix Sort comes into the picture. With Radix sort, words will be alphabetized easily and efficiently with each entry. Radix sort gets its name from the number of letters in the alphabet, which are referred to as English's radix. Radix sort is made specifically for words, and it works better than other types of sorting which tries to work the same way on letters and numbers.

Speed is the main benefit of Radix sort. Radix sort will put a new entry into its place faster than other methods. When dealing with data, one of the most noticeable aspects is speed. Radix sort can sort words noticeably faster than competing sorts. Nobody wants to wait while an algorithm runs in the background and stare at a spinning loading icon. With Radix sort, you will spend less time waiting, and more time doing. That frees up time to do other tasks, which leads to being more productive. Time is the most important resource. Finding ways to save time or better utilize time allows more tasks to be completed. If Radix sort can make one person better, it can certainly make others better too. Implementing Radix sort across all our spreadsheets will allow everyone to take advantage of this time saving update, and the company will be better for it.

Switching to Radix sort will make the company better. Saving workers' time will make them happier, which makes for less stress, which will make them more productive. Having more productive workers is always a goal of a company, and Radix sort can help achieve that goal.

Radix sort can sort strings better than competing sorts because it does not need to perform comparisons. This removes the lower bound of $O(N*\log_2(N))$, which is the minimum runtime to sort a data set of length N with comparison-based sorting. Radix sort can beat this lower bound; the runtime for Radix sort is O(N*w), where N is the number of items and w is the width of the longest item. This runtime is obtained from doing a traversal of the dataset for each digit in the longest item. When w is less than log(N), Radix sort will sort faster. The average word length in English is 4.7 characters, which can be practically treated as 5. $\text{Log}_2(N)$ will be greater than 5 when N is greater than 32. Typically, more than 32 words will be sorted if the spreadsheet is more than small. This makes the overall runtime Radix sort faster for larger quantities of data with small width.

Radix sort is ideal for data with small width. Unlike numbers, which can have over 10 digits commonly, strings have a lower average width. Radix sort is best with lower width because of the "w" in the runtime. The lower w, the lower the runtime. This is proven from the theory behind Radix sort. Each data element must be placed into a bucket each pass of the data. The number of passes of the data needed is equal to the width of the longest element. All the digits of the element must be placed into a bucket and removed before that element is considered sorted. If dealing with numbers, one number that is much larger than the others will cause many unnecessary placements of the numbers which are already sorted. With strings, it is much less common to have a word that is much larger in width than the other words. This allows Radix sort to sort strings more effectively than comparison sorts.

Many comparison sorts are designed to work on numerous types of data. Comparison sorts cannot get lower runtime than $O(N*\log_2(N))$. Merge sort, a commonly used sorting algorithm, has an average case runtime of this bound. A generalized sort, Merge sort can sort

different types of data, such as strings or integers, in $O(N*\log_2(N))$ time. It accomplishes this by splitting the data into halves, until only 1 element is left in each group. It then merges all the groups together and compares each element to another to determine where it should be placed in the new group. With integers, this will be better for large data sets because it gets rid of the unnecessary operations when one number is much bigger than the others. However, with strings, it is slower than Radix sort. Because strings are typically small in width, "w" will be less than $\log_2(N)$. This is also true for Quick sort, which is another comparison sort that rotates data elements based on a middle pivot value. For large integer sets, it is better than Radix sort, but for strings, Radix sort again is superior.

Making the switch to Radix sort will speed up spreadsheet usage and manipulation. Spending less time waiting for a simple sort to finish will cause cascading benefits throughout the company. Stop wasting time sorting your strings with methods designed for numbers and use the one designed for strings!