Hashing II

Andrew C. Lee

EECS, Syracuse

Contents

Dictionary Problem Revisited

S = Set of items
We can classify the dictionary problem over S into two types:

1. *Static:* (More restrictive)
   Dictionary is fixed. We want to look up the items in S quickly
2. *Dynamic:* (More General)
   The items are being processed continually; have to handle a sequence of insert, delete and search operations

Question: What will be a good strategy for the more restrictive case ?
Question: Can we beat *binary search* ? Will Hashing perform better ?

Universal Hashing

Idea: *Randomize* the hash function

1. Construct a class of hash functions $\mathcal{H}$
2. Each hash function $h \in \mathcal{H}$ is:

$$h : U \to \{0, \ldots, m-1\} \quad m \text{ is the table size}$$

3. $\mathcal{H}$ is universal, which means for any pair of distinct keys $k, l \in U$:

$$\text{No. of hash functions in } \mathcal{H} \text{ with } h(k) = h(l) \leq \frac{|\mathcal{H}|}{m}.$$

An Example of Universal Hash functions

1. Choose a large prime number p, $p > k$ for any possible key k
2. Let $a \in \{1, \ldots, p-1\}$, $b \in \{0, 1, \ldots, p-1\}$. The hash function in $\mathcal{H}$ is of the following form:

$$h_{a,b} = [(ak + b) \bmod p] \bmod m$$

Perfect Hashing I

Perfect Hashing (Komlos and Szemeredi);

Use a two level hashing scheme (use universal hashing in each level)

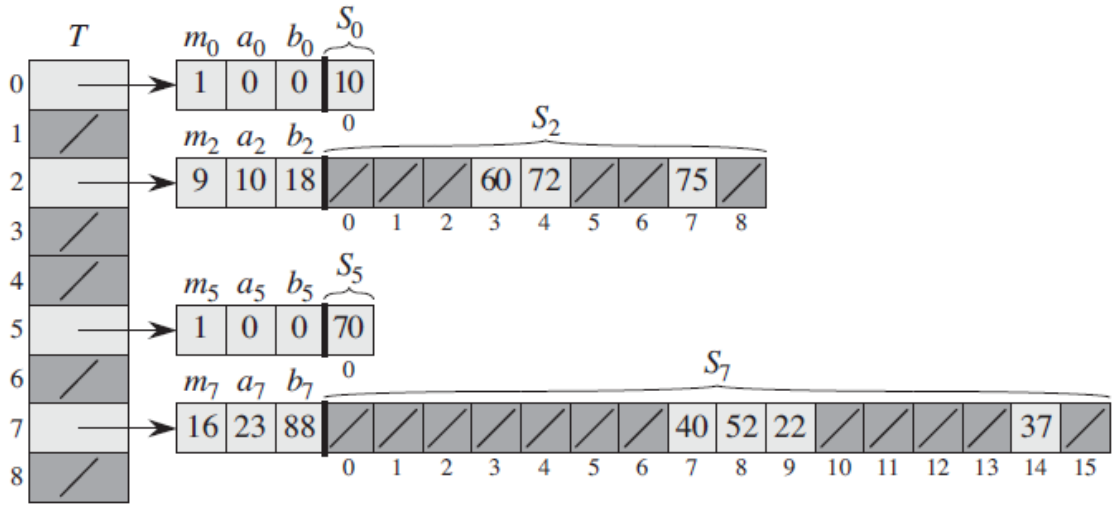

Figure 11.6 Using perfect hashing to store the set $K = \{10, 22, 37, 40, 52, 60, 70, 72, 75\}$. The outer hash function is $h(k) = ((ak + b) \bmod p) \bmod m$, where $a = 3$, $b = 42$, $p = 101$, and $m = 9$. For example, $h(75) = 2$, and so key 75 hashes to slot 2 of table T. A secondary hash table S_j stores all keys hashing to slot j. The size of hash table S_j is $m_j = n_j^2$, and the associated hash function is $h_j(k) = ((a_j k + b_j) \bmod p) \bmod m_j$. Since $h_2(75) = 7$, key 75 is stored in slot 7 of secondary hash table S_2. No collisions occur in any of the secondary hash tables, and so searching takes constant time in the worst case.

Figure : Example: perfect hashing. The size of hash table $S_j = m_j = n_j^2$

Perfect Hashing II

1. Search: $O(1)$ worst case time.
2. The expected amount of storage for all secondary hash tables is less than $2n$ (See CLRS, Col 11.11)

Final remarks

1. Hashing is a practical tool to support search
2. A good hashing scheme should have:
   - Few collisions
   - Hash table size $m = O(n)$ (n: size of the collection of possible keys)
   - the hash function h is quick to compute
3. In your computations, are the keys static ? If so, consider using a perfect hashing scheme