CS301 Assignment 2 Asnwers

Uğur Öztunç 28176

Problem 1 (Order statistics)

a) For better worst-case time, merge sort algorithm can be used among the comparison-based sorting algorithms. Using merge sort is going to take O(nlogn) time. After numbers are sorted, the first k elements of the list must be iterated. Time of this operation will be O(k). In total, sorting all numbers and taking first k of them is going to be O(k + nlogn) in worst-case.

b) In order to find k th smallest element, using order-statistic algorithm with naive approach will take O(n) time. After completing selection, using quick sort partition is going to take O(n) time. Finally, merge sort can be used again in this part with O(klogk) time complexity in worst-case. In total, the time complexity of this method will be O(n + klogk)

- Since $k \leq n$, $O(n + k\log k) \leq O(k + n\log n)$ always be satisfied; thus, using the algorithm in part b will be more effective.

Problem 2 (Linear-time sorting)

a) Radix sort algorithm can be used to sort numbers only. However, it is possible to modify the algorithm so that strings can be considered as numbers which are not in decimal, but in another numeral system. In our case, lets say we are going to sort strings containing characters from the English alphabet only. Since there are 26 letters in English alphabet we can consider every word as a number and every letter as a digit from base-27 (26 + 1 = 27, there will be extra 1 character which will be explained below) numeral system. In this way, we can represent and compare strings as numbers, and ultimately sort them. As for modifying the algorithm, we do not need to do anything so far, since strings can be compared with respect to the ASCII Table. However, there is a small problem while sorting strings(base-27 numbers) based on to their digits, as in the radix order: While comparing decimal numbers, if a number has more digits, it is definitely greater than the number with fewer digits, but the same is not true for strings. Example is shown below:

Integer comparison

535 < 35

String comparison

"aba" < "ba"

Corresponding Numbers = (121) > (21)

As seen above, when we try to sort the strings by their digits, as in numbers, an error occurs if there is a difference in the number of characters(digits) between two strings. In order to solve this problem we need to find the string that has the highest length and we must add extra character, which will have the smallest value in the system, at the end of all strings so that their length will be the same as the longest string. That's why 1 extra character was added to the numeral system above. With this way the comparison will be like following, which the extra character is '#' in this case:

"a" < "aba" < "ba"

"a##" < "aba" < "ba#"

Corresponding Numbers = (100) < (121) < (210)

In summary, at first, length of the longest string must be found, and then a character which have smaller value in ASCII table (may be '#') must be added (longestLength - currentStringLength) times at the end of all strings. Finally, regular radix sort operations must be carried out.

b) Input list = [“BATURAY”, “GORKEM”, “GIRAY”, “TAHIR”, “BARIS”]

1) Find the length of the longest string:

   * The longest string is ”BATURAY” with length of 7

2) Fill all strings with ‘#’ until they reach length 7:

   * [“BATURAY”, “GORKEM#”, “GIRAY##”, “TAHIR##”, “BARIS##”]

3) Apply radix sort:

   1. [“BATURAY”, “GORKEM#”, “GIRAY##”, “TAHIR##”, “BARIS##”]
   2. [“GORKEM#”, “GIRAY##”, “TAHIR##”, “BARIS##”, “BATURAY”]
   3. [“GIRAY##”, “TAHIR##”, “BARIS##”, “BATURAY”, “GORKEM#”]
   4. [“GORKEM#”, “TAHIR##”, “BATURAY”, “BARIS##”, “GIRAY##”]
   5. [“GIRAY##”, “TAHIR##”, “BARIS##”, “GORKEM#”, “BATURAY”]
   6. [“TAHIR##”, “GIRAY##”, “BARIS##”, “GORKEM#”, “BATURAY”]
   7. [“TAHIR##”, “BARIS##”, “BATURAY”, “GIRAY##”, “GORKEM#”]
   8. [“BARIS##”, “BATURAY”, “GIRAY##”, “GORKEM#”, “TAHIR##”]

4) Remove ‘#’ characters:

   * [“BARIS”, “BATURAY”, “GIRAY”, “GORKEM”, “TAHIR”]

c)
Finding longest string: O(n)
Filling all strings with ‘#’: O(n)
Radix Sort: O(nk)
Removing ‘#’s: O(n)
Asmptotic worst-case running time: O(3n+kn) = O(n) where k is the length of longest string and n is the size of the list.