RADIX SORT LITURATURE SERVERY
INFO 6205:Program Structure and Algorithms

Abhishikth Daniel Merugu (NUID:001548340), Northeastern University, USA.
Ram Charan Teja Moodapally(NUID:002950328), Northeastern University, USA.

ABSTRACT
We present and evaluate implementation technique for string sorting. We will be using the sorting technique for sorting "Telugu Language" String which uses Unicode characters. Our experiment results indicate the comparisons of the sorting efficiency with different sorting algorithm like Tim Sort, Dual Pivot Quick Sort, Husky Sort, and LSD(Least Significant Digit) Radix sorting using the Benchmark Technique and suggest the optimization techniques and method used for sorting based on the input data.

ANALYSIS OF MSD RADIX SORT
Radix sorting is a simple and very efficient sorting method. As the sort can also be implemented in a stable way, Radix sort has a greater edge over other the other sorting technique. The nature of the algorithm is to scan the input string with the most significant digit. This algorithm is known as MSD radix sort. The algorithm functions as below.

a. Group the entire list of arrays into different group based on the first character, arranged in the ascending order.
b. Apply the same MSD algorithm recursively on each group separately, leaving the first character. Continue this until there are no further groups.

The main problem with the MSD radix sort is that it might process high empty brackets. Hence when there are too many of such groups with little data, it is advisable to switch to the comparisons-based sorting algorithm. This sorting includes Insertion sort, or bucket sort.

With a different approach and modification of the MSD sort, we can develop Adaptive Radix Sort and Forward Radix Sort.

a. **Adaptive Radix Sort:** This algorithm is the modified version of MSD radix sort. This is primarily for sorting real number. However, we can use this algorithm by converting every Unicode to the Integer. The algorithm is

extremely simple and has a good, expected running time. The algorithm sorts of n real numbers by disturbing them into n intervals of equal width. The process is continued until each interval that contain more than 1 element. Then using quick sort partitioning gets a sorting algorithm with O(n) expected time for uniform data and 0(nlogn) worst-cate time.

b. **Forward Radix Sort:** Forward radix sort overcomes the bad worst -case time complexity. This algorithm combines the advantages of LSD and MSD radix sort. The main strength of LSD radix sort is that it inspects a complete horizontal strip and inspects only the prominent characters.

   This algorithm maintains the invariant that after it passes , the strings are sorted according to the first I character. Here, we group the string based on the first character. We need to re-apply the algorithm recursively on each group. If the group contain only one string, then we do not sort process sorting for that group, as is self-sorted. For the I steps, the group will be sorted for the for their first I characters.

IMPLEMENTATION OF MSD RADIX SORT:

The major decision is whether to represent the input by an array or by a linked list. The array implementation has more advantage but on the compromise of the stability. On the other hand, linked list has advantage of being simple to implement. However, the primary goal is to compare different sorting algorithms, as the results vary based on the computer architecture.

DISCUSSIONS:

The problem we faced is the grammar part of the Telugu language. There are total 36 Consonants called as (Hallulu) and , 16 vowels called as Achchulu.

Situation: More often time, the alphabets in the Telugu language are phonetically combined with at max 3 characters. Which mean, a single character in Telugu language may have 1 Unicode to 4 Unicodes. Which results in multiple level of priority judging. Unfortunately, we don't any API, which will be grouping all the Unicode associated to a single Telugu character.

However, after a detailed analysis of over 100 various Telugu complex words, we found a pattern on the Unicode. When implemented, the proposed function has so far given the expected results which when we pass the Telugu words, we will get the grouping of the Unicode for that Telugu character. Giving priority to all such combination of Unicode is one way where we can implement the sorting of Telugu characters.

However, after running few test cases, we had found that, the proposed priority is being coinciding with the priority of the default system (UNICODE to integer conversion priority). Hence , with a small limitation, we can achieve the Telegu string sorting.

However, with the insight we got from the Unicode patters, we had modified the MSD sorting algorithm such as skip the comparisons of certain Unicode. This has increased the efficient of the algorithm.

FUTURE SCOPE:

For sorting words in Telugu, improvements can be done in such a way that, the first alphabet in the word which include both vowels and consonants can be converted into single code (Unicode or a priority identifier digit) and we can use this identifiers to perfectly sort all the Telugu words. Due this challenge, Telugu word sorting is not widely used in the real life. Hence, the proper API development can help resolving this issue for sorting names in Telugu.

REFERENCE:

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.54.4536&rep=rep1&type=pdf
Arne Andersson, Department of Computer Science, Lund University.
Stefan Nilsson, Department of Computer Science, Helsinki Univerisity of Technology
Algorithms (4th Edition) by Robert Sedgewick