Problem Description

Given a sequence of positive integers $S = a_1, a_2, \cdots a_m$ with $max(S) \leq n^{10}$. Sort the sequence in linear time.

Approach

Since the maximum value of S is less than or equal to n^{10}, if written in base n all of the numbers an be written with 11 digits or less, with the maximum always being 10 000 000 000 regardless of what value n is. (if $max(S) < n^{10}$ only 10 digits would be needed)

Algorithm

Knowing that the list consists of 11-digit number in base n a radix of size n can be introduced to sort digit by digit, similar to LSD radix sort. This will use a modified version of the counting sort.

Algorithm 1 Counting Sort Modified

Input: $L, DIGIT$

1: $n \leftarrow L.length$
2: $div \leftarrow n^{DIGIT}$
3: $count \leftarrow \{0, \overset{n}{\ldots}, 0\}$ // fill an array of length n with zeroes
4: **for** $e \in L$ **do**
5: $\quad index \leftarrow \frac{e}{div} \bmod n$
6: $\quad count[index] \leftarrow count[index] + 1$
7: **end for**
8: **for** $i \in 1$ to n **do** // for i from 1 (inclusive) to n (exclusive)
9: $\quad count[i] \leftarrow count[i] + count[i-1]$
10: **end for**
11: $res \leftarrow [n]$ // Set res to array of length n
12: **for** $e \in L.reverse()$ **do**
13: $\quad index \leftarrow \frac{e}{div} \bmod n$
14: $\quad res[count[index]] \leftarrow e$
15: $\quad count[index] \leftarrow count[index] - 1$
16: **end for**
17: **return** res

In the algorithm above one key difference is introduced. A new argument $DIGIT$ is passed to the function, this value is used in create the variable div which is used in two places, on line 4 and line 9. Note that if $DIGIT = 0$ the function would act identically to normal counting sort for elements less than n, since $div = n^0 = 1$ and $(\frac{e}{1} \bmod n) = e$ for all $e < n$.

The key Difference from normal counting sort is that the counting is based on $\frac{e}{div} \bmod n$, this means that when $DIGIT = i$ the $i'th$ digit of the number will be isolated when the number is written in base n. For example in base 7 given the list $L = 34, 534, 2323, 5254, 2354, 53$ **counting sort modified(L, 2)** would return $L = 34, 53, 5254, 2323, 2354, 534$. In this case the list is sorted based on the 3rd LSD counting from 1. It is also important to note that this sorting is stable. Using the arguments from the E Question it is sufficient to say that this algorithm will run in linear time, and that the small changes introduced are all done in constant time. I will also not go into the correctness proof as that is also covered in the E Question.

Algorithm 2 Base n Sort

Input: L

$n \leftarrow L.length$
for $i \in 0$ to 11 **do** // for i from 0 (inclusive) to 11 (exclusive)
    $L = CountingSortModified(L, i)$
end for
return L

What this algorithm does is to sort the numbers in L one digit at the time starting with the LSD. This process is repeated 11 times, once for each of the 11 digits in a number to cover all corner cases. It is again important to note that the **CountingSortModified** is stable sorting algorithm.

Proof of Correctness

The principle of the sorting algorithm is very similar to radix sort. An invariant for the *for* is that at any point in the loop the list L is sorted for the digits in position less than or equal to $i+1$ when the number is written in base n. Since **CountingSortModified** is a stable sorting algorithm if it has once been sorted for the digit i and then for $i+1$ it will also remain sorted for all digits less than or equal to $i+1$. Since the *for* loop is run for i up to and including 10, every number that can be written with 11 digit in base n will be sorted. Since the largest number in L can be n^{10} and this number will have 11 digits (written like 10000000000 in base n) each element in the list will be sorted when the loop is completed.

Time Complexity

We have already established that the **CountingSortModified** is linear in n. This function is called 11 times. Thus the time complexity is $\mathcal{O}(11n) = \mathcal{O}(n)$ and thus this sorting is linear.