COMP 340 **Analysis of Algorithms**

Problem Set 4

Assigned: March 04, 2020 *Due: March 27, 2020*

Always provide explanations and show as much work as possible. Solutions to TADM's odd-numbered exercises are available at `http://www.algorist.com/algowiki/index.php/The_Algorithms_Design_Manual_(Second_Edition)`. Designing algorithms often involves some creativity, so start early and work consistently. If you are stuck on a problem, move on and come back to it. If you get stuck again, discuss it with your classmates and/or come see me in office hours.

Sorting/Divide and Conquer

1. Give an $O(n)$ algorithm to compute the mode of an unsorted array of n numbers.
2. TADM 4-18.
3. TADM 4-31.

Parallel Algorithms

4. We analyzed sequential mergesort in class. Now let's parallelize it.
   (a) Recall that mergesort recursively sorts the two halves, then merges the results. Consider a parallel mergesort algorithm that sorts each half in parallel, then merges sequentially. Give recurrence relations for the work and span and solve them.
   (b) The sequential merge is slowing us down. Consider the following parallel merge algorithm. The input is two sorted sequences L and R of length m and n, respectively.

   > Find the middle element of L. Use binary search to find the appropriate partition element of R, call that index s. Now recursively merge $L[0..\text{mid}_L]$ with $R[0..s]$ and $L[\text{mid}_L..m]$ with $R[s..n]$. The combine step is to concatenate the two results, but we can just allocate one $m+n$ array to place the results in so that concatenation is basically free.

   For simplicity, you may assume $m = n$. (A more tedious analysis works even when $m \neq n$.) Give recurrence relations for such a merge algorithm (both work and span), and solve them.

   Start with the span — it's nothing you haven't seen before. The work is harder, since the Master theorem does not apply. We will work through a proof by induction in class.
   (c) Now plug in this parallel merge to our parallel mergesort algorithm. Again, develop recurrence relations for the work and span and solve them.

Average Case Analysis (in class)

5. What is the expected maximum value of throwing two dice?
6. Peer-to-peer systems on the Internet often grow by linking arriving participants into the existing structure. Here's a simple model of network growth for these systems. We begin

with a single "node" v_1. When a new node joins (one at a time), it chooses an existing node uniformly at random and links to this node.

Consider running this procedure until we have n nodes $v_1, v_2, \ldots, v_n$. Then we'll have a directed network in which every node other than v_1 has exactly one outgoing edge, but perhaps many incoming links (or perhaps none at all). If some node v_j has many incoming links, it may have to deal with a large load. For example, it may need to handle lots of users uploading the hottest new movie. We'd prefer all nodes to have a roughly equal number of incoming links. Let's quantify the imbalance.

(a) What is the expected number of incoming links to node v_j in the resulting network? Give an exact formula in terms of n and j, and also try to express this quantity asymptotically (via an expression without large summations) using Big-O notation.

(b) Given the above process, we expect that some nodes will end up with no incoming links at all. Give a formula for the expected number of nodes with no incoming links.

7. We know that binary search trees do not perform well in the worst case unless we balance them. What about in the average case? Consider inserting n items into a BST, all drawn independently and uniformly at random from some suitable range. Give a recurrence relation $C(n)$ for the average (expected) number of recursive calls required (in the standard BST insert algorithm) to insert n elements. You do not need to solve this recurrence.

   Hint: All inserts pay the initial call that compares to the root. The root is the jth smallest element with probability $1/n$, which determines how many elements will go left and how many will go right.