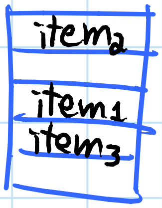
6.006 | Lecture 7 | Feb. 28, 2008

Outline: Hashing III
- open addressing, probing strategies
- uniform hashing, analysis
- advanced hashing

Reading: CLRS 11.4
(and 11.3.3 & 11.5 if interested)

Open addressing: another approach to collisions

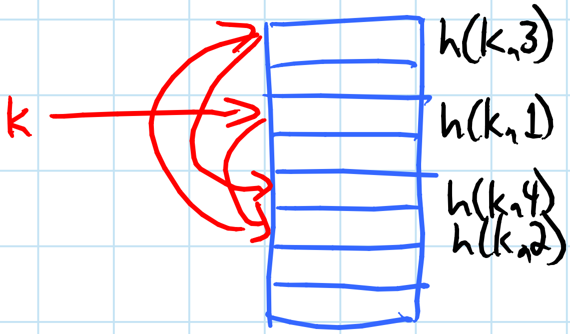


- no linked lists
- all items stored in table
- one item per slot $\Rightarrow m \geq n$

- hash function specifies order of slots to probe (try) for a key, not just one slot:

$$\langle h(k,0),\ h(k,1),\ \ldots,\ h(k,m-1)\rangle \leftarrow \text{permutation}$$

$$h : \underset{\text{all possible keys}}{\mathcal{U}} \times \underset{\text{which probe}}{\{0,1,\ldots,m-1\}} \rightarrow \underset{\text{slot to probe}}{\{0,1,\ldots,m-1\}}$$

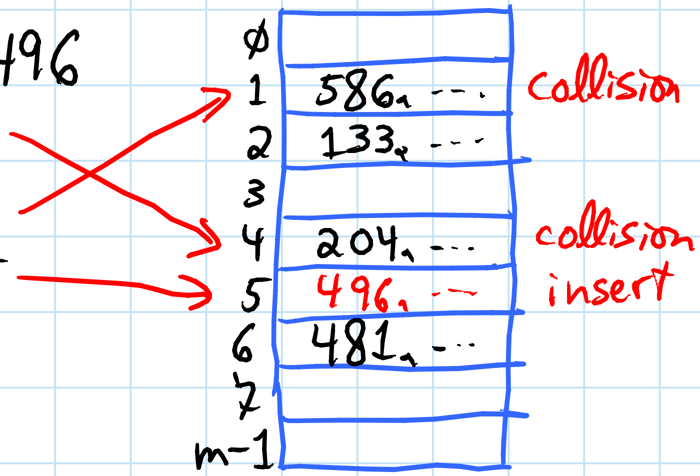


Insert(k,v):

```
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'
```

Example: insert k=496

- probe $h(496,0)=4$
- probe $h(496,1)=1$
- probe $h(496,2)=5$



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)][0] == k:      # matching key?
        return T[h(k,i)]         # return item
return None                      # exhausted table
```

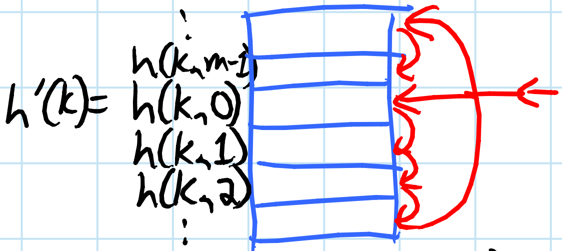
Delete(k):

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

Probing strategies:

Linear probing: $h(k,i) = (\underbrace{h'(k)}_{\text{ordinary hash function}} + i) \bmod m$

- like street parking
- problem: clustering
- as consecutive group of filled slots grows, gets more likely to grow



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

known

Double hashing: $h(k,i) = (\underbrace{h_1(k)}_{} + i \cdot \underbrace{h_2(k)}_{}) \bmod m$

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: $\alpha = 90\% \Rightarrow 10$ expected probes

Proof: Always make a first probe.
With probability $\frac{n}{m}$, 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)

So expected cost $= 1 + \frac{n}{m}\left(1 + \frac{n-1}{m-1}\left(1 + \frac{n-2}{m-2}\left(\cdots\right.\right.\right.$
Now: $\frac{n-i}{m-i} \leq \frac{n}{m} = \alpha$ for $i = 0, 1, \ldots, n \, (\leq m)$
So expected cost $\leq 1 + \alpha(1 + \alpha(1 + \alpha(\cdots$

$$= 1 + \alpha + \alpha^2 + \alpha^3 + \cdots$$

$$= \frac{1}{1-\alpha}. \qquad \square$$

Open addressing vs.	chaining
- better cache performance	- less sensitive to hash functions & α
- rarely allocate memory	

Advanced hashing:

Universal hashing: instead of defining one hash function, define a whole family & select one at random

- e.g. multiplication method with random a
- can prove $\Pr\{h(x)=h(y)\} = \frac{1}{m}$ for every $x \neq y$

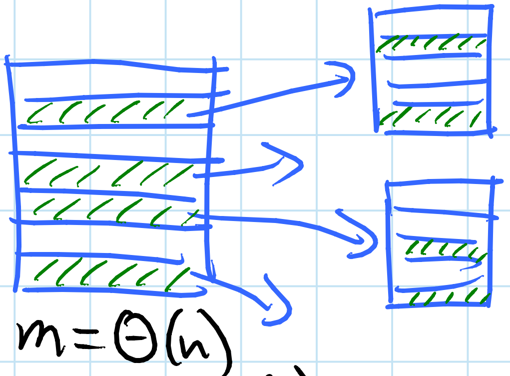
↑over random h ↑not random

$\Rightarrow O(1)$ 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 $E[\#\text{collisions}] \approx \frac{1}{2}$
  $\Rightarrow$ get $\emptyset$ after $O(1)$ tries... but $O(n^2)$ space
- use this structure for storing chains



k items $\Rightarrow m=k^2$
NO COLLISIONS

2 levels

[CLRS 11.5]

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