Data Structures and Algorithms
Lecture Outline
March 29, 2016

Collision Resolution Using Chaining

Each location $T[i]$ in the table points to a linked list L_i that contains all elements that are mapped to location i in T. Thus L_i itself is a dictionary implemented using a linked list, but only containing elements whose keys are mapped to location i in the hash table T.

Searching for an element with key k takes time proportional to the length of the linked list $T[h(k)]$. *Insert*(T, e) procedure first checks to see if the element e with key k is there in the table and if the element is not there then it inserts the element at the end of the linked list at $T[h(k)]$. Thus inserting an element with key k takes time proprtional to the length of $T[h(k)]$. *Delete*(T, e) operation similarly takes time $O(T[h(k)])$ finds the element to be deleted and then deletes it.

Analysis

Let T be a hash table with m slots and let n be the number of elements stored in T. We define the load factor $\alpha = n/m$ as the average number of elements stored in the chain. Note that the worst case time for searching is $O(n)$ as all elements may be mapped to the same slot in T. The average case depends on how well the hash function distributes the keys of the elements to be stored in the table.

We will analyze the average time to search for an element under the *simple uniform hashing* assumption, which is that any given element is equally likely to be hashed to any of the m slots, independently of where the other elements are hashed to. It is easy to see that $\mathbf{E}[|T[j]|] = \alpha$.

We will now show that an unsuccessful search takes average-case time $O(1 + \alpha)$, under the assumption of simple uniform hashing. Let k be the key that we are searching for and that it does not exist in T. Since k is equally likely to be mapped to any of the m locations, the expected search time is the expected number of elements in $T[h(k)]$. We know that $\mathbf{E}[|T[j]|] = \alpha$ and taking into account the $O(1)$ time to compute $h(k)$ yields a search time of $Theta(1 + \alpha)$.

Now consider successful search. In this case, the probability that a list is searched is proportional to the number of elements in the list. Let $x_1, x_2, \ldots, x_n$ be the order in which the elements are inserted. Let X_j be the random variable denoting the number of elements from $\{x_1, x_2, \ldots, x_{j-1}\}$ that are mapped to the same location in T as x_j. Let X_{ij} be an indicator random variable that is 1, if x_i and x_j are mapped to the same location in the hash table, and 0, otherwise. The expected number of elements examined in a successful search is given by

$$\begin{aligned}
&\frac{\sum_{j=1}^{n} \mathbf{E}[X_j]}{n} \\
&= \frac{\sum_{j=1}^{n}(1 + \sum_{i=1}^{j-1} \mathbf{E}[X_{ij}]}{n} \\
&= \frac{n + \sum_{j=1}^{n} \sum_{i=1}^{j-1} \Pr[X_{ij} = 1]}{n} \\
&= 1 + \frac{\sum_{j=1}^{n}(j-1)/m}{n} \\
&= 1 + \frac{n+1}{2m} - \frac{1}{m} \\
&= 1 + \frac{\alpha}{2} - \frac{1}{2m} \\
&= 1 + \frac{\alpha}{2} - \frac{\alpha}{2n}
\end{aligned}$$

Counting $O(1)$ time to compute the hash function, we get the average search time for a successful search as $\Theta(1 + \alpha)$.

If $n = O(m)$ then $\alpha = O(1)$ and the average search time is a constant.

Example. If n balls are thrown independently and uniformly at random into n bins, the probability that the maximum load is more than $\frac{3 \ln n}{\ln \ln n}$ is at most $1/n$, for n sufficiently large.

Solution. Let X_j be a random variable that denotes the number of balls in bin j. Let $X = \max_j\{X_j\}$. Clearly, $\mathbf{E}[X_j] = 1$, for all j.

Consider the probability that bin j contains at least k balls. Since there are $\binom{n}{k}$ ways in which k balls can be chosen and the probability of a particular ball landing in bin j is $1/n$, we have

$$\Pr[X_j \geq k] \leq \binom{n}{k}\left(\frac{1}{n}\right)^k \leq \frac{n^k}{k!}\left(\frac{1}{n}\right)^k = \frac{1}{k!} \leq \left(\frac{e}{k}\right)^k.$$

The first inequality follows because $\Pr[X_j = k] = \binom{n}{k}\left(\frac{1}{n}\right)^k (1 - \frac{1}{n})^{n-k}$. The last term in this expression is upper bounded by 1 when we are considering $\Pr[X_j \geq k]$, since we don't care where the rest of the $n - k$ balls land. To obtain the last inequality, we use $k! \geq (k/e)^k$, which follows from Stirling's approximation.

Setting $k = 3 \ln n / \ln \ln n$ and then using the union bound, the probability that any bin

receives more than k balls is given by

$$\begin{aligned}
\Pr[X \geq k] &\leq n \left(\frac{e}{k}\right)^k \\
&\leq n \left(\frac{e \ln \ln n}{3 \ln n}\right)^{3 \ln n / \ln \ln n} \\
&\leq n \left(\frac{\ln \ln n}{\ln n}\right)^{3 \ln n / \ln \ln n} \\
&= e^{\ln n} \left(e^{\ln \ln \ln n - \ln \ln n}\right)^{3 \ln n / \ln \ln n} \\
&= e^{-2 \ln n + 3 (\ln n)(\ln \ln \ln n) / \ln \ln n} \\
&= e^{-2 \ln n + o(\ln n)} \qquad \text{(for values of } n \text{ that are sufficiently large)} \\
&\leq \frac{1}{n}
\end{aligned}$$

for sufficiently large values of n.

It can also be shown that the maximum load is $\Omega(\ln n / \ln \ln n)$.