L3 PROBLEM 5 INTRODUCTION

Recall from the last problem that when creating hash tables, we try to optimize both the size of the table (as small as possible) and lookup time for elements (as short as possible). It turns out that by making the hash table large enough, we can reduce the number of collisions sufficiently to allow us to treat the complexity of lookup as almost $O(1)$. I.e. we can trade space for time. But what is the tradeoff?

First, let's get a formulation of the problem. Assume:

- The range of the hash function is `range(n)`
- The number of insertions is K
- The hash function produces a perfectly uniform distribution of the keys used in insertions. This means that for all keys, k, and integers, i, in `range(n)`, the probability that `hash(k) = i` is $1/n$.

So, what is the probability that at least one collision occurs?

The question is equivalent to asking "given K randomly generated integers in `range(n)`, what is the probability that at least two of them are equal?" As is often the case, it is easiest to start by answering the inverse question, "given K randomly generated integers in `range(n)`, what is the probability that none of them are equal?"

When we insert the first element, the probability of not having a collision is clearly 1 (since the table is empty!). How about the second insertion? Since there are $n-1$ hash results left that are not equal to the result of the first hash, $n-1$ out of n choices will not yield a collision. So, the probability of not getting a collision on the second insertion is $(n-1)/n$, and the probability of not getting a collision on either of the first two insertions is $1 \cdot (n-1)/n$

We can multiply these probabilities - for each insertion the value produced by the hash function is independent of anything that has preceded it. Thus the probability of not having a collision after three insertions is:

$$1 * \frac{(n-1)}{n} * \frac{(n-2)}{n}$$

And after K insertions,

$$1 * \frac{(n-1)}{n} * \frac{(n-2)}{n} * \ldots * \frac{(n-(K-1))}{n}$$

To get the probability of having at least one collision, we merely subtract this value from 1:

$$1 - \left[\frac{(n-1)}{n} * \frac{(n-2)}{n} * \ldots * \frac{(n-(K-1))}{n} \right]$$

Given the size of hash table and the number of expected insertion, we can use this formula to calculate the probability of at least one collision. If K is reasonably large, say 10,000, it would be a bit tedious to compute the probability with pencil and paper. That leaves two choices, mathematics and programming. Mathematicians have used some fairly advanced mathematics to find a way to approximate the value of this series. But unless K is very large, it is easier to run some code to compute the exact value of the series. In this problem we'll look at some simulations that examine hashing probabilities.