COMP 340 **Analysis of Algorithms**

Problem Set 4

Assigned: February 28, 2018 *Due: March 23, 2018*

Always provide explanations and show as much work as possible. 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.

1. What is the expected maximum value of throwing two dice?

2. 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.

3. 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.

4. Give an $O(n)$ algorithm to compute the mode of an unsorted array of n numbers.

5. TADM 4-17.

6. TADM 4-18.

7. TADM 4-31.