6.006 Introduction to Algorithms
Spring 2008

Lecture 7: Hashing III: Open Addressing

Lecture Overview

- Open Addressing, Probing Strategies
- Uniform Hashing, Analysis
- Advanced Hashing

Readings

CLRS Chapter 11.4 (and 11.3.3 and 11.5 if interested)

Open Addressing

Another approach to collisions

- no linked lists
- all items stored in table (see Fig. 1)

item_2
item_1
item_3

Figure 1: Open Addressing Table

- one item per slot $\implies$ $m \geq n$
- hash function specifies <u>order</u> of slots to probe (try) for a key, not just one slot: (see Fig. 2)

Insert(k,v)

```
for i in xrange(m):
    if T[h(k, i)] is None:        ♯ empty slot
        T[h(k, i)] = (k, v)       ♯ store item
        return
raise 'full'
```

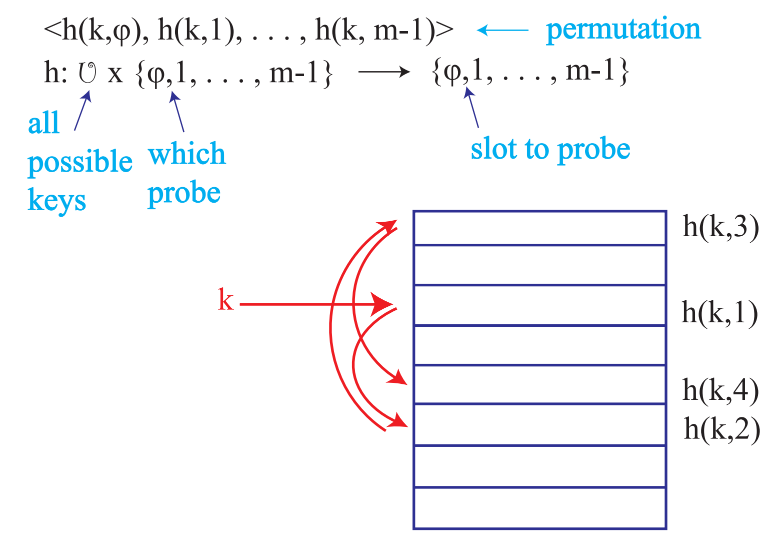

Figure 2: Order of Probes

Example: Insert $k = 496$

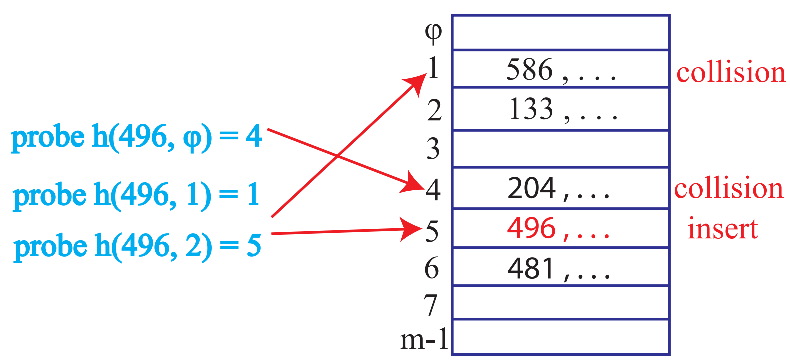

Figure 3: Insert Example

Search(k)

```
for i in xrange(m):
    if   T[h(k,i)] is None:        ♯ empty slot?
         return None               ♯ end of "chain"
    elif T[h(k,i)][φ] == k:        ♯ matching key
         return T[h(k,i)]          ♯ return item
return None                        ♯ exhausted table
```

Delete(k)

- can't just set $T[h(k,i)] = \text{None}$
- *example*: delete(586) $\Longrightarrow$ search(496) fails
- replace item with DeleteMe, which Insert treats as None but Search doesn't

Probing Strategies

Linear Probing

$h(k,i) = (\underline{h'(k)} + i) \bmod m$ where $h'(k)$ is ordinary hash function

- like street parking
- problem: *clustering* as consecutive group of filled slots grows, gets *more* likely to grow (see Fig. 4)

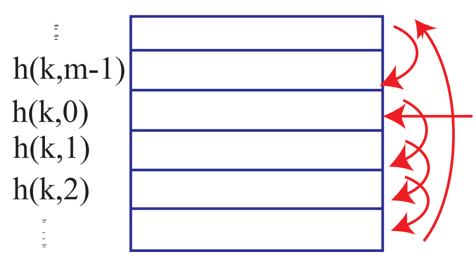

Figure 4: Primary Clustering

- for $0.01 < \alpha < 0.99$ say, clusters of $\Theta(\lg n)$. These clusters are known
- for $\alpha = 1$, clusters of $\Theta(\sqrt{n})$ These clusters are known

Double Hashing

$h(k,i) = (h_1(k) + i.\ h_2(k)) \bmod m$ where $h_1(k)$ and $h_2(k)$ are two ordinary hash functions.

- actually hit all slots (permutation) if $h_2(k)$ is relatively prime to m
- e.g. $m = 2^r$, make $h_2(k)$ always odd

Uniform Hashing Assumption

Each key is equally likely to have any one of the $m!$ permutations as its probe sequence

- not really true
- but double hashing can come close

Analysis

Open addressing for n items in table of size m has expected cost of $\leq \frac{1}{1-\alpha}$ per operation, where $\alpha = n/m(< 1)$ assuming uniform hashing

Example: $\alpha = 90\% \implies 10$ expected probes

Proof:

Always make a first probe.
With probability n/m, first slot occupied.
In worst case (e.g. key not in table), go to next.
With probability $\frac{n-1}{m-1}$, second slot occupied.
Then, with probability $\frac{n-2}{m-2}$, third slot full.
Etc. (n possibilities)

$$\begin{aligned}
\text{So expected cost} \quad &= \quad 1 + \frac{n}{m}(1 + \frac{n-1}{m-1}(1 + \frac{n-2}{m-2}(\cdots) \\
\text{Now } \frac{n-1}{m-1} \leq \frac{n}{m} \quad &= \quad \alpha \text{ for } i = \phi, \cdots, n(\leq m) \\
\text{So expected cost} \quad &\leq \quad 1 + \alpha(1 + \alpha(1 + \alpha(\cdots))) \\
&= \quad 1 + \alpha + \alpha^2 + \alpha^3 + \cdots \\
&= \quad \frac{1}{1-\alpha}
\end{aligned}$$

Open Addressing vs. Chaining

Open Addressing: better cache performance and rarely allocates memory

Chaining: less sensitive to hash functions and α

Advanced Hashing

Universal Hashing

Instead of defining one hash function, define a whole family and select one at random

- e.g. multiplication method with *random a*
- can prove Pr (over random h) $\{h(x) = h(y)\} = \frac{1}{m}$ for *every (i.e. not random)* $x \neq y$
- $\Longrightarrow$ $O(1)$ expected time per operation without assuming simple uniform hashing! CLRS 11.3.3

Perfect Hashing

Guarantee $O(1)$ worst-case search

- idea: if $m = n^2$ then E[$\sharp$ collisions] $\approx \frac{1}{2}$

  $\Longrightarrow$ get ϕ after $O(1)$ tries ... but $O(n^2)$ space
- use this structure for storing chains

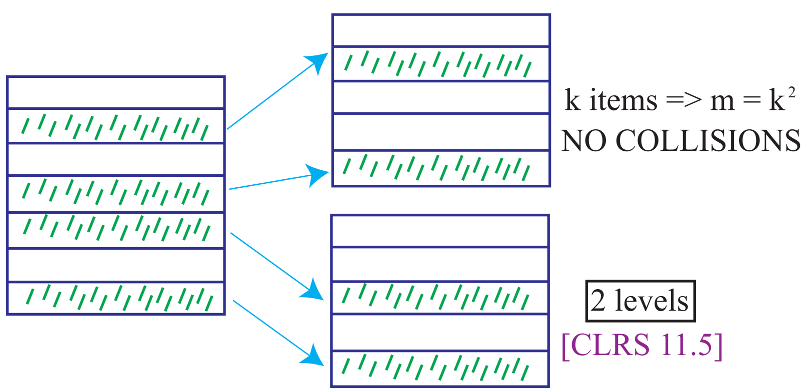

Figure 5: Two-level Hash Table

- can prove $O(n)$ expected total space!
- if ever fails, rebuild from scratch