Perfect Hashing
(Notes for ESTR2102)

Yufei Tao

CSE Dept
CUHK

In this lecture, we will revisit the approach of using a hash table to answer dictionary search queries. Recall that currently we can answer a query in $O(1)$ expected time with a hash table of $O(n)$ space that can be constructed in $O(n)$ time (where n is the size of the underlying set).

We will show that it is possible to improve the query time to $O(1)$ in the worst case without affecting the space cost. The tradeoff is that the construction time becomes $O(n)$ expected.

Recall:

The Dictionary Search Problem

S is a set of n integers in $[U]$ (recall that $[x]$ denotes the set of integers $\{1, 2, ..., x\}$). We want to preprocess S into a data structure so that queries of the following form can be answered efficiently:

- Given a value v, a query asks whether $v \in S$.

Recall:

Hash Function

Let U and m be positive integers.

A hash function is a function h that maps $[U]$ to $[m]$, namely, for any integer $k \in [U]$, $h(k)$ returns a value in $[m]$.

Recall:

Universality

Let $\mathcal{H}$ be a family of hash functions. $\mathcal{H}$ is universal if the following holds:

Let k_1, k_2 be two distinct integers in $[U]$. By picking a function $h \in \mathcal{H}$ uniformly at random, we guarantee that

$$\boldsymbol{Pr}[h(k_1) = h(k_2)] \quad \leq \quad 1/m.$$

Recall:

A Universal Family

Pick a prime number p such that $p \geq \max\{U, m\}$. Choose an integer α uniformly at random from $\{1, 2, ..., p-1\}$, and an integer β uniformly at random from $\{0, 1, ..., p-1\}$. Design a hash function as:

$$h(k) \quad = \quad 1 + ((\alpha \cdot k + \beta) \mod p) \mod m$$

Markov Inequality

Theorem: Let X be a positive real-valued random variable. For any $t > 0$, it holds that

$$\boldsymbol{Pr}[X \geq t] \quad \leq \quad \boldsymbol{E}[X]/t.$$

Proof: Let $f(x)$ be the probability density function of X.

$$\begin{aligned}
\boldsymbol{Pr}[X \geq t] &= \int_t^\infty f(x)dx = \frac{1}{t}\int_t^\infty t \cdot f(x)dx \\
&\leq \frac{1}{t}\int_t^\infty x \cdot f(x)dx \\
&\leq \frac{1}{t}\int_0^\infty x \cdot f(x)dx \\
&= \boldsymbol{E}[X]/t.
\end{aligned}$$

$\square$

Quadratic m—Collision Free Hashing

In the main class, we said that we should set $m = \Theta(n)$ in order to achieve constant query time. Now we will challenge this conventional wisdom by choosing $m = n^2$.

Lemma 1: By picking $m = n^2$, the following holds with probability at least $1/2$: every linked list in the hash table has length at most 1.

We actually already proved this in discussing the birthday's paradox. The proof is included again in the next slide for your convenience.

Quadratic m—Collision Free Hashing

Proof: We will prove that with probability at least $1/2$, no two integers in S get hashed to the same value. Define X_{ij} to be 1 if the i-th element and j-th element have the same hash value. By universality, we know that $\boldsymbol{Pr}[X_{ij} = 1] \leq 1/m$. It thus follows that $\boldsymbol{E}[X_{ij}] \leq 1/m$. Define:

$$X = \sum_{i,j \text{ s.t. } i<j} X_{ij}.$$

Note that the summation is on $n(n-1)/2$ pairs of (i,j). Hence, $\boldsymbol{E}[X] \leq n(n-1)/(2m) < 1/2$. By the Markov inequality, we know that

$$\boldsymbol{Pr}[X \geq 1] \leq 1/2.$$

Since X is an integer, it follows that with probability at least $1/2$, $X = 0$, namely, no two elements in S have the same hash value. $\square$

It is clear that we can obtain such a collision free hash table with $m = n^2$ by 2 trials in expectation (as each trial succeeds with probability $1/2$).

Doesn't this already ensure $O(1)$ query time in the worst case? Yes, but unfortunately, setting $m = n^2$ incurs $\Omega(n^2)$ space! Next, we will bring the space back down to $O(n)$ using an idea called double hashing.

Double Hashing

Set $m = n$.

Choose a hash function $h : [U] \to [m]$ randomly from our universal family. Compute the hash value of every integer in S.

Let S_i $(1 \le i \le m)$ be $\{k \in S \mid h(k) = i\}$. Define $n_i = |S_i|$.

If $\sum_{i=1}^{m} n_i^2 > 4n$, we declare a global failure, and repeat from scratch by choosing another h randomly.

Otherwise, proceed to the next slide.

Double Hashing

So now we have $\sum_{i=1}^{m} n_i^2 \leq 4n$.

For every $i \in [1, m]$, we create a hash table T_i for S_i as follows:

1. Set $m_i = n_i^2$.
2. Choose a hash function $h_i : U \to [m_i]$ randomly from our universal family.
3. Create T_i based on h_i.
4. If any linked list in T_i has length at least 2, declare an i-local failure, and repeat from Step 2.

Note that the final T_i is collision free, namely, every linked list therein has a length at most 1.

Space consumption is $O(\sum_{i=1}^{m} n_i^2) = O(n)$.

Query

Given a dictionary search query with search value q, we answer it as follows:

- Compute $i = h(q)$.
- Compute $j = h_i(q)$.
- Scan the linked list of T_i for value j – note that the linked list contains at most 1 element.
- Report "yes" if q is in the linked list, or "no" otherwise.

The query time is clearly $O(1)$.

Next we will prove the most non-trivial fact: the construction time is $O(n)$ in expectation. What is the major obstacle in the proof? Note that global failure sustains until we get $\sum_{i=1}^{m} n_i^2 \leq 4n$. This inequality appears rather difficult to ensure, because we know $\sum_{i=1}^{m} n_i = n$! Nonetheless, as shown in the next, the inequality actually holds with probability at least $1/2$.

Lemma: $\boldsymbol{Pr}[\sum_{i=1}^{m} n_i^2 > 4n] \le 1/2$.

Proof: We will prove that $\boldsymbol{E}[\sum_{i=1}^{m} n_i^2] \le 2n$, after which the lemma will follow from the Markov inequality.

Define X_{ij} to be 1 if the i-th element and j-th element have the same hash value under h. By universality and $m = n$, we know that $\boldsymbol{Pr}[X_{ij} = 1] \le 1/n$. It thus follows that $\boldsymbol{E}[X_{ij}] \le 1/n$. Define:

$$X = \sum_{i,j \text{ s.t. } i<j} X_{ij}.$$

In other words, X is the number of distinct pairs of elements that collide in their hash values.

Clearly, $\boldsymbol{E}[X] \le (n(n-1)/2) \cdot (1/n) = (n-1)/2$.

Let us now compare $\sum_{i=1}^{m} n_i^2$ to X. Recall that n_i is the size of S_i, i.e., the set of elements that obtain hash value i under h. Hence, S_i should contribute $n_i(n_i - 1)/2$ to X. It follows that

$$\begin{aligned} X &= \sum_{i=1}^{m} \frac{n_i(n_i - 1)}{2} = \frac{1}{2}\left(\sum_{i=1}^{m} n_i^2 - \sum_{i=1}^{m} n_i\right) \\ &= \frac{1}{2}\sum_{i=1}^{m} n_i^2 - \frac{n}{2}. \end{aligned}$$

Hence:

$$\sum_{i=1}^{m} n_i^2 \leq 2X + n$$

indicating that $\boldsymbol{E}[\sum_{i=1}^{m} n_i^2] \leq 2\boldsymbol{E}[X] + n \leq 2n - 1$. $\square$

Construction Time

Now we can proceed to analyze the expected time of constructing our hash table.

From the previous lemma, we know that we expect to have only 1 global failure before $\sum_{i=1}^{m} n_i^2 \leq 4n$ holds (i.e., 2 trials, each with success probability at least $1/2$). Hence, the decision of h takes only $O(n)$ time in expectation.

It remains to analyze the time of creating each T_i. We have already done so – recall that we have $1/2$ probability of success by choosing a quadratic $m_i = n_i^2$. In other words, we expect only 1 i-local failure. The time of building T_i is therefore $O(n_i)$ expected.

The total cost of building all of $T_1, T_2, ..., T_n$ is therefore $O(\sum_{i=1}^{n} n_i) = O(n)$ in expectation.