Part 1

Marlan McInnes-Taylor

4/13/18

In order to create a new hash function with a different sequence of jumps, it became clear that using the value passed into the function was key. The initial $h_1(x)$ is

$$val \mod m$$

where *val* is the value to be inserted and m is the size of the hash table. An additional component was added to make each probing sequence different. This component was

$$i * (\frac{val}{m} * 2 + 1)$$

where i is the i^{th} iteration of a given function for a given value. This creates a different probing sequence for different values, since the result of the division operation differs based upon the value of *val* passed into the function. Furthermore, multiplying the result of the division by 2 and adding 1 guarantees that the number is odd. This is an important constraint which, as the textbook explains, allows for the entire hash table to be searched, assuming the table size is a power of two.

This is an efficient method since computing the additional component requires only an additional division, multiplication, and addition operation. It is clear the method works, as a hash table of size 2^{20} was filled entirely, searched, and removed from, without issue in multiple tests.

Thus, the final function is:

$$((val \mod m) + i * (\frac{val}{m} * 2 + 1)) \mod m$$