CS174 Fall 98: Lecture Note 9 Supplement

Alistair Sinclair, November 1998

Hashing in Worst-Case Constant Time

We have seen that, if the hash function h is chosen u.a.r. from a 2-universal family $\mathcal{H}$, then the expected cost for any dictionary operation on a set S is $1 + \frac{|S|}{n} \leq 2$, provided we ensure that the the hash table size n is at least as large as $|S|$. Therefore, the expected cost of any sequence of k dictionary operations is at most $2k$. Unfortunately this says nothing about the cost of the worst operation in the sequence: there might be a small number of operations that are very slow.

Ex: To see this point, give a simple example of positive integer-valued random variables $X_1, X_2, \ldots, X_k$ each of which has constant expectation but such that, with probability 1, $\mathrm{E}(\max_i X_i)$ grows with k. Obviously your random variables will not be independent. □

There is a clever way of getting around this problem, which was discovered by Fredman, Komlós and Szemerédi in 1984. Their scheme achieves constant time for *every* operation in a sequence, though it works only in the more restricted model where we assume that all INSERT operations precede all FINDs and there are no DELETEs. (This is usually referred to as a static dictionary.)

The high-level idea is known as "double hashing": we apply two hash functions to each key. More precisely, we first apply a *primary* hash function to all keys. Then, for each group of keys that collide under the primary hash function, we apply a *secondary* hash function to separate that group. All hash functions used will be chosen from a 2-universal family $\mathcal{H}$ such as the one defined on page 2 of Note 9.

First, let's consider the secondary hash functions. Suppose we want to completely separate a set of b keys, i.e., we want to hash them with *no collisions*. To be precise, let's define a collision to be a pair of keys $\{x, y\}$ such that $h(x) = h(y)$.

Claim 1: If we use a 2-universal hash family to hash a set of b keys into a table of size b^2, then $\Pr[\text{there is a collision}] \leq \frac{1}{2}$.

Proof: By definition of a 2-universal family, we know that

$$\Pr[h(x) = h(y)] \leq \tfrac{1}{b^2} \qquad \text{for every pair } \{x, y\},$$

since the table size is $n = b^2$. But there are exactly $\binom{b}{2}$ pairs $\{x, y\}$, so by linearity of expectation we get

$$\mathrm{E}(\#\text{ collisions}) \leq \tbinom{b}{2} \times \tfrac{1}{b^2} = \tfrac{b(b-1)}{2b^2} < \tfrac{1}{2}.$$

Now by Markov's inequality we conclude the claimed result. □

Ex: How does this result relate to the fact about collisions we established about balls and bins on page 3 of Note 1? □

The above claim tells us how to do the secondary hashing: suppose the primary hash function creates "buckets" (i.e., groups of colliding keys) of sizes $b_1, b_2, \ldots, b_r$ (for some r). Then for each group i, we pick a secondary hash function h_i from a 2-universal family with table size b_i^2, where b_i is the size of the group. This function hashes the keys in group i to a secondary table of size b_i^2, and by Claim 1 it will create a collision with probability at most $\frac{1}{2}$. So we can keep trying random secondary functions until we find one that works (i.e., has no collisions) for the group; the expected number of trials we'll need is at most 2.

Once we have all the required secondary functions, we store them in the cells of the primary hash table. To hash a key, we first use the primary hash function to identify its group (cell in the primary

table), and then use the secondary function stored there to hash the key to its secondary table, where the key itself is stored.

Let's turn now to the primary hash function. What properties do we need this function to have? Well, notice that the size of the above data structure will be proportional to the size of the primary table plus $\sum_i b_i^2$ (which is the total size of all the secondary tables). So, to be consistent with our requirement that the data structure size be proportional to the size of the stored set S, we need that the quantity $\sum_i b_i^2$ (the sum of squares of the bucket sizes) is at most const $\times |S|$, while the size of the primary table is also of this order of magnitude.

Once again, this is easy to achieve using 2-universal hash functions:

Claim 2: If we use a 2-universal hash family to hash a set S into a table of size $n \geq |S|$, then the bucket sizes b_i satisfy

$$\Pr[\textstyle\sum_i b_i^2 \geq 4|S|\,] \leq \frac{1}{2}.$$

Proof: Note that $\sum_i b_i^2$ is exactly equal to $2 \times (\#\text{collisions}) + \sum_i b_i$. (Why?) Using linearity of expectation, and the fact that $\Pr[h(x) = h(y)] \leq \frac{1}{n}$, we therefore get

$$\mathrm{E}\left(\textstyle\sum_i b_i^2\right) = 2\mathrm{E}\left(\#\text{collisions}\right) + \mathrm{E}\left(\sum_i b_i\right) \leq 2\binom{|S|}{2} \times \frac{1}{n} + |S| \leq 2|S|,$$

using the fact that $|S| \leq n$. Then a simple application of Markov's inequality gives us the claim. □

Just as for the secondary hash functions, we can pick primary hash functions from a 2-universal family with table size $n \approx |S|$ until we find one for which $\sum_i b_i^2 \leq 4|S|$. By Claim 2, we expect to have to try at most twice.

Putting it all together, once we have picked primary and secondary hash functions satisfying Claims 2 and 1 respectively, we have constructed a data structure that supports *any* FIND operation on our set S in constant time. The size of the data structure is proportional to $|S|$, and the expected time to construct it is also proportional to $|S|$.