Hashing Examples

CS 2604
Summer II 2003
JP Vergara

Hashing Example 1

- Classic Hashing with M = 16 slots
  - h(K) = K % 16
  - Conflict Resolution Policy:
    (home + 5i) % M p(K,i) = 5i
- Probe Sequence for
  - K = 35: 3,8,13,2,7,12,1,6,11,0,5,10,15,4,9,14
  - K = 87:

Analyzing Probe Sequences

- Probe Sequence for
  - K = 35: 3,8,13,2,7,12,1,6,11,0,5,10,15,4,9,14
  - K = 87: 7,12,1,6,11,0,5,10,15,4,9,14, 3,8,13,2
- Ideally, when keys hash to different home positions, they should land in different slots
  - Above example OK except when slots 3,8,13, & 2 are all occupied (when else?)
- Note: clustering problem not really solved

Hashing Example 2

- M = 7 slots
  - h(K) = K % 7
  - Conflict Resolution Policy:
    (home + i^2) % M $p(K,i) = i^2$
- Probe Sequence for
  - K = 35: 0,1,4,2,2,4,1,0,…
  - K = 87: 3,4,0,5,2,0,4,3,…
- Undesirable because no guarantee all slots will be in sequence

Hashing Example 3

- M = 7 slots
  - h(K) = K % 7
  - Conflict Resolution Policy:
    (home + 3^i) % 7 p(K,i) = 3^i
- Probe Sequence (assume i starts with 0) for
  - K = 35: 0,1,3,2,6,4,5
  - K = 87: 3,4,6,5,2,0,1
- Clustering does not happen
  - Look at probe sequences for other home positions

Hashing Example 4

- If p( ) uses K, not just i,
  secondary clustering can be avoided
- Example: h(K) = K % 11;
  collision resolution: home + (1+K%10)i % 11
- Probe sequence for
  - K = 100: 1,2,3,4,5,6,7,…
  - K = 122: 1,4,7,10,2,5,8,…

  (same home position)

"Probe Sequence" for Bucket Hashing

- Suppose h(K) = K % 10^d
  and bucket size = s
- Absolute positions in hash table:
  0,1,2,…10^ds-1,10^ds, 10^ds+1,…
  overflow space
- If s = 5, and d = 1,
- "Probe sequence" for K = 135:
  25,26,27,28,29,50,51,52,53, …

What you need to know

Given the hash function, collision resolution policy, and a key

- Compute for the home position of the key
- Determine its probe sequence
- Determine slot where element will land, given state of the hash table

Other considerations:

- Consequences on time complexity
- Distribution of elements in input and hash table