PSO #4 Solutions Sketch (Week 5)

Week of 2021-09-20

1 Divide and Conquer Questions

1. Given k sorted linked lists, with n total nodes, merge them into one.

   Solution: Pair up the k lists, and merge each pair in linear time with $O(1)$ space. This leaves $k/2$ lists of average size $2n/k$. Repeat.

   Final time complexity is $O(n \cdot \log k)$ where n is the total number of nodes.

2. Given a set of n strings of length m, find the longest common prefix

   Solution: Divide the set into two halves, compare the left half's prefix with the right half's prefix character by character until they no longer match, and return. Return the whole word for a set of size 1.

   In the worst case we must iterate through all $n \cdot m$ characters, so our total complexity is $O(n \cdot m)$

3. A sorted array A with n elements is known to have a few distinct elements that each have many repetitions. Design an efficient algorithm to deduce the number of distinct elements as well as the number of times each such element appears and use m to define the number of distinct elements.

   Solution: We begin by checking the first index of the array A for the minimum element. We then use binary search to find the last index of the minimum element. We add one to the number of unique elements and then repeat the process by cutting out the indices that are composed of the minimum element and starting again from the new subarray without the previous minimum element. In this manner, we find the total number of unique elements, call it m, and then use binary search on A for each of them, resulting in a $O(m \log n)$ runtime.

4. Show the linear time selection algorithm does not yield a linear time algorithm for selecting the k^{th} smallest element if the elements in the array are partitioned into groups of 3.

   Solution: If we partition A into groups of 3 and find the median of each group and recursively proceed, then once we find the median of these medians, we know that there are 2 elements in half of the $\lceil \frac{n}{3} \rceil$ groups we partitioned that are guaranteed to be on one side of the median of the medians except the group that contains the median of medians and the last group of 3, which may contain less than 3 elements. Thus, the number of elements guaranteed to be on one side is $2 * (\lceil \frac{n}{3} \rceil * \frac{1}{2} - 2)$, which means the most we recurse on is $n - 2 * (\lceil \frac{n}{3} \rceil * \frac{1}{2} - 2) = \frac{2n}{3} + 4$.

However, we can then realize that $T(n) = \Theta(n) + T(\lceil \frac{n}{3} \rceil) + T(\frac{2n}{3} + 4)$ is not $\leq c * n$, as $n + c\frac{n}{3} + c + \frac{2}{3}cn + 4c = (c+1)n + 5c$, which is not $\leq cn$, so the property $T(n) \leq cn$ is not satisfied for when partitioning into groups of 3 for the median of medians.

5. Find the median of two sorted arrays A and B of length n and m (without merging them)

   Solution: Find the median of A and B. Let A be the smaller array, B be the larger array. Compare the two medians.

   If the median of A is less than the median of B, then the first half of A must lie strictly in the first half of the combined array. Note that there exists an element in the first half of B and the second half of A array that is the median. Thus, we can restrict our search to the first half of B and the second half of A.

   Similarly, if the median of A is greater than the median of B, we can restrict the search space to the first half of A and the second half of B.

   Lots of base/corner cases to consider here, we will not state them.

   Complexity is $O(\log(m+n))$