Jade Yu Cheng
ICS 311
Homework 11
Oct 9, 2008

Question for lecture 14

Problem 12-3 on p. 250

Quadratic probing

Suppose that we are given a key k to search for in a hash table with positions $0, 1, \ldots, m-1$, suppose that we have a hash function h mapping the key space into the set $\{0,1, \ldots, m-1\}$. The search scheme is as follows.

1. Compute the value $i <- h(k)$, and set $j <- 0$.

2. Probe in position i for the desired key k. If you find it, or if this position is empty, terminate the search.

3. Set $j <- (j+1) \bmod m$ and $i <- (i+j) \bmod m$, and return to step 2.

Assume that m is a power of 2.

a. Show that this scheme is an instance of the general "quadratic probing" scheme by exhibiting the appropriate constants c_1 and c_2 for equation (11.5).

$$h(k,i) = (h'(k) + c_1 i + c_2 i^2) \% m$$

Answer: According to the description, for a given hash table with m slots, a given auxiliary hash function $h(k)$, and a variable $i = 0, 1, \ldots, m-1$, we could easily calculate the initial steps for the hashing.

Step 1. The initial position probed would be:

$$i_0 = T[h'(k)]$$

$$j_0 = 0$$

Step 2. The second position probed would be:

$$i_1 = (T[h'(k)] + j_1)\%m = (i_0 + 1)\%m$$

$$j_1 = j_0 + 1 = 0 + 1 = 1$$

Then we update the value of i from i_0 to i_1

Step 3. The third position probed would be:

$$i\_2 = (i\_2)\,\%\,m = (i\_1 + j\_2)\,\%\,m = (i\_0 + j\_1 + j\_1 + 1)\,\%\,m$$
$$= (i\_0 + 1 + j\_0 + 2)\,\%\,m = (i\_0 + 1 + 2)\,\%\,m$$

j_0 = 0

Then we update the value of i from i_1 to i_2

Step 4. The fourth position probed would be:

$$(i\_3)\,\%\,m = (i\_2 + j\_3)\,\%\,m = (i\_2 + j\_2 + 1)\,\%\,m$$
$$= (i\_1 + j\_2 + j\_1 + 2)\,\%\,m = (i\_0 + 1 + 2 + 3)\,\%\,m$$

$$j_0 = 0, j_1 = 1, and\ j_2 = 2$$

Then we update the value of i from i_2 to i_3

Step n: Therefore, we get the general formula to calculate the position to probe at the nth operation, while $i = 0, 1, \ldots, m$, m is the size the given hash table. The size of the hash table is considered as one of the constraints. The formula is as below:

$$h(k,i) = \left(T[h'(k)] + 1 + 2 + 3 + \ldots + i\right)\%m$$
$$= \left(T[h'(k)] + \frac{i\cdot(i+1)}{2}\right)\%m$$

Compare this formula to the universal equation of quadratic hashing:

$$h(k,i) = \left(T[h'(k)] + \frac{i\cdot(i+1)}{2}\right)\%m$$
$$= \left(T[h'(k)] + \frac{1}{2}i + \frac{1}{2}i^2\right)\%m$$
$$= \left(T[h'(k)] + c_1 i + c_2 i^2\right)\%m$$
$$= \left(h'(k) + c_1 i + c_2 i^2\right)\%m$$

Where $T[h'(k)]$ is the initial value of the probe position, and $c_1 = \frac{1}{2}$, $c_2 = \frac{1}{2}$. Therefore we showed that this scheme is an instance of the general "quadratic probing" scheme. This is also a commonly used way to select the constraints to create a quadratic hash function.

b. Prove that this algorithm examines every table position in the worst case.

Answer: In order to prove this hashing algorithm examines every position in a table before any repeating, we could consider using induction.

The first step is to prove a base case, which would be obvious. If I have a hash table with 4 positions, so $m = 4$, and let the initial probe position calculated using the auxiliary hash function $h(k)$ be position 2, we would get the first probing position as $2 \% m = 2$. We would get the next probing position $(2 + 1) \% m = 3 \% m = 3$. The following probing position is $(2 + 1 + 2) \% m = 5 \% m = 1$. The fourth probing position is $(2 + 1 + 2 + 3) \% m = 8 \% m = 0$. Assuming we didn't find the element we are looking for and the positions are all occupied during the probing process, we observe the fact that after the first round of probing, every slot of this hashing table with 4 positions would be visited. If none of these positions contain the element we try to look for, then after the first round of probing, we can be sure that the element is not contained in this particular hash table at all. Therefore no repeating work is necessary.

The second step is to make to assumption that if we have a hash table with a size of m, we assume the algorithm meet the requirement of visiting every single slot of the table before repeating. The size of the table m is a number of the power of 2, we assign a constant k to this equation: $m = 2^k$.

We need to prove the case when k' = k + 1, the assumption condition still holds. That is to say, if we increase the size of the table by doubling the number m, which is also done by increase k by 1, that our hashing algorithm will still provide a hashing strategy that visit all slots before repeating.

To simplify the equation, we ignore the constant $T[h'(k)]$. So we have $\left(\frac{1}{2}i + \frac{1}{2}i^2\right)\%m$, where i = 0, 1, 2, ... m, and assume it go through the list exactly once before repeating. We need to prove $\left(\frac{1}{2}i + \frac{1}{2}i^2\right)\%2 \cdot m$, where i = 0, 1, 2, ... 2m.

We consider the first half of the new hash table first. If $\left(\frac{1}{2}i+\frac{1}{2}i^2\right)\%m$ cover all where $i = 0, 1, 2, \ldots m$ the spaces of hash table with size m, then the first half of the new hash table will be covered exactly the same way. Because by mod a number double the size wouldn't change the mod calculated value.

We consider the second half of the new hash table. We can increase i and m by the same amount at the same time. Then we get $\left(\frac{1}{2}i+\frac{1}{2}i^2\right)\%2\cdot m$, where $i = 1 + (m-1), 2 + (m-1), \ldots m + (m-1)$, which is also: $i = m, m+1, \ldots 2m-1$. This part with cover the second half of the hash table exactly the same way as the hash table with a size m.

We put it in an example: If we have has table size 4, and we increase it to 8. Let assume the initial probing position is 2, as we show in the previous question, the order of slot visiting would be $2, 3, 1, 0$. They are calculated from $2\%4, 3\%4, 5\%4, 8\%4$.

In the first case of our proof, the positions would be $2\%8, 3\%8, 5\%8$, and $8\%8$, so we get a sequence of visiting to positions: $2, 3, 5, 0$. It covers the first half of the new hash table.

Then we consider the second case in the proof. We could get a sequence of positions: $6\%8, 7\%8, 9\%8, 12\%8$. From this several probing, we get to visit position $6, 7, 1$, and 4.

Therefore, combine the positions visited in these two cases, we get $0, 1, 2, 3, 4\ 5, 6, 7$. This is the full contents of the new hash table with a doubled table size, 8. So, up to now, we proved the base case, and the induction step. We can safely claim that this hashing function promise a visit to every slot before repeating. This is an essential characteristic of hashing functions.