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.

   $$\mathbf{E}[L_j] = \sum_{i=j+1}^{n} \mathbf{Pr}[X_{ij}] = \sum_{i=j+1}^{n} \frac{1}{i-1} = H_{n-1} - H_{j-1} = O\left(\log \frac{n}{j}\right)$$

   (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.

   $$\begin{aligned}\mathbf{Pr}[i \text{ has no incoming edges}] &= \prod_{k=i+1}^{n} 1 - \frac{1}{k-1} \\ &= \prod \frac{k-2}{k-1} \\ &= \frac{i-1}{i} \frac{i}{i+1} \frac{i+1}{i+2} \cdots \frac{n-2}{n-1} \\ &= \frac{i-1}{n-1}\end{aligned}$$

Thus

$$\begin{aligned}\mathbf{E}[N] = \sum_{i=1}^{n} \mathbf{Pr}[i \text{ has no incoming edges}] \\ &= \sum_{i=1}^{n} \frac{i-1}{n-1} \\ &= \frac{1}{n-1} \sum_{i=1}^{n} i-1 \\ &= \frac{1}{n-1} \frac{(n-1)n}{2} \\ &= \frac{n}{2}\end{aligned}$$

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.

   Since all elements touch the root which contains the jth smallest element out of n with probability $\frac{1}{n}$. In that case (jth smallest), we pay the cost of inserting the $j-1$ smaller elements into the left subtree $C(j-1)$ and the $n-j$ larger elements into the right subtree $C(n-j)$. The result is

$$n + \frac{1}{n} \sum_{j=1}^{n} (C(j-1) + C(n-))$$

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

   Use a hash table. Map each number to a count of how many times you've seen it, incrementing as necessary and keeping track of the maximum.

5. ~~TADM 4-17~~.

6. TADM 4-18.

7. TADM 4-31.