Timothy Holland
Seed number: 73

Hash Experimentation Report

Introduction

In this report I will evaluate the performance of different systems of hash functions in applying a linear probing hash table. Linear probing is a form of open addressing hashing, where each bucket contains at most one item, and a probe sequence is constructed as a linear progression across the table. With this in mind, two specific experiments are performed. First, a *growth* experiment, where different hash table sizes N (as a power of 2) are created and for each size small keys are inserted in random order until 60% of the table is occupied, wherein a lookup operation is applied for keys 0 to N-1. Second, a *usage* experiment, where the size of the hash table remains constant at 2^{20}, wherein insertions are performed to increase the usage of the table by 1% up to an upper bound of 90%. Both experiments are repeated 40 times, and the average number of buckets probed per operation and standard deviation across instances are reported in two respective graphs per experiment. The systems of hash functions are as follows: multiply-shift which uses the top 32-bit word (ms-low), multiply-shift which uses the lower bits of the upper half of 64-bit word (ms-high), polynomial hash function of degree 1 (poly-1), polynomial hash function of degree 2 (poly-2), and tabulation hash. To evaluate their performance, I will consider each experiment independently, starting with the *growth* experiment and finishing with the usage experiment.

Growth Experiment

For the first test, *growth*, different hash table sizes N (as a power of 2) are created, small keys are inserted in random order until 60% of the table is occupied, wherein a lookup operation is performed on keys 0 to N-1. For each table size, the experiment is repeated 40 times, the average number of buckets probed per operation, and standard deviation across iterations are reported in separate graphs. For the sake of simplicity, I will consider the performance of each function independently, and attempt to support these results with claims established within the lectures. Moreover, I will concentrate comparitive evaluations mainly between families of functions (like polynomials or multiply-shift). The experiment results can be viewed in figures 1 and 2:

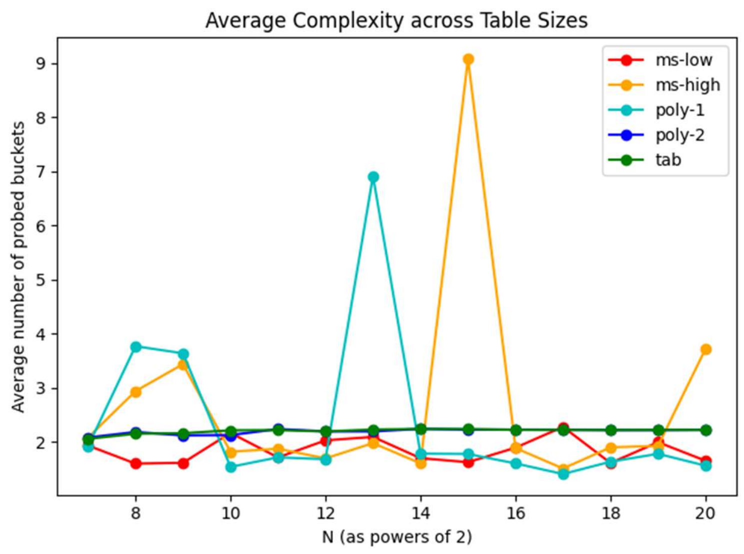

Figure 1: Growth Experiment Average Complexity

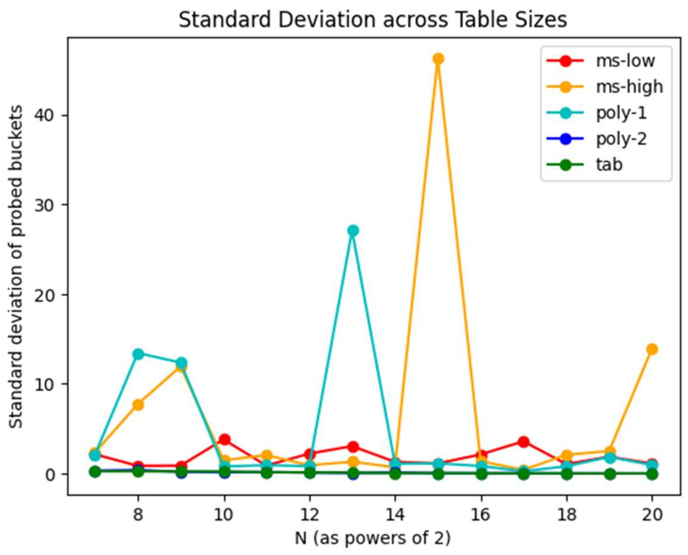

Figure 2: Growth Experiment Standard Deviation

The two figures chart essentially the same graph for each hashing function. This corresponds to the facts of the test, as each point on the graph is itself a new table and function, when poor average results show, this means the worst case scenarios is occuring, wherein a few instances of poor results generate a large standard deviation, which result in a larger average probes per operation. Because of this monotonicity between graphs, I will mainly consider the trajectories of each plot.

First, the most consistent and therefore simplest results. Tabulation hashing remains constant across all sizes of N, whilst minimising standard deviation, suggesting that an upper bound is holding its performance at that level. This is confirmed by results on linear probing claimed within the lectures, that for $m \geq (1+\epsilon)n$, the expected number of probs is bounded by $\mathcal{O}(\frac{1}{\epsilon^2})$. As this is an upper bound on a constant result, standard deviation remains small and constant across the multiple experiments. Note, as per the lectures, while tabulations hashing is only 3-independent, this performance holds even though constant expcted number of probes is theoretically for hash functions shosen from a 5-independent family.

For the polynomial family, lecture theory makes the claim that for $m \geq (1+\epsilon)n$, the expected number of probes during operation for a 2-independent family is $\Omega(\sqrt{n})$. Indeed, the function of degree 1 performs erratically, occasionally with performance analogous or even better than tabulation, but frequently significantly worse. We can attribute this to the function being (2,4)-independent, which corresponds to the claim about linear probing hash functions of 2-independence having a lower bound of $\Omega(\sqrt{n})$ on expected number of probes during an operation. Notably, the curve doesn't conform to such a lower bound, but we can attribute this to the *probabilistic* nature of the bound, that it conforms to *expected* number of probes. Taking this fact into account allows us to understand the irregularities within the plot, accounting for good and bad performance. Indeed, the worst average results within this plot can be understood as increasing over n. Notably, this erratic performance is in stark contrast to the great results from a polynomial of degree 2. By the lectures, this family is (2,1)-independent, and thus still accords to the expected lower bound of $\Omega(\sqrt{n})$. Why the stark difference? Well, we can attribute this to the stronger form of universality the function holds, making it closer to a random function. Even so, we might wonder why it so closely follows tabulation hash's properties. I hypothesise that it too likely holds properties that allow its performance to be better than its independence maintains.

For the multiply-shift hashing, lecture theory makes the claim that for $m \geq (1+\epsilon)n$ their expected number of probes during operation for is $\Omega(\log n)$. We see this performance irregularity in the high multiply-shift function, which for the most part remains consistently below the performance of polynomial of degree 1, and thus conforms to the better lower bound. However, it should be observed that this function results in the worst instance of average complexity and standard deviation. We can attribute this to our claims about polynomial and multiply-shift functions conforming to a lower bound of big Ω. Thus, whilst multiply shift perform better overall, there remains the potential for worse performance, as no upper bound is stated. Again, as with the polynomial family, when comparing the high multiply shift to the low multiply shift, there is a stark contrast. For the most part, the low-multiply shift function performs best in terms of average complexity over all iterations, but has a worse standard deviation that tabulation and polynomial of degree 2, pointing to irregularity confirmed by its lower bound complexity. The main question is why this multiply-shift function performs better than its sibling function? We can attribute this to where the bits to provide randomness are taken from. Despite the high function having a greater word size, it essentially takes from the middle of the word, thus the bits form randomness from only half a word. On the other hand, even though the representation is smaller, the low function takes from end of a word, where the dependence is thus over the entire representation and is therefore more random. Considering this, we could therefore achieve better performance if we combined the best of each function, if we took the top bits of the 64-bit word, the representation would indeed be the most random.

Usage Experiment

For the second experiment, *usage*, a hash table of size 2^{20} is created, insertions are perfomed to increase the usage of the table by 1% up until 90%, across 40 instances, the efficiency of the insert is reported in terms of average number of probed buckets per operation, and the standard deviation of these instances. Again, for the sake of simplicity, I will consider the performance of each function independently, and make comparative claims mainly around families of functions. The experimental results can be viewed in figures 3 and 4 below:

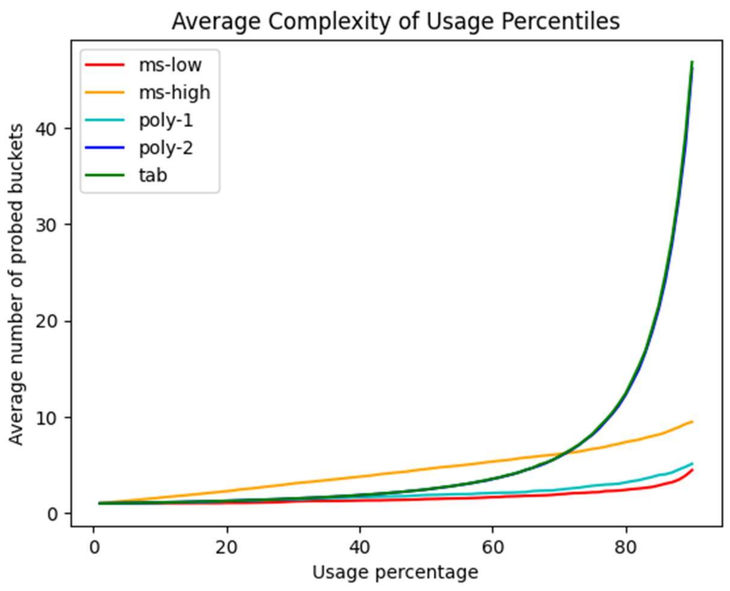

Figure 3: Usage Experiment Average Complexity

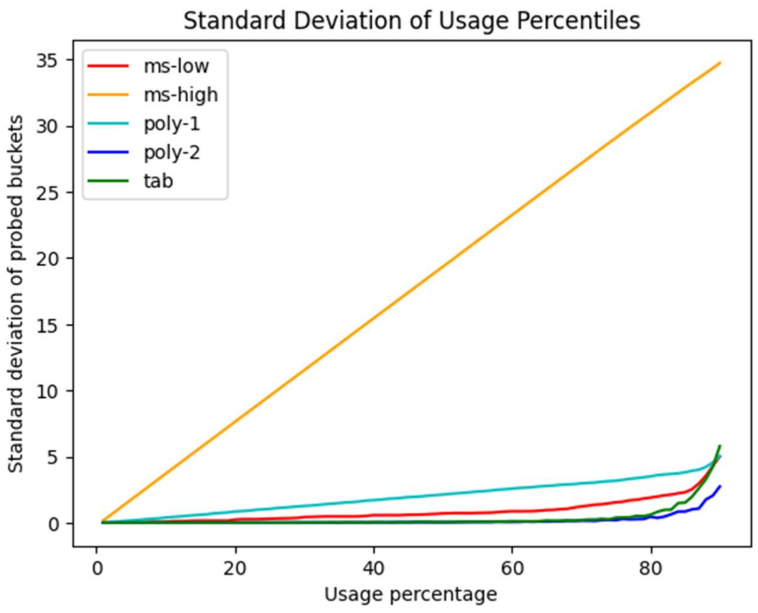

Figure 4: Usage Experiment Standard Deviation

The essential difference between this experiment and the previous one is an inversion of the constancy of the table. Here, the size remains constant, but we're testing how each function degrades with the occupancy of the table. The main thing I will be considering is how the complexity results shown above cease as we increase the usage percentage. This evaluation will thus be more general, comparing functions to one another rather than claims maintained within the lecture.

Notably, the tabulation hash degrades exponentially after usage exceeds 50-60%. This point corresponds with constant results shown in the growth experiment. But how do we reconcile this horribly performance with expected constancy of average operations? The tabulation function's standard deviation remains essentially constant up until exceeding 80% usage. Thus, even though performance degrades, it does so at a consistent rate, revealing that the performance decline is itself constant across the increase in usage. Curiously, the polynomial of degree 2 again displays similar results to tabulation hash. Again, this is counter to the expected complexity as stated within the lectures of a 2-independent function. However, we could hypothesise that similar to tabulation, despite the indepdence not being at the required k=5, this function has extra properties similar to tabulation hashing that allows its performance to be analogously good. Indeed, I further hypothesise that these two functions could be described as having these properties for usage under 60%, wherein after this limit is exceeded, their performance degrades to their the complexity of their independence.

The two functions, polynomial of degree 1, and low multiply shift, perform in accordance with their complexity described above. That is, the later has both lower average number of probes, despite the increasing usage, as well as lower standard deviation, which conforms to it having the better lower bound. Lastly, the multiply shift function performs as expected within the average complexity, performing the worst up until the extreme degradation of tabulation hash and polynomial of degree 2. This accords with our observations and hypothesis surrounding its lack of randomness when compared with its sibling function. However, the most striking observation is in its standard deviation which is linear across increasing usage. That is, the degree to which the function ranges increases a linear function of usage. Again, in accordance with my above hypothesis, I perceive this as correlating to a lack of randomness in taking the middle bits of a word (regardless of its size).