Assignment 4 Problem 1

Suppose you want to sort a set of n strings over an alphabet of size d. Furthermore, suppose that the maximum-length string in your set is of length $\ell_{\max}$, and that the average length of the strings in your set is ℓ_{avg}. We assume that our sorted set will follow the standard alphabetical ordering; for example, `a` $<$ `ab` $<$ `b`.

a) One method we saw in class for accomplishing this task is *radix sort*. First, we pad all strings of length less than $\ell_{\max}$ with a suffix of end-of-word characters so that all strings have length $\ell_{\max}$, then we run radix-sort on our set of strings as usual. Using the parameters defined above, describe the runtime and the amount of space used by this approach.

b) Another method to accomplish this task makes use of uncompressed *multiway tries*. First, we build a multiway trie containing every string in our set. Each node contains an array of children pointers of size $d + 1$. Then, we perform a preorder traversal and read each string, giving us a sorted ordering. Using the parameters defined above, describe the runtime and the amount of space used by this approach. A big-O bound is acceptable.

Hint. Assume the trie has k nodes, and try to express the runtime and space using k along with the other parameters. Then try to bound k in terms of the other parameters.

c) For which situations would the radix sort approach be preferable? For which situations would the multiway trie approach be preferable?

solution:

a) For the radix sorts we learned in the lecture, we know that the LSD-Radix-Sort seems more efficient than the MSD-Radix-Sort, so we choose to use the method of LSD-Radix-Sort. From lecture, we know that time cost is $\Theta(m(n+R))$, and auxiliary space is $\Theta(n + R)$ where n is the size of array, which contains m-digit radix-R numbers. So similarly, extend this method to the string, we can define that the end-of-word character has the hightest priority (i.e. end-of-word is less than any other characters in the string)
Therefore, we have $n = n \quad m = l_{max} \quad R = d$
The runtime should be $\Theta(l_{max}(n + d))$ and the amount of space should be $\Theta(n + d)$

b) Assume the trie has k nodes. And the worst case is that all strings has an unique path (have no common nodes except the root) in the trie, and this gives us the upper bound of k: $k \in O(l_{avg} \cdot n)$
For the trie of k nodes, since we are tranversing the trie, and each node including leaves are visit exactly once. Therefore, the runtime for this method should be $\Theta(k) = P(l_{avg} \cdot n)$.
For the space analysis, since each node contains an array of children pointer of size

$d+1$, therefore, each node has are considered to have size $\Theta(d+1) = \Theta(d)$, therefore, for the whole trie with k nodes, the total space needed is $\Theta(k(d)) = O(l_{avg} \cdot d \cdot (n+d))$

c) Consider the runtime and space of two approach, it is not hard to find that when the difference between l_{max} and l_{avg} is large, then the multiway trie performs better in the runtime. Consider the space consumes, if d is too large or l_{avg} is too large, then the radix sort may be a better choice. Generally, in we need better runtime, then we choose trie, if we need few space, then we choose radix sort.