Your name
CS 261, Spring 2020
Homework 2
Due Monday, April 13

why load factor is this

1. Suppose that we use hash chaining for a hash table with n keys, but with a hash table that is too small, so that the load factor α is $3 \log n$. (This will cause operations on the hash table to take logarithmic time rather than constant time.) Use the Chernoff bound and linearity of expectation to prove that, for a random hash function and for this load factor, the expected number of hash chains that are at most twice their expected size is $o(1)$.

change to at least

■

2. In Python, there is a function hash() built into the language that can map most types of object to an integer. For an integer x, the value of hash(x) is just $x \bmod 2^{31} - 1$, so for all integers smaller than $2^{31} - 1$, the hash of x is just x itself. Other types of objects have less-predictable hash values. The hash value, computed in this way, is used as the hash function for dictionaries by taking the result modulo the dictionary size. Python uses open addressing, but not linear probing, for its dictionaries.

Suppose you are given a hash table of fixed size s, initially empty, and that (unlike Python) you are going to use linear probing for this table, using the Python hash() function. Describe a sequence of $s/2$ integers such that, when inserted in that sequence into the table, the average time per insertion is $\Omega(s)$.

■

3. Suppose that we insert n keys into an initially-empty cuckoo hash table, all of them successfully. Suppose also that, of the two hash functions $h1$ and $h2$ used for this insertion, $h1$ is bad and returns the same hash value for all n of the keys. In this scenario, how many pairs of keys can have equal values of $h2$? Use the case analysis from the lecture notes to prove that, in this scenario, all insertions take worst-case time $O(1)$.

■

4. The algebraic method for constructing a 2-independent hash function from the lecture notes chooses a large non-random prime number p and two random numbers a_0 and $a_1 \bmod p$, and defines the hash value for x to be $(a_1 x + a_0) \bmod p \bmod N$. Suppose we try to simplify this by skipping the "mod p" part. Instead, we choose two random numbers a_0 and $a_1 \bmod N$ defining the hash value to be $(a_1 x + a_0) \bmod N$. Find a value of N for which this is not 2-independent, and explain why not.

■