Lec10_Open addressing, cryptographic hashing

Open Addressing

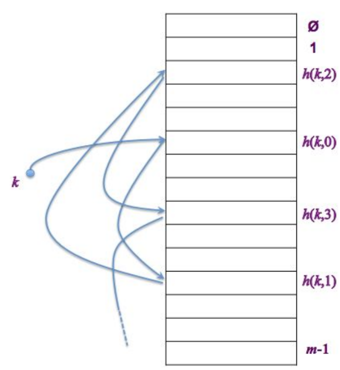
Another approach to collision :

- no chaining; instead all items stored in table
- One item per slot ⇒ m ≥ n
- hash function specifies order of slots to probe (try) for a key (for insert/search/delete)
  We want to design a function h, with the property that for all k ⊂ U :

$$h : \underbrace{\mathcal{U}}_{\text{universe of keys}} \times \underbrace{\{0, 1, \ldots, m-1\}}_{\text{trial count}} \to \underbrace{\{0, 1, \ldots, m-1\}}_{\text{slot in table}}$$

$$\langle h(k,0), h(k,1), \ldots, h(k,m-1) \rangle$$

is permutation of 0,1,2, … , m-1.



- Insert(k,v) : keep probing until an empty slot is found. Insert item into that slot.

```
for in xrange(m)
    If T[h(k,i)] is None → empty slot
      T[h(k,i)] = (k,v)    → store item
    return
raise “full”
```

- Search(k) : As long as the slots you encounter by probing are occupied by keys /= k, keep probing until you either encounter k or find an empty slot → return success or failure respectively.

```
for  i in xrange(m) :
    if T[h(k,i)] is None :
```

```
        return None
    elif T[k(h,i)][∅] == k :
        Return T[h(k,i)]
    return None
```

- Deleting Items
    - Replace item with special flag : “DeleteMe”, which Insert treats as None but Search doesn’t

Probing Strategies

- Linear Probing
  h(k,i) = (h’(k) + i) mod m where h’(k) is ordinary hash function
    - problem : clustering-cluster : consecutive group of occupied slots as clusters, become longer, it gets more likely to grow further
    - can be shown that for $0.01 < \alpha < 0.99$ say, cluster of size $\Theta(\log n)$
- Double Hashing
  h(k,i) = (h1(k) + i*h2(k)) mod m where h1(k) and h2(k) are two ordinary hash functions
  ⇒ actually hit all slots if h2(k) is relatively prime to m for all k
  (h1(k) + i*h2(k) mod m = h1(k) + j*h2(k) mod m ⇒ m divides (i-j)

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
    - Suppose we have used open addressing to insert n items into table of size m. Under the uniform hashing assumption the next operation has expected cost of $\le 1/(1-\alpha)$, where $\alpha = n/m$

Open Addressing vs. Chaining

- Open Addressing : better cache performance
- Chaining : less sensitive to hash function