CSCI 6220/4030: Homework 4

Assigned Monday October 28 2018. Due at beginning of class Thursday November 7 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 $1 - n^{-c}$.

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 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.

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

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

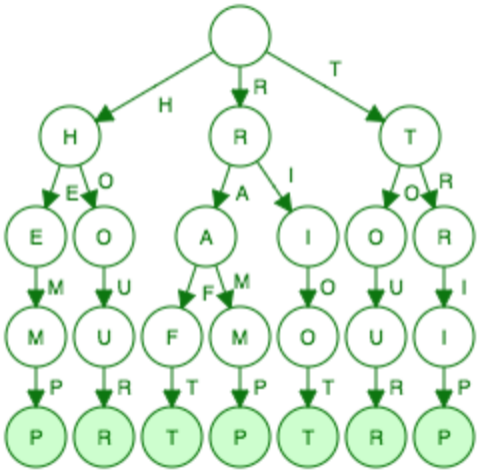

Figure 1: A radix tree containing the words "HEMP", "HOUR", "RAFT", "RAMP", "RIOT", "TOUR", and "TRIP".

4. 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?

5. [**required only for CSCI6220**] Remember that one of the uses of randomization is to reduce the storage space of algorithms.

   As an illustration of that fact, we will design an algorithm that, given a streaming view of n arbitary items in some universe U, returns an accurate estimate of the number of unique items d in that stream, using a near optimal amount of storage space.

   For simplicity, we assume that $U = \{0, \ldots, n-1\}$ and that n is known, is greater than 2, and is a power of 2. We will also assume that we have access to an ideal hash function from U to U[1].

   (a) Argue that any algorithm that computes d exactly must use at least $\log_2 n$ bits of storage.

   (b) Let zeros(x) denote the number of zeros that precede the first 1 in the binary expansion of x. For example,

$$\begin{aligned}\text{zeros}(1) &= \text{zeros}(001_2) = 0\\ \text{zeros}(2) &= \text{zeros}(010_2) = 1\\ \text{zeros}(4) &= \text{zeros}(100_2) = 2\\ \text{zeros}(5) &= \text{zeros}(101_2) = 0.\end{aligned}$$

[1] This is unrealistic, but makes the analysis simple. One can show that a 2-universal hash function suffices, with more effort; further, there exist such hash functions that can be stored using $O(\log_2 n)$ bits. The algorithm in this problem becomes both practical and space-optimal if such a hashing function is selected.

Let d random numbers be sampled i.i.d. uniformly at random from U. Show that, for any $z \in \{0, \ldots, \log_2(n)\}$, the expected amount of these numbers that satisfy $\{\text{zeros}(x) \geq z\}$ is $\frac{d}{2^z}$.

(c) The above observation tells us that *if* the d unique items making up our stream were selected i.i.d uniformly at random from U, then we could estimate d as 2^z, where z is the largest $\text{zeros}(a_i)$ of the items a_i in our stream. This is not generally the case, but we can use an ideal hashing function h to randomize our stream.

Consider the following algorithm that reads in a stream of n items a_i and employs h to calculate an estimate $\hat{d}$ of d:

1: $z \leftarrow 0$
2: **for** $i \leftarrow 1 \ldots n$ **do**
3: $\quad z \leftarrow \max\{z, \text{zeros}(h(a_i))\}$
4: $\hat{d} \leftarrow 2^{z+1}$

Argue that $\hat{d} \leq 2d$ deterministically. To find a lower bound on $\hat{d}$, express the event $z \leq \log_2(d) - 1$ as the event that a sum of d independent r.v.s is less than 1, then show that this implies that $\hat{d} \geq d$ with probability at least $1 - \mathrm{e}^{-1}$.

(d) The above algorithm uses optimal, $O(\log_2 n)$, storage space and has constant failure probability. Explain how to use $O(\log_2^2 n)$ space to reduce the failure probability to $\frac{1}{n}$. That is, give an explicit algorithm in the above format, and analyze its failure probability.