CSCI 6220/4030: Homework 4

Assigned Monday October 22 2018. Due at beginning of class Monday November 5 2018.

Remember to typeset your submission, and label it with your name. Please start early so you have ample time to see me during office hours. Provide mathematically convincing arguments for the following problems. Ask me if you are unclear whether your arguments are acceptable.

1. Arlie did well in a randomized algorithms course and goes on to land a well-paying job at company A. The first solo project Arlie is assigned is to design a load balancing scheme for company A's new private cloud computing platform. To prevent information leakage, the load balancing algorithm must not use knowledge of the execution times of each job, and because execution times could be inferred from the length of the job queues, it also must not use knowledge of the size of the job queues.

   (a) Arlie's first thought is to randomly assign jobs to servers as they come in. The CIO is convinced when Arlie argues that with high probability over the assignment of the jobs, if m jobs are assigned to n servers, then the maximum loading is $O(\frac{m \log(n)}{n})$.

   Prove Arlie correct: find constants C and $c > 1$ such that the probability that the most loaded server has more than $C\frac{m}{n}\log_2 n$ jobs is smaller than $n^{1-c\frac{m}{n}}$.

   (b) Unfortunately, the team assigned to Arlie's project incorrectly implements Arlie's algorithm. Their implementation actually assigns the ith job to a uniformly randomly chosen one of the first i servers. Surprisingly, this error is not caught in test runs. The CIO is very concerned when this error comes to light after the product's release, but Arlie argues that randomized algorithms tend to be robust.

   Prove Arlie correct: show that when this slightly different algorithm is used to assign n jobs to n servers, the maximum loading is $O(\log n)$. Specifically, find constants C and $c > 1$ such that the most loaded server has less than $C\log_2 n$ jobs with probability at least n^{-c}.

2. Consider a random treap T with n vertices. As in the lectures, order the vertices in decreasing order of their search keys, so $x_1.k > \cdots > x_n.k$, and the priorities of the vertices are i.i.d Uniform$[0, 1]$ random variables.

   (a) What is the expected number of leaves in T?

   (b) What is the expected number of nodes in T with two children?

   (c) What is the expected number of nodes in T with exactly one child?

   (d) Prove that the expected number of proper descendants of any node in T is exactly equal to the expected depth of that node.

   (e) What is the probability that x_j is a common ancestor of x_i and x_k?

   (f) What is the expected length of the unique path from x_i to x_k in T?

3. Suppose we are given two skip lists, one storing a set A of m keys, the other storing a set B of n keys. Describe and analyze an algorithm to merge these into a single skip list storing the set $A \cup B$ in $O(n + m)$ expected time. We do not assume that every key in A is smaller than every key in B; the two sets may be arbitrarily intermixed.

4. Suppose that we are using an open-addressed hash table of size m to store n items, where $n \leq m/2$. Assume an ideal random hash function. For any i, let X_i denote the number of probes required for the ith insertion into the table, and let $X = \max_i X_i$ denote the length of the longest probe sequence.

   (a) Prove that $\mathbb{P}[X_i > k] \leq \frac{1}{2^k}$ for all i and k.

   (b) Prove that $\mathbb{P}[X_i > 2\ln n] \leq \frac{1}{n^2}$ for all i.

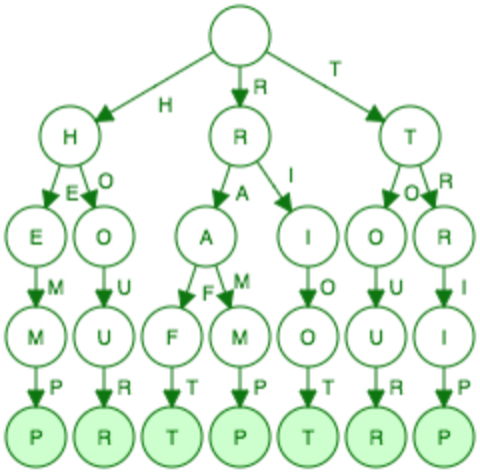

Figure 1: A radix tree containing the words "HEMP", "HOUR", "RAFT", "RAMP", "RIOT", "TOUR", and "TRIP".

(c) Prove that $\mathbb{P}[X > 2\ln n] \leq \frac{1}{n}$.

(d) Prove that $\mathbb{E}[X] = O(\ln n)$.

5. A radix tree over an alphabet of size m is a tree data structure where each node has up to m children, each corresponding to one of the letters in some alphabet. A word is represented by a node at the end of a path whose edges are labeled with the letters in the word in order. The only nodes created in the radix tree are those corresponding to stored keys or ancestors of stored keys. Radix trees are particularly useful for string matching and string completion problems. See Figure 1 for an example.

   Suppose you have a radix tree into which you have already inserted n strings of length k from an alphabet of size m, generated uniformly at random with replacement. What is the expected number of new nodes you need to create to insert a new string of length k?

6. [**required only for CSCI6220**] Consider the following variant of multiplicative hashing, which uses slightly longer salt parameters. For any integers $a, b \in [2^{w+\ell}]$ where a is odd, let

$$h_{a,b}(x) = \left\lfloor \frac{(ax+b) \bmod 2^{w+\ell}}{2^w} \right\rfloor,$$

   and let $\mathcal{MB}^+ = \left\{h_{a,b} \mid a, b \in [2^{w+\ell}] \text{ and } a \text{ is odd}\right\}$. Prove that the family of hash functions $\mathcal{MB}^+$ is strongly near-universal (aka near-uniform):

$$\mathbb{P}\left[\{h(x) = i\} \cap \{h(y) = j\}\right] \leq \frac{2}{m^2}$$

   for all items $x \neq y$ and all (possibly equal) hash values i and j.