Advanced Algorithm Design: Hashing and Applications to Compact Data Representation

Lectured by Prof. Moses Charikar
Transcribed by John McSpedon[*]

Feb 11th, 2013

1 Cuckoo Hashing

Recall from last lecture the dictionary *cuckoo hashing*. It grows in size linear to the number of elements and has worst case constant lookup time. Here we have:

- n elements.
- table size of $m = 4n$.
- two hash functions $h_1(x)$ and $h_2(x)$.

An element x is stored in either $T[h_1(x)]$ or $T[h_2(x)]$. There are no linked-lists nor is there linear probing. Insertions are the only nontrival procedure. For this we use:

```
procedure INSERT($x$)
    if LOOKUP($x$) then return
    loop MaxLoop times
        SWAP($x$, $T[h_1(x)]$)
        if $x$ = NULL then return
        SWAP($x$, $T[h_2(x)]$)
        if $x$ = NULL then return
    end loop
    REHASH( ); INSERT(x)
end procedure
```

For our analysis, we will set MaxLoop such that our algorithm rehashes after performing $6\log(n)$ swaps without terminating. To estimate the probability of rehashing, we must consider three cases separately: while attempting to insert, we may encounter a chain of $6\log(n)$ distinct elements; we may encounter a

[*]mcspedon@princeton.edu

chain of $6\log(n)$ evictions but with some repeated elements, i.e. one cycle; or we may encounter two such cycles (in which case even letting MaxLoop$\to\infty$ would not guarantee termination).

Case 1: No Cycles

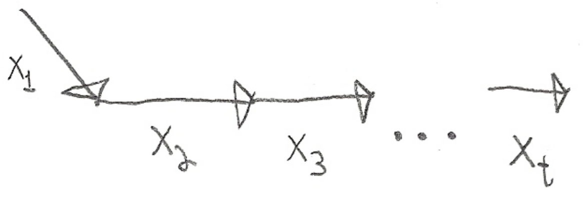


Figure 1: An insertion without cycles

Let x_1 be the element we are trying to insert. We can say :

$$\begin{aligned}\Pr[\text{one eviction}] &\leq \Pr[\exists\ y \mid (h_1(x) = h_1(y)) \text{ or } (h_1(x) = h_2(y))] \\ &\leq n\cdot(\frac{1}{m}+\frac{1}{m}) = \frac{2n}{m} = \frac{2n}{4n} = \frac{1}{2}\end{aligned}$$

So if our hash functions are truly random, then:

$$\Pr[t \text{ evictions}] \leq \frac{1}{2^t} \tag{1.1}$$

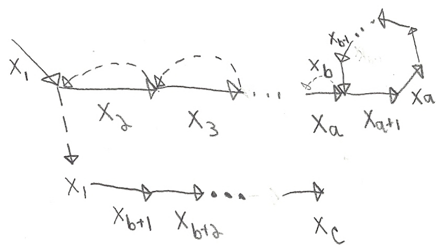
Hence for Case 1 we can say:

$$\Pr[\text{rehashing}] = \Pr[6\log(n) \text{ evictions}] \leq \frac{1}{2^{6\log(n)}} \in \mathcal{O}\left(\frac{1}{n^6}\right)$$

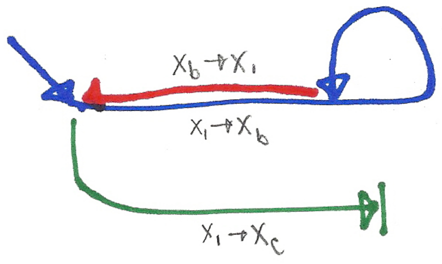
Ignoring the possibility of rehashing, we can also say that the expected time for insertions here is:

$$\mathbb{E}[\text{insertion time}] \leq \sum_t t\cdot 2^{-t} \leq \sum_t 2^{-t} \in \mathcal{O}\left(1\right)$$

Case 2: One Cycle



(a) Insertion with one cycle



(b) Three "chains" of distinct elements

Figure 2

Here we can reuse equation (1.1) from Case 1. To do so, note that the figure above can be divided into three separate "chains" each composed of distinct elements: $\{x_1 \to x_b, x_b \to x_1, x_1 \to x_c\}$. If our procedure requires t swaps before insertion (i.e. t arrows in the above figure), then at least one the separate chains is of size $\geq \frac{t}{3}$. Hence $\Pr[\text{Case 2 requires} \geq t \text{ steps}] \leq \frac{1}{2^{\frac{t}{3}}}$. Accordingly:

$$\begin{aligned}
\Pr[\text{rehash}] &= \Pr[\text{Case 2 requires} \geq 6\log(n) \text{ steps}] \leq \frac{1}{2^{\frac{6\log(n)}{3}}} = \frac{1}{2^{2\log(n)}} \in \mathcal{O}\left(\frac{1}{n^2}\right) \\
\mathbb{E}[\text{insert time}] &\leq \sum_t \frac{t}{2^{\frac{t}{3}}} \in \mathcal{O}(1)
\end{aligned}$$

Case 3: Two Cycles

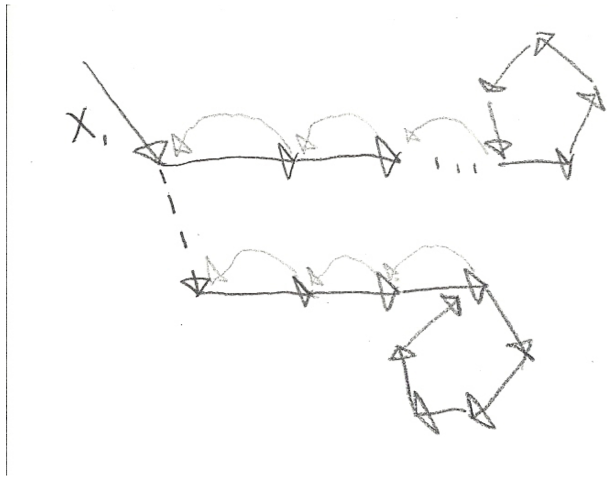


Figure 3: An insertion with 2 cycles

The last case to consider is that with two cycles. In the absence of a MaxLoop cut-off, we would infinite loop when attempting to insert in this situation – therefore having two cycles guarantees that we rehash. We can approximate the chance of this occurring by cardinality. Here we are attempting to hash t elements into $t-1$ hash locations. For problematic configurations, we have:

- $t-1$ elements besides x_1 meaning there are approximately n^{t-1} elements to choose from
- t^3 choices for the start and end positions of the cycles
- m^{t-1} choices for hash locations

For unproblematic choices, we need two hash locations per element, hence we can estimate this with $\binom{m}{2}^t \in \mathcal{O}\left(\frac{m^{2t}}{2^t}\right)$. Combining this information we have:

$$\begin{aligned}
\Pr[\text{2 cycles in } t \text{ elements}] &\leq \frac{n^{t-1} \cdot m^{t-1} \cdot t^3}{\frac{m^{2t}}{2^t}} \\
&= \frac{n^{t-1} \cdot t^3 \cdot 2^t}{m^{t+1}} \\
&= \frac{n^{t-1} \cdot t^3 \cdot 2^t}{(4n)^{t+1}} \\
&\in \mathcal{O}\left(\frac{t^3}{n^2 2^t}\right)
\end{aligned}$$

It follows that:

$$\Pr[\text{2 cycle rehash}] \leq \sum_t \frac{t^3}{n^2 2^t} \in \mathcal{O}\left(\frac{1}{n^2}\right)$$

Synopsis

Through the above analysis we have shown Cuckoo Hashing provides constant lookup time along with expected constant insertion time. The analysis holds for $6\log(n)$-wise independent hash functions and there are constructions of such hash functions which take constant time. Alternatively, we can generalize the hash function from previous lectures to a polynomial: $f(x) = \sum_{i=0}^{k-1} a_i x^i \mod p$ is k-wise universal.

2 Min-wise Hashing for Set Similarity

Here we introduce algorithms that analyze data streams where the input is much larger than the algorithm's available memory. Typically we are allowed just one pass at the data ("online-" or "streaming-" algorithms are both common descriptors for this situation). In this example we consider a method used by the early search-engine contender AltaVista for measuring the similarity of two documents, perhaps to remove redundancy in search results.

We begin by replacing our input with a short "sketch", or summary of the data stream. For example, we could convert a document to a sketch by running a sliding window across the text and recording all consecutive 5-word phrases. In comparing two documents, we want to measure the similarity of two documents' respective sets by calculating their *Jaccard Coefficient*: for sets A and B we define $sim(A,B) = \frac{|A \cap B|}{|A \cup B|}$.

We now choose a random hash function h which returns, for example, a 4-byte integer.
Apply h to all elements in sets A and B then return $\min(h(a)) = \min(h(b))$.
If h is collision-free, then $\Pr[\min(h(a)) = \min(h(b))] = \frac{|A \cap B|}{|A \cup B|}$. To see this, observe that for $\min(h(a))$ to equal $\min(h(b))$, the minimum element of both sets must lie in $|A \cap B|$. Now note that the probability of arbitrary element (which we encounter) being the minimum is $\frac{1}{|A \cup B|}$.

Clearly we don't expect our documents to be strictly 0% or 100% matches, nor do we want to repeat this process every time we want to compare two documents. Instead we choose (for example) 100 distinct hash functions $\{h_1, h_2, \ldots h_{100}\}$. For any set A, we can store the sketch $(\min(h_1), \min(h_2), \ldots \min(h_{100}))$. Because $\Pr[\min(h_i(a)) = \min(h_i(b))] = \frac{|A \cap B|}{|A \cup B|}$, we can say the expected value for any given comparison of minimums (using the same hash function) is $\frac{|A \cap B|}{|A \cup B|}$, and we can estimate this value by calculating the proportion of matches between two documents' sketches.

Boosting Accuracy using Median-of-Means

In the above example, we approximated $sim(A, B)$ by averaging together several random variables with an expected value of $sim(A, B)$. In terms of precision, we can do better than this. Consider k random variables $x_1, x_2, ...x_k$ with $\mathbb{E}[x_i] = \mu$ and $\text{Var}[x_i] = \sigma^2$. Applying the same technique here as we did above, we effectively create a new random variable $y = \frac{x_1 + x_2 + ... + x_k}{k}$ with $\mathbb{E}[y] = \mu$ and $\text{Var}[y] = \frac{\sigma^2}{k}$. Here we can use Chebyshev's Inequality[1] to estimate the probability of our measurement deviating by a factor of ε from the true mean:

$$\begin{aligned} \Pr[|y - \mathbb{E}[y]| > \varepsilon\mathbb{E}[y]] &\leq \frac{\text{Var}[y]}{(\varepsilon\mathbb{E}[y])^2} \\ &= \frac{\text{Var}[x]}{\varepsilon^2 k(\mathbb{E}[x])^2} \\ &\leq \frac{1}{\varepsilon^2 k} \qquad (2.1) \end{aligned}$$

with (2.1) following from the inequality $Var[x] \leq (\mathbb{E}[x])^2$. Letting $k = \frac{4}{\varepsilon^2}$, we can say the probability of a bad estimate is $\leq \frac{1}{4}$.

We can further decrease the likelihood of a bad estimate by employing the "median-of-means" technique. For this, we separately average t groups of k variables (essentially forming t independent copies of the variable y above), then take the median of these t means.

$$\begin{aligned} y_1 &= [\frac{x_1 + x_2 + ... + x_k}{k}] \\ y_2 &= [\frac{x_{k+1} + x_{k+2} + ... + x_{2k}}{k}] \\ ... \\ y_t &= [\frac{x_{(t-1)k+1} + x_{(t-1)k+2} + ... + x_{tk}}{k}] \\ z &= med(y_1, y_2, ..., y_t) \end{aligned}$$

Suppose we want to guarantee our estimate is within ε of the true mean with probability $1-\delta$. Using just the variable y, by (2.1) we would need $k \geq \frac{1}{\varepsilon^2\delta}$. This becomes impractically large for small enough δ. We make the same guarantees

[1]Chebyshev's Theorem states that given a random variable X with finite $\mathbb{E}[X] = \mu$ and finite $\text{Var}[X] = \sigma^2 \neq 0$, then for any real $k > 0$:

$$\Pr[|X - \mu| \geq k\sigma] \leq \frac{1}{k^2}$$

but with a reduced sample size by using the median of means technique. The insight is that because z is the median of several variables, for z to be ε too high at least half of the y_i's must be ε too high (and for z to be too low at least half of the y_i's must be ε too low). Suppose k remains $\frac{4}{\varepsilon^2}$ (so that the probability of each $y_i \in z$ being more than ε off is $\leq \frac{1}{4}$. Then the expected number of bad y_i's is $\frac{t}{4}$. We can use this fact with Chernoff Bounds[2] to guarantee the probability that the number of bad estimates is at least $\frac{t}{2}$ is at most $e^{(-2\ln 2 - 2 + 1)t/4} < e^{(-t/11)}$. Hence we can choose t such that $t \geq 11\ln(\frac{1}{\delta})$. In short we now have a technique to construct a satisfactory estimate which requires space $\in \mathcal{O}\left(\log(\frac{1}{\delta})\right)$ instead of $\in \mathcal{O}\left(\frac{1}{\delta}\right)$.

[2]The Chernoff Bound we are interested in here is as follows:
Given $y_1, ...y_t$ *independent* random variables (perhaps from separate distributions), assume $0 \leq y_i \leq 1$ always $\forall i$,
and define $z = \sum\limits_i y_i,\ \mu = \mathbb{E}[z] = \sum\limits_i \mathbb{E}[y_i]$.
Then $\forall \varepsilon \geq 0$:

$$\Pr[z \geq (1+\varepsilon)\mu] \leq \exp(-\frac{\varepsilon^2}{2+\varepsilon}\mu)$$

$$\text{and,} \qquad \Pr[z \leq (1-\varepsilon)\mu] \leq \exp(-\frac{\varepsilon^2}{2}\mu)$$