Intro Algo Chapter 11

Mengxi Wu (mw4355@nyu.edu)

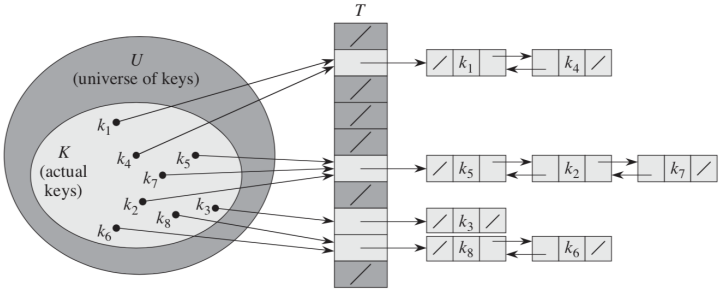
March 19, 2020

1 Hash Table

We use a hash function h to compute the slot from the key k. h maps the universe U of keys into slots of a hash table T. h: $U \to \{0, 1...m-1\}$. The size m of the hash table is much less than $|U|$. When 2 keys may hash to the same slot, we call this hash collision. $k, l \in U$ and $k \neq l$ but $h(k) = h(l)$.

Simple Uniform Hashing
Any given element is equally likely to hash into any of the m slots, independently of where any other element has hashed to. $Pr[h(k) = i] = \frac{1}{m}$, $\forall i \in \{0, 1, 2..m-1\}$.

2 Collision resolution by chaning



We place all the keys that hash to a same slot in a linked list. The operations on hash table T with chaning are shown as follow:

CHAINED-HASH-INSERT(T, x)
1 insert x at the head of list $T[h(x.key)]$

CHAINED-HASH-SEARCH(T, k)
1 search for an element with key k in list $T[h(k)]$

CHAINED-HASH-DELETE(T, x)
1 delete x from the list $T[h(x.key)]$

x is a node in the list. The worst-case running time for insertion is $O(1)$. If the list is a double linked list, the delete takes $O(1)$, since we know the *prev* and *next* attributes of x. If the list is single linked list, we need to search x to update the *next* attribute of x to its predecessor. The worst-case running time of delete will same as search. Given a hash table T with m slots that stores n elements, we define the load factor α for T as $n = m$, that is, the average number of elements stored in a chain.

The worst-case behavior of hashing with chaining is terrible: all n keys hash to the same slot, creating a list of length n. The worst-case time for searching is thus $\Theta(n)$ plus the time to compute the hash function.

Unsuccessful Search
The expected time to search unsuccessfully for a key k is the expected time to search to the end of list $T[h(k)]$. The expected number of elements hash to the same slot is equal to the expected length of a list. Thus, $E[L] = \sum_{j=0}^{n} 1 \cdot Pr[h(j) = h(k)] = \frac{n}{m} = \alpha$. The expected time is 1 (hashing step) + $E[L] = 1 + \alpha = \Theta(1 + \alpha)$.

Successful Search
Let x_i be the target element. Let L_i be the number of element searched before reaching x_i. The expected time to find x_i is average $E[L_i] + 1$. We define an indicator variable X_{ij}. $X_{ij} = 1$ if $h(k_i) = h(k_j)$. $L_i = \sum_{j=i+1}^{n} X_{ij}$. $E[L_i]$=X_{ij}· Pr$[h(k_i) = h(k_j)] = \frac{n-i}{m}$ for a particular i. x_i is the ith indserted element. To find the expected length, we need to average the length for each element.

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n} E[L_i] + 1 &= \frac{1}{n}\sum_{i=1}^{n} \frac{n-i}{m} + 1 \\
&= \frac{1}{nm}\sum_{i=1}^{n} n - \sum_{i=1}^{n} i + 1 \\
&= \frac{1}{nm}(n^2 + \frac{n(n+1}{2}) + 1 \\
&= 1 + \frac{\alpha}{2} - \frac{\alpha}{2n} \\
&< 1 + \frac{\alpha}{2}
\end{aligned}$$

In a hash table in which collisions are resolved by chaining, a successful search takes average-case time $\Theta(1 + \alpha)$, under the assumption of simple uniform hashing.

3 Collision resolution by open addressing

In open addressing, each entry in the hash table stores only one element (so, in particular, we only use it when $n < m$). If we try to insert a new element and we get collision, then we have to look for a new location to store the new element. But we have to put it somewhere where we can find it if we're searching for it. To insert it, we check a well-defined sequence of other locations in the hash table until we find one that's not full. This sequence is called a probe sequence. We will consider three different types of probe sequences.

The deletion in open addressing cannot replace the slot that contains the deleted element to be NULL, since it will cause error in search. For example, when we insert key k we find slot A is occupied, so we inset key k to the next slot B. Now we remove the element in slot A. If we replace the value in slot A to NULL, when do search key k, the search algorithm will return NULL since when we search to slot A, slot A is empty so the search will stop. The search algorithm will not visit slot B. Thus, the error occurs. Key k is in slot B but the search algorithm returns not found. We can solve this problem by marking the slot, storing in it the special value DELETED instead of NULL.

In open addressing, we also assume simple uniform hashing. the probe sequence of each key is equally likely to be any of the m! permutations of $< 0, 1...m - 1 >$. There are three commonly used techniques to compute the probe sequence: linear probing, quadratic probing, and double hashing.

Linear Probing
Let $h' : U \to \{0, 1, ...m - 1\}$

$h(k,i) = (h'(k) + i) \bmod m$ for $i = 0, 1, 2...m-1$

Quadratic Probing
$h(k,i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$ for $i = 0, 1, 2...m-1$
As in linear probing, the initial probe determines the entire sequence, and so only m distinct probe sequences are used.

Double Hashing
$h(k,i) = h_1(k) + ih_2(k) \bmod m$

We assume uniform hashing. The probe sequence $< h(k,0), h(k,1), ...h(k,m-1) >$ is equally likely to be any permutation of $< 0, 1, 2...m-1 >$. A given key has a unique fixed probe sequence associated with it.

Unsuccessful Search
Given an open-address hash table with load factor $\alpha = \frac{n}{m} < 1$, the expected number of probes in an unsuccessful search is at most $\frac{1}{1-\alpha}$, assuming uniform hashing.

In an unsuccessful search, the last slot we search is empty and the other slots we search are occupied by irrelevant keys. Let X =the number of probes made in an unsuccessful search. Let A_i =be the event that an ith probe occurs and it is to an occupied slot.

$$Pr[X \geq i] = Pr[A_1 \cap A_2 \cap A_3 ... \cap A_{i-1}]$$

It is not equal to $Pr[X = i]$, since $Pr[X = i] = Pr[A_1 \cap A_2 \cap A_3 ... \cap A_{i-1}] \cdot Pr[\neg A_i]$.

$Pr[A_1 \cap A_2 \cap A_3 ... \cap A_{i-1}] = Pr[A_{i-1}|A_1 \cap A_2 ... \cap A_{i-2}] \cdot Pr[A_{i-2}|A_1 \cap A_2 ... \cap A_{i-3}] ... \cdot Pr[A_3|A_1 \cap A_2] \cdot Pr[A_2|A_1] \cdot Pr[A_1]$

$Pr[A_1] = \frac{n}{m}$. For $i > 1$, the probability that there is a i-1 th probe and it is to an occupied slot, given that the first i-2 probes were to occupied slots $Pr[A_{i-1}|A_1 \cap A_2 ... \cap A_{i-2}] = \frac{n-(i-2)}{m-(i-2)}$. This probability follows because we would be finding one of the remaining n-(i-2) elements in one of the m-(i-2) unexamined slots, and by the assumption of uniform hashing, the probability is the ratio of these quantities.

$$Pr[X \geq i] = \frac{n}{m} \cdot \frac{n-1}{m-1} \cdot \frac{n-2}{m-2} ... \cdot \frac{n-i+2}{m-i+2} \leq (\frac{n}{m})^{i-1} = \alpha^{i-1}$$

$$\begin{aligned}
E[X] &= \sum_{i=1}^{\infty} Pr[X \geq i] \\
&= \sum_{i=1}^{\infty} i[Pr[X \geq i] - Pr[X \geq i+1]] \\
&= \sum_{i=1}^{\infty} iPr[X \geq i] - \sum_{i=1}^{\infty} (i-1)Pr[X \geq i] \\
&= \sum_{i=1}^{\infty} Pr[X \geq i] \\
&\leq \sum_{i=1}^{\infty} \alpha^{i-1} \\
&\leq \frac{1}{1-\alpha}
\end{aligned}$$

Successful Search
Given an open-address hash table with load factor $\alpha < 1$, the expected number of probes in a successful

search is at most $\frac{1}{\alpha}\ln\frac{1}{1-\alpha}$ assuming uniform hashing and each key in the table is equally likely to be searched for.

Expected number of probes for insertion is same as expected number of probes for an unsuccessful search. if x is the target element and it is (i+1)th inserted item. The expected number of probes to reach x is at most $\frac{1}{1-\frac{i}{m}}$. Thus, we average over all n keys in the hash table gives us the expected number of probes in a successful search.

$$\begin{aligned}
\frac{1}{n}\sum_{i=0}^{n-1}\frac{m}{m-i} &= \frac{m}{n}\sum_{i=0}^{n-1}\frac{1}{m-i} \\
&= \frac{1}{\alpha}\sum_{k=m-n+1}^{m}\frac{1}{k} \\
&\leq \frac{1}{\alpha}\int_{m-n}^{m}\frac{1}{x}dx \\
&= \frac{1}{\alpha}\ln\frac{m}{m-n} \\
&= \frac{1}{\alpha}\ln\frac{1}{1-\alpha}
\end{aligned}$$