University of Washington
Department of Computer Science and Engineering
CSEP 521, Winter 2021

January 31, 2021

Homework 4, Due Thursday, February 4, 2021

Problem 1 (10 points):

Show that the number $1048579 = 2^{20} + 3$ is not prime without factoring the number. Hint: If p is prime, then $a^{p-1} \bmod p = 1$ (Fermat's theorem) for all a, $1 \le a \le p-1$. Find an a where this is not true, and you show that a number is composite.

Problem 2 (10 points):

Open Addressing stores all elements in a hash table. If there is a collision, and value is re-hashed, until an empty cell is eventually found. In the uniform case, we assume that each of the hash values is independent. (See CLRS, Section 12.4 for more details. This is problem 12.1 from that book.)

A hash table of size m is used to store n items, with $n \le m/2$, so the load factor is at most $\frac{1}{2}$. Open addressing is used for collision resolution.

a) Assuming uniform hashing, show that for $i = 1, 2, \ldots, n$, the probability that the i-th insertion requires strictly more than k probes is at most 2^{-k}.

b) Show that for $i = 1, 2, \ldots, n$, the probability that the i-th insertion requires more than $2 \log n$ probes is at most $1/n^2$.

Let the random variable X_i denote the number of probes required by the i-th insertion. You have shown in part (b) that $Pr\{X_i > 2\log n\} \le 1/n^2$. Let the random variable $X = \max_{1 \le i \le n} X_i$ denote the maximum number of probes required by any of the n insertions.

c) Show that $Pr\{X > 2 \log n\} \le 1/n$.

d) Show that the expected length of the longest probe sequence is $E[X] = O(\log n)$.

Hint for part d: The expectation of X can be written as $E(X) = \sum_{k=1}^{n} kPr\{X = k\}$. Consider the terms with $k \le 2 \log n$ and $k > 2 \log n$ separately.

Problem 3 (10 points):

Suppose that you have an implementation of Bloom filters that uses a table of size n with k hash functions. You are asked to adapt your Bloom filter to a setting where the table is of size $n/2$. (For example, you may have to target a new hardware device with less available memory.) How do you adapt your existing Bloom filter. You must be able to utilize the data already stored in the Bloom filter and you do not have access to the un-hashed values, so you can't just re-hash the values with new hash functions. What do you think the impact would be on performance?

Programming Problem 4 (20 points):

Experimentally evaluate the"two choice" hashing scheme discussed in class. Two choice hashing uses two hash function and assigns items to the least used bucket. For this problem, you will assume a chaining implementation of hashing, since you are counting the number of elements assigned to buckets (but you don't need to implement chaining, since you only need the counts.)

For this problem, you should compare one choice hashing, with two choice hashing, and three choice hashing (defined analogously). You may assume that your hash functions are perfectly random. For each experiment you run, you should hash n items into n cells. The statistic of most interest is the maximum number of items assigned to a cell, but computing the standard deviation is also of interest.

Recommended values of n are $1,000$, $10,000$, $100,000$ and $1,000,000$. You may go higher if you wish, but it is not necessary to code up special versions for very large sizes. Run enough executions at each value of n to get a good estimate on the expected maximum number of items.

I am curious about the results for this one. The theory says that for one choice, you should get $O(\log n/\log\log n)$ and for two and three choice you should get $O(\log\log n)$ for the expected maximum number of items per cell. However, it is unclear what are the constants hidden in big-Ohs. I expect that there will be a difference between one and two choices for large n and I expect that the difference between two choice and three choice to be small. For changes in values of n, I expect a big difference for the one choice, but I doubt there will be much difference between values of n for the two choice algorithm (because $\log\log n$ doesn't change much.)

If you are interested in running an extra experiment, evaluate how k-choice hashing performs as k ranges from 1 to n. (Obviously, n-choice hashing will perform pretty well with respect to balancing the load.)