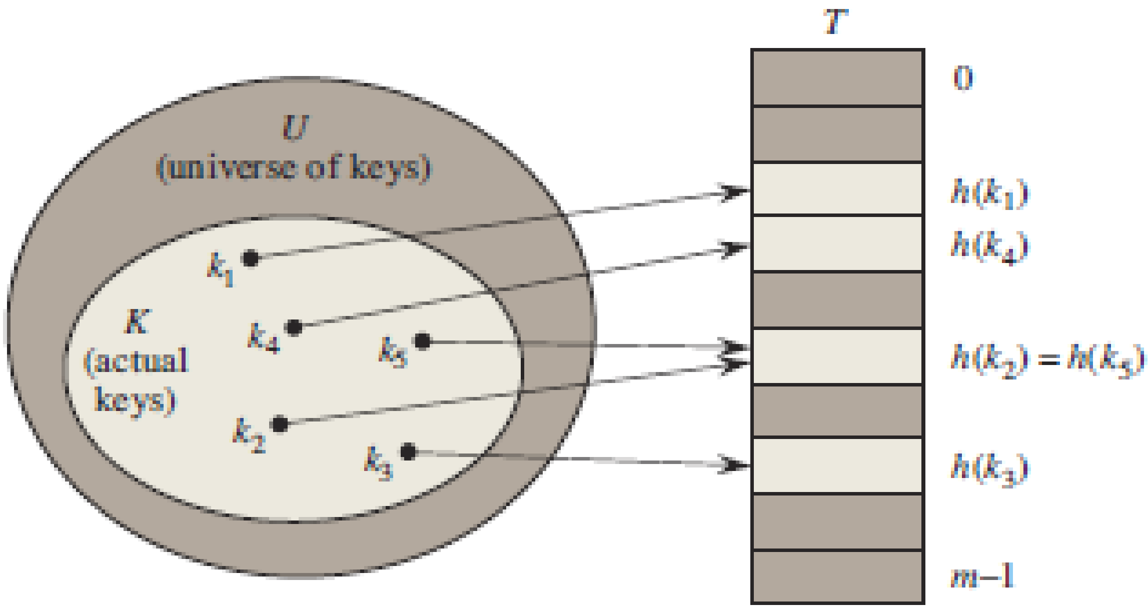
计算机问题求解 – 论题2-13
- Hashing方法

2014年06月10日

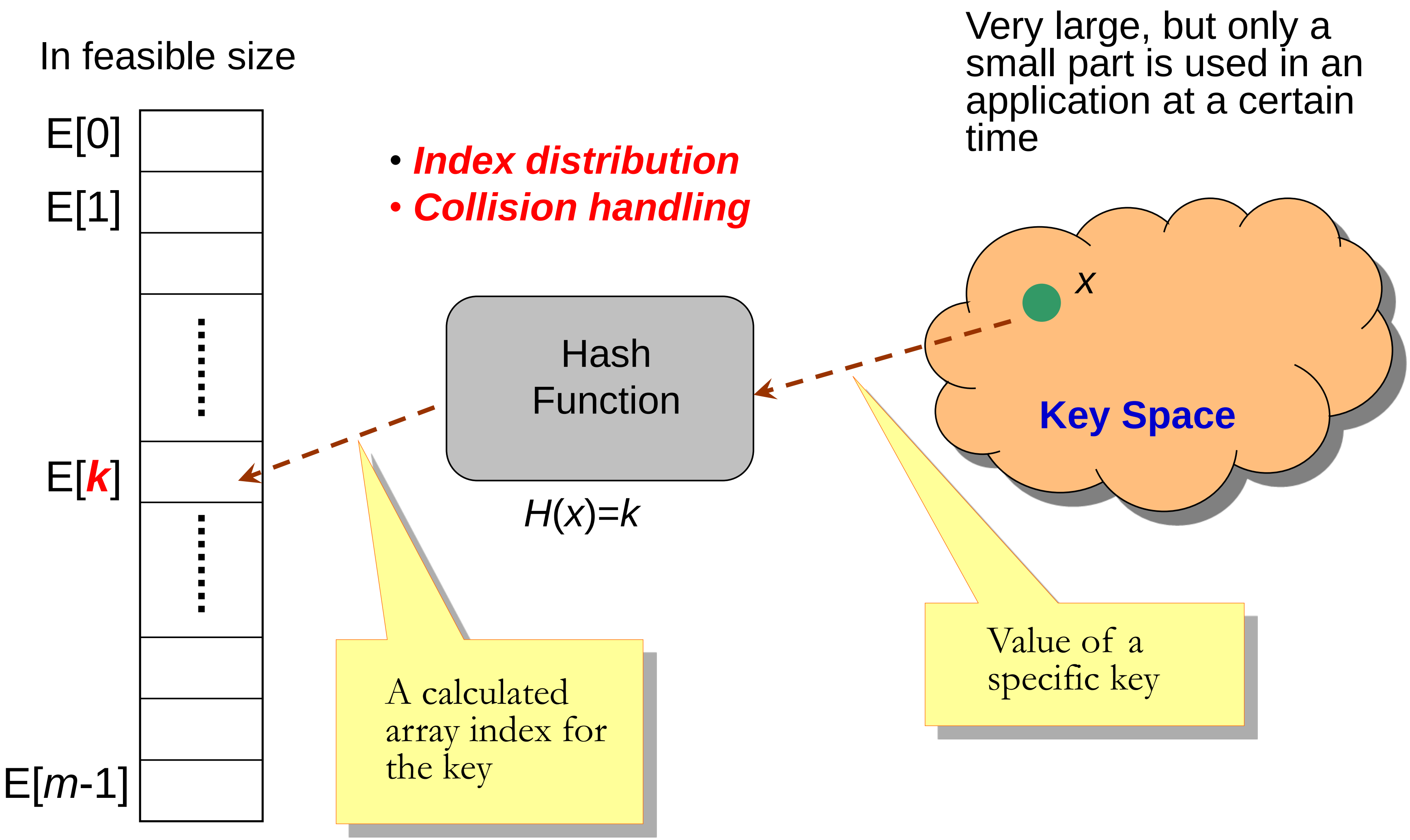
Hashing



问题1:

所谓"Hashing"方法是用来解决什么问题的？

Hashing: the Idea



问题2：

Collision是什么意思？它是如何产生的？

问题3:

假设分配的存储区为*k*个单元，插入*n*个键值。对于某个特定位置，落到该位置的对象的期望值是多少？为什么？

顺便问一下，只插入2个键值，发生碰撞的概率是多大？

模型：将插入n个对象看作n个独立试验的序列。每个试验的结果是$\{1,2,\ldots,k\}$中的一个值。

假设：每个实验的结果是任意一个允许值的概率是一样的。
(uniformly distributed)

In hashing n items into a hash table of size k, the expected number of items that hash to any one location is n/k.

α: loading factor(负载因子)

现在考虑特定单元为空的概率

- 同样的independent trials process，可以根据需要指定不同的outcomes:
  - k个不同的outcomes: 单元1,2,…,k
  - 2个outcomes: 单元i, 非单元i

问题4:

插入n个对象后，单元i仍然是空的，概率是多少？

问题4':

插入n个对象后,空单元的期望是多少？为什么？

$$(1-1/k)^n$$

一个概率悖论

假设有n个存储单元，在插入n个对象后

- 第i个单元已放入对象数期望值是：

$$\frac{n}{n}=1$$ 换句话说，没有空的单元（？）

- 整个存储区内空单元的期望数是：

$$n\left(1-\frac{1}{n}\right)^n=\frac{n}{e}\approx 0.368n$$

问题5：

你能解释这个“悖论”吗？

冲突: 可能性有多大?

在k个单元的存储区内插入n个对象：

$$E(\text{collisions}) = n - E(\text{occupied locations})$$
$$= n - k + E(\text{empty locations}).$$

In hashing n items into a hash table with k locations, the expected number of collisions is $n - k + k(1 - 1/k)^n$.

找一点感觉：

假如在100个单元的存储区内插入100个对象，
发生的碰撞数的期望值就是大约37次。

没有空单元：需要插入多少对象？

先考虑一个“子问题”：使得被占单元数从达到 i-1 增加到达到 i, 需要插入多少对象（期望）？

$$E(X_1) = 1, E(X_2) = k/(k-1), \ldots\ldots$$

In general, we have that X_i counts the number of trials until success in an independent trials process with probability of success $(k-i+1)/k$, and thus, the expected number of steps until the first success is $k/(k-i+1)$, which is the expected value of X_i.

$$E(X) = \sum_{j=1}^{k} E(X_j) = \sum_{j=1}^{k} \frac{k}{k-j+1} = k \sum_{j=1}^{k} \frac{1}{k-j+1} = k \sum_{k-j+1=1}^{k} \frac{1}{k-j+1} = k \sum_{i=1}^{k} \frac{1}{i}$$

$$\boxed{\Theta(k \log k)}$$

给你一点感觉：

当 k=10000, 这个值大约是98000。

问题6：

上面的讨论与实际的Hashing有什么差别？

选择好的Hashing函数很重要！

两种设计Hashing函数的简单方法

In the *division method* for creating hash functions, we map a key k into one of m slots by taking the remainder of k divided by m. That is, the hash function is

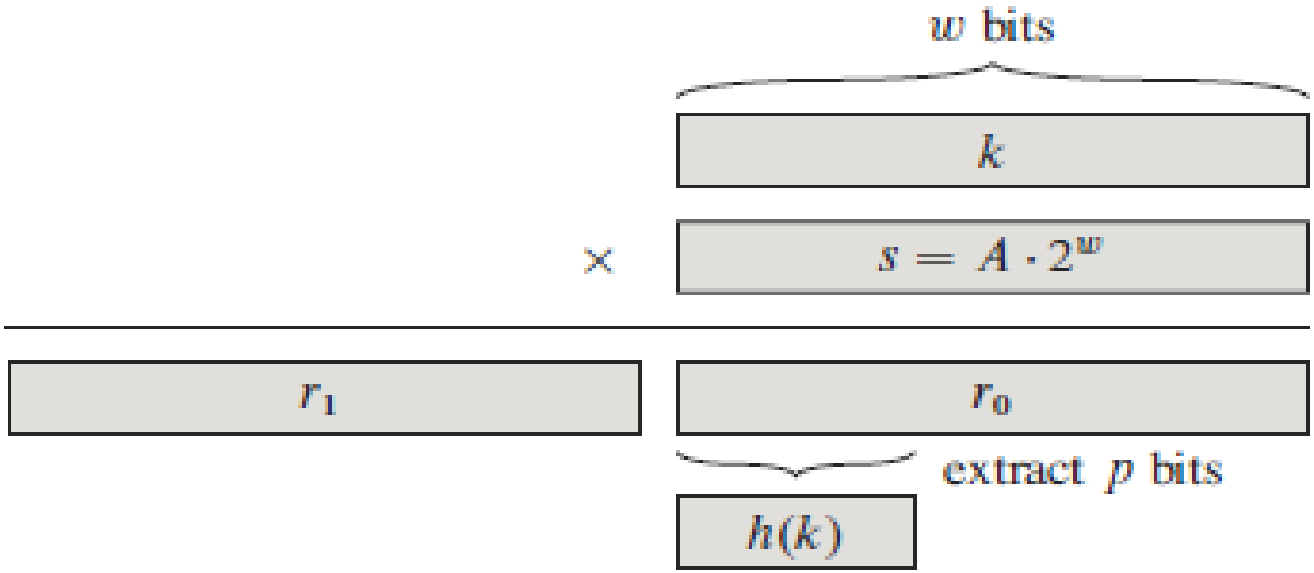
$$h(k) = k \bmod m .$$

The *multiplication method* for creating hash functions operates in two steps. First, we multiply the key k by a constant A in the range $0 < A < 1$ and extract the fractional part of kA. Then, we multiply this value by m and take the floor of the result. In short, the hash function is

$$h(k) = \lfloor m\,(kA \bmod 1) \rfloor ,$$

问题7：

为什么在除法方法中，应该避免m是2的整次幂，而在乘法方法中却往往选择m的值为2的整次幂？



问题8：

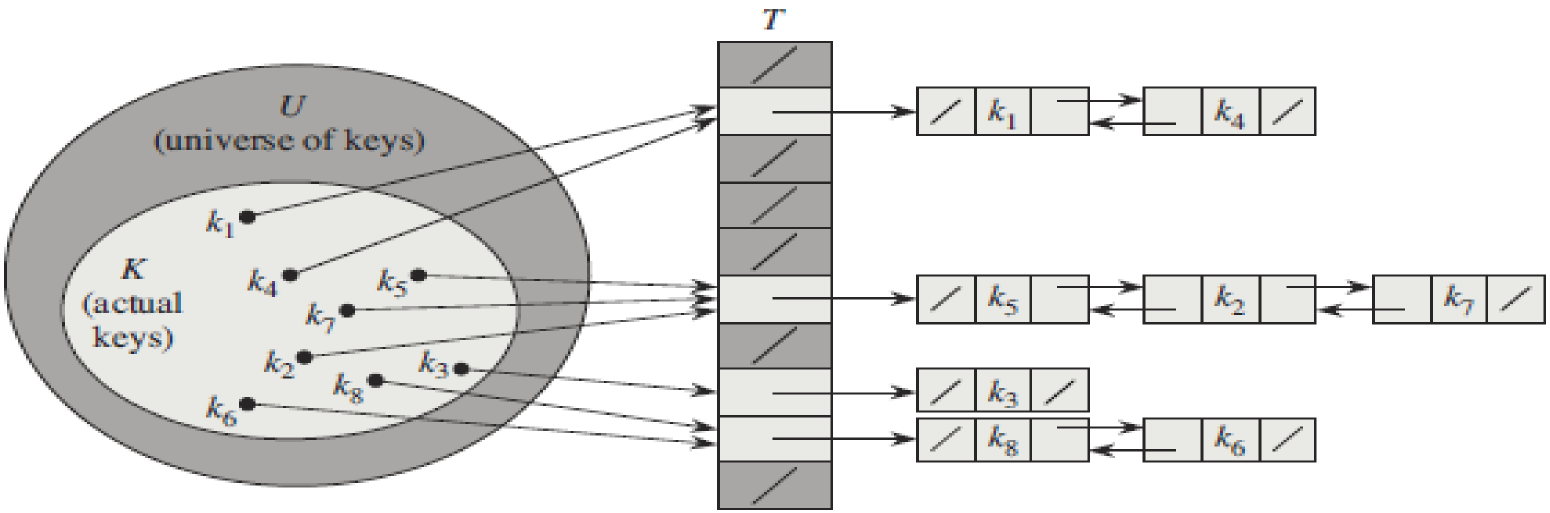
你能解释一下这个图吗？

问题9：

“Hashing by division and Hashing by multiplication are heuristic in nature.”

这是什么意思？

Collision Resolution by Chaining



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 $T[h(x.key)]$

Closed addressing

问题10:

采用Hashing by Chaining,不成功搜索的平均代价是多少?为什么?

Hashing by Chaining: 不成功搜索

- Assumption: simple uniform hashing:
  - for j=0,1,2,...,k-1, the average length of the list at $E[j]$ is $n/k=\alpha$.
- The average cost of an unsuccessful search:
  - Any key that is not in the table is equally likely to hash to any of the k addresses. The average cost to determine that the key is not in the list $E[h(k)]$ is the cost to search to the end of the list, which is α. So, the total cost is $\Theta(1+\alpha)$.

问题11:

采用Hashing by Chaining计算成功搜索与不成功搜索的代价有什么不同?

Hashing by Chaining: 成功搜索

- For successful search: (assuming that x_i is the ith element inserted into the table, i=1,2,...,n)
  - For each i, the probability of that x_i is searched is $1/n$.
  - For a specific x_i, the number of elements examined in a successful search is t+1, where t is the number of elements inserted into the same list as x_i, after x_i has been inserted. And for any j, the probability of that x_j is inserted into the same list of x_i is $1/m$. So, the cost is:

Cost for computing hashing

$$\frac{1}{n}\sum_{i=1}^{n}\left(1+\sum_{j=i+1}^{n}\frac{1}{m}\right)$$

Expected number of elements in front of the searched one in the same linked list.

Hashing by Chaining: 成功搜索

- The average cost of a successful search:
  - Define α=*n*/*k* as ***load factor***,

$$\text{The average cost of a successful search is:}$$

$$\frac{1}{n}\sum_{i=1}^{n}\left(1+\sum_{j=i+1}^{n}\frac{1}{m}\right)=1+\frac{1}{nm}\sum_{i=1}^{n}(n-i)=1+\frac{1}{nm}\sum_{i=1}^{n-1}i$$

$$=1+\frac{n-1}{2m}=1+\frac{\alpha}{2}-\frac{\alpha}{2n}=\Theta(1+\alpha)$$

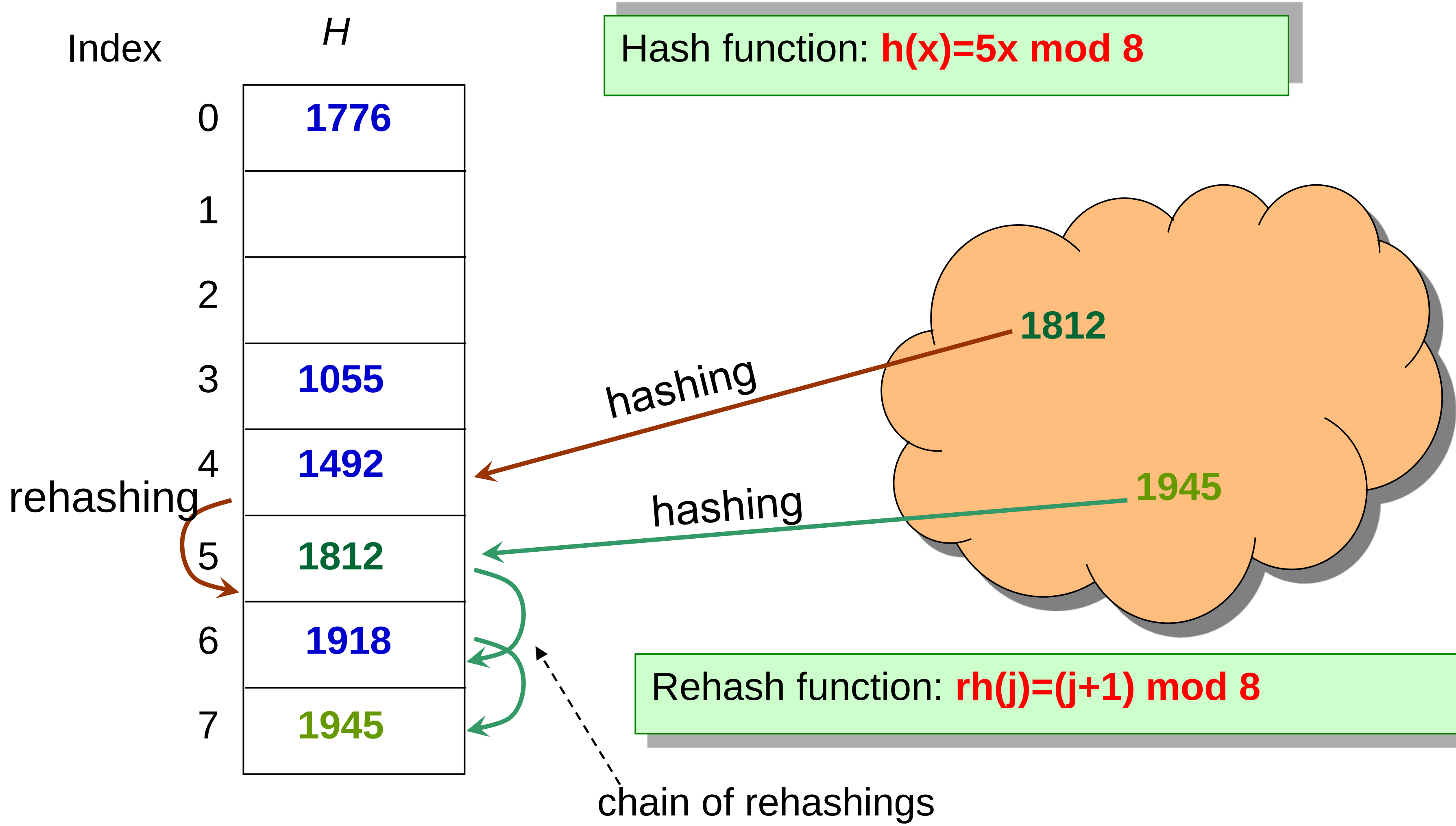
Cost for computing hashing

Number of elements in front of the searched one in the same linked list.

另一种冲突处理方法：Open Addressing

- All elements are stored in the hash table, no linked list is used. So, α, the load factor, can not be larger than 1.
- Collision is settled by "rehashing": a function is used to get a new hashing address for each collided address, i.e. the hash table slots are *probed* successively, until a valid location is found.
- The probing sequence can be seen as a permutation of (0,1,2,..., *m*-1)
  - <h(k,0), h(k,1),…, h(k,m-1)>

Linear Probing: an Example



Hash function: **h(x)=5x mod 8**

Rehash function: **rh(j)=(j+1) mod 8**

问题12：

Open Addressing方法为什么不适合用于支持删除操作的结构？

Commonly Used Probing

Linear probing:

Given an ordinary hash function h', which is called an auxiliary hash function, the hash function is: **(clustering may occur)**

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

Quadratic Probing:

Given auxiliary function h' and nonzero auxiliary constant c_1 and c_2, the hash function is: **(secondary clustering may occur)**

$$h(k,i) = (h'(k)+c_1 i+ c_2 i^2) \bmod m \quad (i=0,1,...,m-1)$$

Double hashing:

Given auxiliary functions h_1 and h_2, the hash function is:

$$h(k,i) = (h_1(k)+ ih_2(k)) \bmod m \quad (i=0,1,...,m-1)$$

问题13：

一般如何判断一种probing方法的好坏？

Equally Likely Permutations

- Assumption: each key is equally likely to have any of the m! permutations of (1,2...,m-1) as its probe sequence.

- Note: both linear and quadratic probing have only m distinct probe sequence, as determined by the first probe.

Analysis for Open Address Hash

- Assuming uniform hashing, what is the average number of probes in an unsuccessful search?

Let us define the random variable X to be the number of probes made in an unsuccessful search

When a random variable X takes on values from the set of natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$, we have a nice formula for its expectation:

$$\begin{aligned} \mathrm{E}[X] &= \sum_{i=0}^{\infty} i \cdot \Pr\{X = i\} \\ &= \sum_{i=0}^{\infty} i(\Pr\{X \geq i\} - \Pr\{X \geq i+1\}) \\ &= \sum_{i=1}^{\infty} \Pr\{X \geq i\} , \end{aligned} \tag{C.25}$$

Analysis for Open Address Hash

- Assuming uniform hashing, the average number of probes in an unsuccessful search is at most 1/(1-α) (α=*n*/*m*<1)

let us also define the event A_i, for $i = 1, 2, \ldots$, to be the event that an ith probe occurs and it is to an occupied slot. Then the event $\{X \geq i\}$ is the intersection of events $A_1 \cap A_2 \cap \cdots \cap A_{i-1}$.

$$
\begin{aligned}
\Pr\{X \geq i\} &= \\
&\Pr\{A_1 \cap A_2 \cap \cdots \cap A_{i-1}\} = \Pr\{A_1\} \cdot \Pr\{A_2 \mid A_1\} \cdot \Pr\{A_3 \mid A_1 \cap A_2\} \cdots \\
&\qquad \Pr\{A_{i-1} \mid A_1 \cap A_2 \cap \cdots \cap A_{i-2}\} . \\
&= \frac{n}{m} \cdot \frac{n-1}{m-1} \cdot \frac{n-2}{m-2} \cdots \frac{n-i+2}{m-i+2} \\
&\leq \left(\frac{n}{m}\right)^{i-1} \\
&= \alpha^{i-1} .
\end{aligned}
$$

$$
\mathrm{E}[X] = \sum_{i=1}^{\infty} \Pr\{X \geq i\} \leq \sum_{i=1}^{\infty} \alpha^{i-1} = \sum_{i=0}^{\infty} \alpha^{i} = \frac{1}{1-\alpha} .
$$

问题14:

采用Open Addressing, 插入一个对象的代价是多少?

- Assuming uniform hashing, the average cost of probes in an successful search is at most $\frac{1}{\alpha}\ln\frac{1}{1-\alpha}$ ($\alpha=n/m<1$)

To search for the $(i+1)$th inserted element in the table, the cost is the same as the cost for inserting it when there are just i elements in the table. At that time, $\alpha = i/m$, so, the cost is $1/(1-i/m) = m/(m-i)$

So, the cost is :

$$\frac{1}{n}\sum_{i=0}^{n-1}\frac{m}{m-i} = \frac{m}{n}\sum_{i=0}^{n-1}\frac{1}{m-i} = \frac{1}{\alpha}\sum_{i=m-n+1}^{m}\frac{1}{i} \le \frac{1}{\alpha}\int_{m-n}^{m}\frac{dx}{x} = \frac{1}{\alpha}\ln\frac{m}{m-n}$$

$$= \frac{1}{\alpha}\ln\frac{1}{1-\alpha}$$

For your reference:

Half full: 1.387; 90% full: 2.559

课外作业

- CS pp.321-: prob.8, 11, 14,
- TC pp.261-: ex.11.2-3, 11-2.5, 11-2.6
- TC pp.268-: ex.11-3.3, 11-3.4
- TC pp.277-: ex. 11-4.2, 11-4.3
- TC pp.282-: prob.11.1, 11.2