COMP 480/580 — Probabilistic Algorithms and Data Structure Jan 17, 2018

Lecture 4

Lecturer: Anshumali Shrivastava *Scribe By: Jarrod Dunne*

1 Separate Chaining

Separate chaining is a hash table strategy where each bucket of the hash table points to a linked list, and elements with the same hash value are inserted into the linked list at that bucket.

In a hash table of size n with m objects inserted, the expected length of a chain is less than or equal to $1 + \frac{m-1}{n}$. The quantity $\frac{m}{n}$ is called the 'load factor', and denoted with a. The expected addition and search time is $1 + a$, and the worst case addition and search time is m. To improve this, we will use the following theorem:

Theorem: For the special case m=n, with probability at least 1 - $\frac{1}{n}$ the longest list is O($\frac{\ln(n)}{\ln(\ln(n)}$).

Proof: Let $X_{i,k}$ be an indicator representing key i hashing to slot k, with Pr[$X_{i,k}$=1]=$\frac{1}{n}$. Therefore (assuming a lot of independence), the probability a particular slot k receives $> \kappa$ keys is:

$$\binom{n}{\kappa}\frac{1}{m^\kappa} = \binom{n}{\kappa}\frac{1}{n^\kappa} < \frac{1}{\kappa!}$$

Setting $\kappa = 3\ \frac{\ln(n)}{\ln(\ln(n))}$, then $\kappa!\ > n^2$ and $\frac{1}{\kappa!} < \frac{1}{n^2}$. Thus, the probability that any n slots receives $> O(\frac{\ln(n)}{\ln(\ln(n))})$ keys is $< \frac{1}{n}$.

2 Linear Probing

Linear probing is a hash table strategy where each bucket holds a single value, and a hashed value will keep incrementing positions past the hashed location until an empty location is found.

For simplicity, assume a load factor $a = \frac{1}{3}$. Define a 'region of size m' as a consecutive set of m locations in the hash table. An element q hashes to region R if $h(q) \in R$, though q may not be placed in R. On expectation, a region of size 2^S should have at most $\frac{1}{3}\ 2^S$ element hash to it. A region is overloaded if at least $\frac{2}{3}\ 2^S$ elements hash to it. Using the following theorem (see https://arxiv.org/abs/1509.04549 for the proof):

Theorem: the probability a query element q ends up between 2^s and 2^{S+1} steps away from its home location is upper bounded by:

$$c * Pr[\text{region of size } 2^S \text{ centered on } h(q) \text{ is overloaded}]$$

for some fixed constant c independent of s.

Thus, we can write the expectation as:

$$\begin{aligned}\mathbb{E}[\text{Lookup Time}] &\leq O(1) \sum_{1}^{\texttt{log}(n)} 2^S * Pr[\text{q is between } 2^S \text{ and } 2^{S+1} \text{ slots away from h(q)}] \\ &\leq O(1) \sum_{1}^{\texttt{log}(n)} 2^S * Pr[\text{the region of size } 2^S \text{ centered on h(q) is overloaded}]\end{aligned}$$

Let the random variable B_S represent the number of keys that hash into the block of size 2^S centered on h(q). Assuming the hash functions are at least 2-independent, we have $\mathbb{E}[B_S] = \frac{1}{3} * 2^S$, thus:

$$\mathbb{E}[\text{Lookup Time}] \leq O(1) \sum_{1}^{\texttt{log}(n)} 2^S * Pr[B_S \geq 2 * \mathbb{E}[B_S]]$$

Using Markov's inequality, we can bound $\Pr[B_S \geq 2^*\mathbb{E}[B_S]] \leq \frac{1}{2}$, thus the expectation is bounded by $O(1)\sum_1^{\texttt{log}(n)} 2^{S-1}$, which is $O(n)$.

However, if we can bound the variance, we can use Chebyshev's.

Let the indicator variable X_i=1 if the i^{th} element maps to a block of size 2^S centered at h(q):

$$\begin{aligned}\mathbb{E}[X_i] &= \frac{2^S}{N} \\ B_S &= \sum X_i \\ \mathbb{E}[B_S] &= \mathbb{E}[\sum X_i] \\ &= \frac{1}{3}2^S\end{aligned}$$

Thus,

$$\begin{aligned}Var[B_S] &= \mathbb{E}[B_S^2] - (\mathbb{E}[B_S])^2 \\ \mathbb{E}[(\sum X_i)^2] &= \mathbb{E}[\sum X_i^2 + \sum X_i X_j] \\ &= \sum \mathbb{E}[X_i^2] + \sum \mathbb{E}[X_i X_j] \\ &= \sum \mathbb{E}[X_i] + \sum \mathbb{E}[X_i]\mathbb{E}[X_j]\end{aligned}$$

This, along with the fact $\sum \mathbb{E}[X_i][X_j] < (\mathbb{E}[B_S])^2$, gives us $\text{Var}[B_S] \leq \mathbb{E}[B_S]$. Thus, we can use Chebyshev's inequality to bound the lookup time even further:

$$
\begin{aligned}
Pr[|B_S - \mathbb{E}[B_S]| \geq \mathbb{E}[B_S]] &\leq \frac{Var[B_S]}{\mathbb{E}[B_S]^2} \\
&\leq \frac{1}{\mathbb{E}[B_S]} \\
&\leq 3 * 2^{-S} \\
O(1) \sum_{1}^{\texttt{log}(n)} 2^S * Pr[B_S \geq 2 * \mathbb{E}[B_s]] &\leq O(1) \sum_{1}^{\texttt{log}(n)} 1 \\
&\leq O(\texttt{log}(n))
\end{aligned}
$$

Thus, the expected cost of looking up an element is $O(\texttt{log}(n))$.

3 Cuckoo Hashing

Cuckoo Hashing provides worst case $O(1)$ lookup time. It works by keeping two tables, with two hash functions, one for each table. An item can be inserted into either the position in table 1 denoted by hash function 1, or table 2 by hash function 2. If both spaces are occupied, it 'evicts' one of the two elements occupying its two potential locations. This evicted element is then moved to its hashed location in the other table, potentially evicting another element. This process of swapping tables and evicting elements continues until an element is evicted and moved to a free space. This creates a potentially infinite loop for inserting, but ensures an element can be located by searching in exactly two places, guaranteeing O(1) lookup time.

In practice, cuckoo hashing is about 20-30% slower than linear probing.