1. Consider a simple hash table with chaining containing 5,000 items, stored in 10,000 buckets. What is the expected number of empty buckets? What is the expected number of buckets with 1 item? 2 items? 5 items?

The probability that a bucket is empty is \((1-1/10,000)^{5,000} \approx e^{-1/2} \approx 0.6065\). Since there are 10,000 buckets, we expect 6,065 empty buckets. The probability that a bucket has one item is

\[
\binom{n}{1} \left( \frac{1}{b} \right)^1 \left( 1 - \frac{1}{b} \right)^{n-1}
\]

where \(b=5,000\) and \(n=10,000\). This is \(\approx 0.3033\), so we expect to have 3,033 buckets with 1 item. The probability that a bucket has two items is

\[
\binom{n}{2} \left( \frac{1}{b} \right)^2 \left( 1 - \frac{1}{b} \right)^{n-2}
\]

This is \(\approx 0.0758\), so we expect to have 758 buckets with 2 items. The probability that a bucket has five items is

\[
\binom{n}{5} \left( \frac{1}{b} \right)^5 \left( 1 - \frac{1}{b} \right)^{n-5}
\]

This is \(\approx 0.00016\), so we expect to have 1 or 2 buckets with 5 items.

Suppose that it takes 100 ns to retrieve one list item from memory in a chained hash table. Suppose that a long stream of arriving packets all access the last item in the longest chain of the hash table. Estimate the number of packets per second that can be forwarded under these conditions.

The longest bucket is likely to have 6-8 items in its list. So we can expect to spend 600-800 ns to process each packet in the case described. This leads to a packet forwarding rate of 1.25-1.67 million packets per second.

2. Consider the design of a 2-left hash table for a conventional processor with a cache line size of 128 bytes and assume that we can pack a 20 bit pointer and a 12 bit fingerprint into a single 32 bit word. Also assume that scanning a bucket for a matching fingerprint takes \(k\) ns where \(k\) is the number of items that can be stored in each bucket. Assume also that it takes 100 ns to retrieve one word from memory 102 ns to retrieve two words from the same cache line, 104 ns to retrieve three words in the same cache line, etc. How much time is required to perform a lookup in the worst-case if \(k=8\)? What if \(k=16\)? What if \(k=32\)? Assume that there are no false positive fingerprint matches. Also, you may ignore the time to retrieve the actual table entry, once you have found the pointer to it.

In the case of \(k=8\), we spend 232 ns retrieving the two buckets from main memory and 16 ns checking all the fingerprints in both buckets for a total of 248 ns. When \(k=16\), we spend 264+32=296 ns. When \(k=32\), we spend 328+64=392 ns.
Suppose you want to be able to implement 100,000 table entries into a 2-left hash table with \( k=8 \) and be confident that there is no overflow? Use the theoretical analysis on page 7 of the notes to determine the total number of buckets required. How does this change if \( k=16 \) and \( k=32 \)?

For \( k=8 \), we need to find values of \( h \) for which \( h+1.05*\ln \ln (n/h) \leq 8 \) where \( n=100,000 \) and \( n/h \) is the total number of buckets. Plugging in values of 4, 5 and 6 in \( h+1.05*\ln \ln (n/h) \), we get values of 6.4, 7.4 and 8.3. So, \( h=6 \) is too large to ensure we don’t overflow, \( h=5 \) is probably a safe value and \( h=4 \) certainly is. Using \( h=5 \) gives us 20,000 buckets, and a space efficiency of \( 5/8=62.5\% \). Note that \( h \) need not be an integer, so if we wanted to accept a higher risk of overflow, we might choose \( h=5.6 \), which makes the expression \( h+1.05*\ln \ln (n/h)=8 \). This gives 17,858 buckets and improves the space efficiency to \( 5.6/8=70\% \).

For \( k=16 \), we need to find values of \( h \) for which \( h+1.05*\ln \ln (n/h) \leq 16 \). Plugging in values of 12, 13 and 14 in \( h+1.05*\ln \ln (n/h) \), we get values of 14.3, 15.3 and 16.3, so \( h=13 \) is a reasonable choice, giving us 7,693 buckets and a space efficiency of 81.25%.

For \( k=32 \), we need to find values of \( h \) for which \( h+1.05*\ln \ln (n/h) \leq 32 \). Plugging in values of 28, 29 and 30 in \( h+1.05*\ln \ln (n/h) \), we get values of 30.2, 31.2 and 32.2, so \( h=29 \) is a reasonable choice, giving us 3,449 buckets and a space efficiency of 90.6%.

Based on the results of your analysis, what choice of \( k \) would you choose for a hash table used in a router or switch in which you need to be able to process 3 million packets per second.

To process 3 million packets per second, we need to be able to complete the hash lookup in less than 333 ns. Based on our simple analysis above, either \( k=8 \) or \( k=16 \) are acceptable options; \( k=8 \) is a bit less space efficient, but the absolute difference in memory is \( 8*17,858 – 16*7693=19,776 \) words or less than 80 KB.