Recitation 7 Notes 6.006 7 October

Note Title 2/27/2009

* Open Addressing

Open Addressing

- all elements stored in the hash table itself.
- systematically examine table slots for searching an element
- load factor α can never exceed 1

$h : U \times \{0, 1, \ldots, m-1\} \rightarrow \{0, 1, \ldots, m-1\}$

For every key k, probe sequence $\langle h(k,0), h(k,1), \ldots, h(k,m-1) \rangle$.

should be a permutation of $\langle 0, 1, 2, \ldots m-1 \rangle$

```
HASH-INSERT(T,k)
  i=0
  while i<m
      j=h(k,i)
      if T[j]==NIL
          T[j]=k
          return j
      else i=i+1
```

```
HASH-SEARCH(T,k)
  i=0
  while i<m
      j=h(k,i)
      if T[j]==k
          return j
      if T[j]==NIL
          return NIL
      i=i+1
  return NIL
```

DELETION

mark the deleted slot as DELETED instead of NIL

- no longer dependent on α
- prefer chaining when keys are to be deleted.

Linear Probing

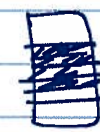
$h' : U \rightarrow \{0, 1, \ldots, m-1\}$

$h(k,i) = (h'(k) + i) \bmod m$

$< T[h'(k)],\ T[h'(k)+1],\ \ldots\ldots >$

determines the entire sequence

m possible values

→ Primary Clustering problem : clusters arise  $\}i \rightarrow$ fill probability $\left(\frac{i+1}{m}\right)$

Quadratic Probing

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

→ better than linear probing, but c_1, c_2 & m constrained

→ secondary clustering : if $h(k_1,0) = h(k_2,0) \Rightarrow h(k_1,i) = h(k_2,i)$

m possible distinct probe sequences

Double Hashing

$h(k,i) = (h_1(k) + i\,h_2(k)) \bmod m$

→ $h_2(k)$ relatively prime to m

- $m = 2^p$, $h_2 \to$ odd number
- $m \to$ prime, $h_2 \to < m$

* $\Theta(m^2)$ probe sequences $(h_1(k), h_2(k)) \Rightarrow$ distinct probe sequence

Q An open addressing table that resolves collisions using linear probing is initially empty. Key k_1 is inserted into the table first, followed by k_2, and then k_3.

i) What is the probability that searching for k_1 takes exactly two probes?

→ 0

ii) Suppose k_2 is deleted. What is the probability that searching for k_3 requires exactly three probes?

→

Case I	Case II
k_1	k_2
k_2	k_1
k_3	k_3

Case I: $Pr(k_2 \text{ below } k_1) = 2/m$

$Pr(k_3 \text{ hashes to } k_1) = 1/m$

$P_1 = 2/m \cdot \frac{1}{m} = \frac{2}{m^2}$

Case II: $Pr(k_2 \text{ above } k_1) = 1/m$

$Pr(k_3 \text{ hashes to } k_2) \rightarrow 1/m$

$P_2 = \frac{1}{m} \cdot \frac{1}{m} = \frac{1}{m^2}$

$P = P_1 + P_2 = 3/m^2$.