Analysis of Algorithms I: Perfect Hashing

Xi Chen

Columbia University

Goal: Let $U = \{0, 1, \ldots, p-1\}$ be a huge universe set. Given a static subset $V \subset U$ of n keys (here "static" means we will never change the set V by inserting or deleting keys, e.g., consider

the $100,000$ *most popular google keywords in 2011*

And we want to store it in a data structure (and maybe burn the data structure onto a CD). Is there such a data structure so that search can be done in $O(1)$ time in the worst case?

$$\text{Search}(x), \quad \text{where } x \in U$$

returns 1 if $x \in V$ and 0 otherwise.

First try: Use a hash table $T[0, 1, \ldots, m-1]$ of size m; randomly pick a hash function h from a universal collection H; and then insert the keys of V into the table T using h?

Wishful thinking: If we are lucky and there is no collision at all between keys in V, then search takes $O(1)$ time in the worst case.

Question: How large does the table need to be? so that with high probability, say $> 1/2$, the hash function we get has no collision.

Recall the definition of a universal collection H:

Definition

Let H be a collection of hash functions from U to $\{0, \ldots, m-1\}$. We say it is universal if for any two distinct keys x and y from U: $[\,\textit{the number of functions } h \in H \textit{ such that } h(x) = h(y)\,] \le |H|/m$.

Here is an equivalent definition: If we pick a hash function h from H uniformly at random (each with probability $1/|H|$), then

$$\Pr\big[h(x) = h(y)\big] \le 1/m, \quad \text{for all } x \neq y \in U$$

That is, for any two keys x and y, the probability that there is collision between them (with respect to h) is bounded by $1/m$.

Theorem

Set the table size $m = n^2$. Assume H is a universal collection of hash functions from U to $\{0, 1, \ldots, m-1\}$. If we pick a hash function h from H uniformly at random, then

$$\Pr\left[\textit{no collision between keys in } V \textit{ w.r.t. } h\right] > 1/2$$

We use $k_1, \ldots, k_n$ to denote the keys in V and use $X_{i,j}$, where $i, j : 1 \leq i < j \leq n$, to denote the following indicator variable: $X_{i,j} = 1$ if there is a collision between k_i and k_j; and $X_{i,j} = 0$ otherwise. We also use X to denote the total number of pairs of keys k_i and k_j, $i, j : 1 \leq i < j \leq n$, that collide.

Then we have $X = \sum_{i,j} X_{i,j}$ and thus,

$$E\left[X\right] = \sum_{i,j} E\left[X_{i,j}\right] = \binom{n}{2} \cdot (1/m)$$

where the first equation uses the linearity of expectations and the second equation uses the assumption that H is universal. Because we set $m = n^2$, it is clear that the expectation of X is $< 1/2$. From this, we use Markov's inequality (next slide) to claim that

$$\Pr\left[\text{no collision at all}\right] = \Pr\left[X = 0\right] > 1/2$$

Theorem (Markov's inequality)

Let X be a nonnegative random variable. Then we have

$$\Pr\left[X \geq t\right] \leq E\left[X\right]/t, \quad \textit{for all } t > 0$$

For example, in the last slide, by setting $t = 1$ we get

$$\Pr\left[X \geq 1\right] \leq E\left[X\right]/1 < 1/2$$

And thus, $\Pr\left[X = 0\right] = 1 - \Pr\left[X \geq 1\right] > 1/2$ because X here is a integer random variable.

Proof of Markov's inequality.

We use S to denote all possible values of X. Because X is nonnegative, we may assume that every $x \in S$ is nonnegative.

$$\begin{aligned}
E\left[X\right] &= \sum_{x \in S} x \cdot \Pr\left[X = x\right] \\
&= \sum_{x < t} x \cdot \Pr\left[X = x\right] + \sum_{x \geq t} x \cdot \Pr\left[X = x\right] \\
&\geq \sum_{x \geq t} t \cdot \Pr\left[X = x\right] \\
&= t \cdot \Pr\left[X \geq t\right]
\end{aligned}$$

$\square$

To summarize, if we set the table size to be $m = n^2$, we can randomly pick a universal hash function h and map the keys of V to the table and with probability $> 1/2$, there is no collision at all. (Intuitively this is because the table is much larger than V.) So if the use of quadratic space is fine, we are done because

1. We can find a hash function with no collision very efficiently. If the first hash function we pick from H has collision, pick another uniformly at random. Unless we are extremely unlucky, a hash function with no collision can be found after picking a few randomly.
2. Once we get a hash function with no collision and map keys of V to the table, Search takes $O(1)$ time in the worst case.

Perfect Hashing: A data structure that only uses $O(n)$ space and can handle Search in $O(1)$ time in the worst case.

The key idea here: Use two levels of Universal Hashing; and make sure there is no collision at the second level.

1. Set up a hash table T of size $m = n$, called the level-1 hash table. Pick a hash function h from a universal collection uniformly at random. As we showed earlier, there will be a lot of collisions when we map the keys in V to the n slots using h. We use V_i, $i \in \{0, 1, \ldots, n-1\}$, to denote the set of keys mapped to slot i of T, and let $t_i = |V_i|$.

2. Next for each slot i, (key difference) instead of creating a list at slot i to link the t_i keys in V_i, we create a new hash table T_i of size t_i^2, called the level-2 hash table at slot i. “Find” a hash function $h_i : U \rightarrow \{0, 1, \ldots, t_i^2 - 1\}$ to map the t_i keys of V_i to table T_i “with no collision in T_i at all”.

First, this two-level hashing data structure can be constructed efficiently. This is because in Step 2, finding a hash function h_i that has no collision between keys in V_i is “easy”. Just pick a hash function from a universal collection H from U to $\{0, 1, \ldots, n_i^2 - 1\}$ and by the theorem on slide 5, with $> 1/2$ probability there will be no collision in T_i. So just randomly pick a few and we can find such a hash function h_i efficiently.

Second, because there is no collision in level 2, Search takes $O(1)$ time. Given a key $k \in U$, first use h to map it to a slot in T, say slot $i = h(k)$. Now we know that k may only appear in the level-2 hash table T_i at slot i of T. Next we use h_i to map k to a slot in T_i, say slot $j = h_i(k)$. The only thing left is to check whether $T_i[j]$ is the key we look for (because there is no collision in T_i so we don't need to use chaining in T_i). This clearly takes only constant many steps, assuming the hash functions can be evaluated in constant steps.

Finally, space wise is this data structure good? It is indeed perfect! To see this, the total space used by the $n + 1$ hash tables is

$$n + t_1^2 + t_2^2 + \cdots + t_n^2$$

Note that $t_1, \ldots, t_n$ are all random variables that depend on the very first hash function h we picked. Let $X = t_1^2 + t_2^2 + \cdots + t_n^2$ and we show that $E\left[X\right] < 2n$. Using Markov's inequality, we get

$$\Pr\left[X \geq 4n\right] \leq E\left[X\right]/4n < 1/2$$

So with probability $> 1/2$, the total space used by all the $n + 1$ hash tables are no more than $5n$. WoW ...

To prove that $E\left[X\right] < 2n$, we have

$$X = \sum_{i=1}^{n} t_i^2 = \sum_{i=1}^{n} \left(t_i + 2\binom{t_i}{2} \right) = n + 2\sum_{i=1}^{n} \binom{t_i}{2}$$

But what is the sum

$$\sum_{i=1}^{n} \binom{t_i}{2}$$

This is exactly the number of pairs of keys k_i and k_j from V, $i, j : 1 \le i < j \le n$, that collide, because there are exactly $\binom{t_1}{2}$ pairs that collide at slot 1, ..., $\binom{t_n}{2}$ pairs that collide at slot n.

But at the beginning (check the proof of the first theorem), we have already showed that the expectation of this number is

$$\binom{n}{2} \cdot (1/m)$$

when the hash table has m slots. Here we have $m = n$ because the size of T is set to be n. As a result, we have

$$E\left[\sum_i t_i^2\right] = n + 2 \cdot \binom{n}{2} \cdot (1/n) < 2n$$