CSE 674

Advanced Data Structures
&
Algorithms

SYRACUSE UNIVERSITY
ENGINEERING
& COMPUTER
SCIENCE

Hashing I

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Contents

1. Reading: CLRS Chapter 11, Section 1 to 3 and Brass Chapter 9.
2. Direct Address Table
3. Hash tables and Hash functions
4. Collision Resolution Strategies

Direct Address Tables

An *ideal* situation

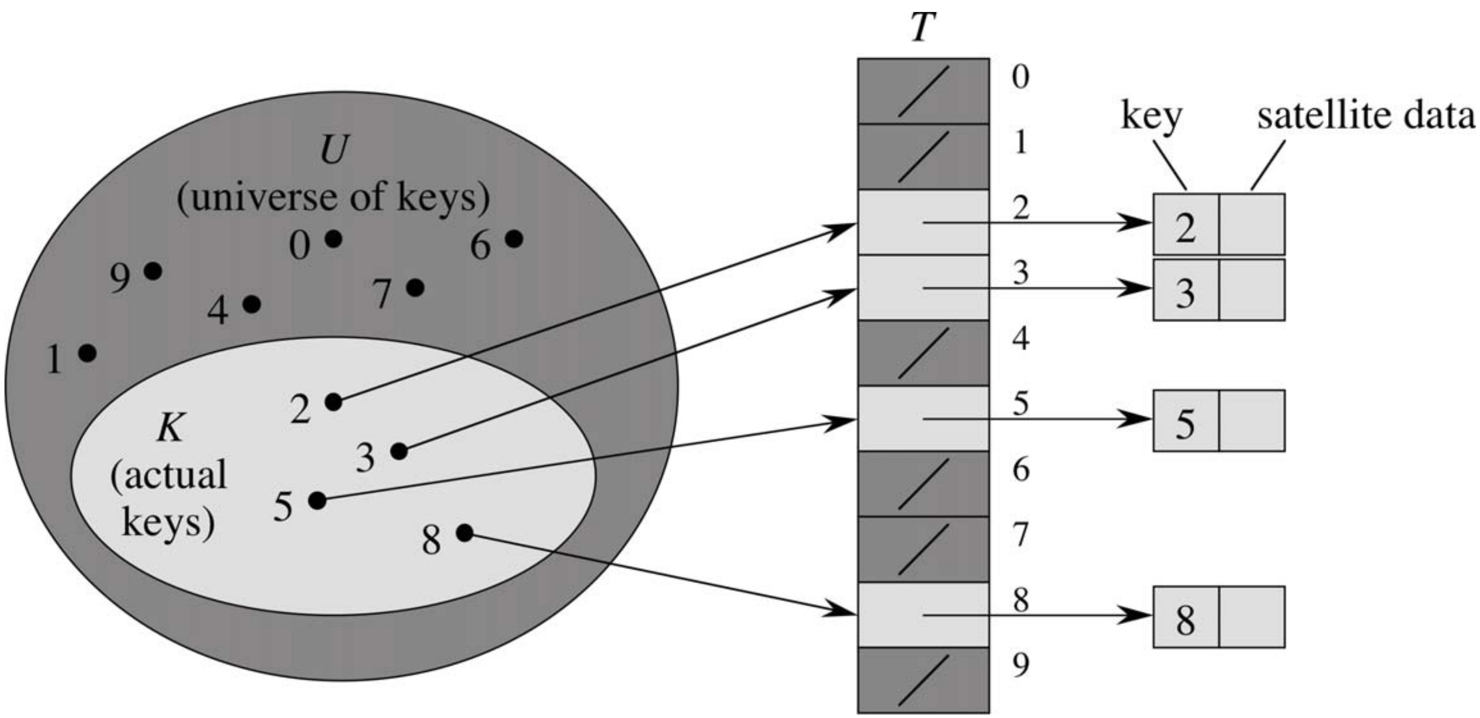

Figure : A Direct Address Table

Hash Tables

An *common* situation

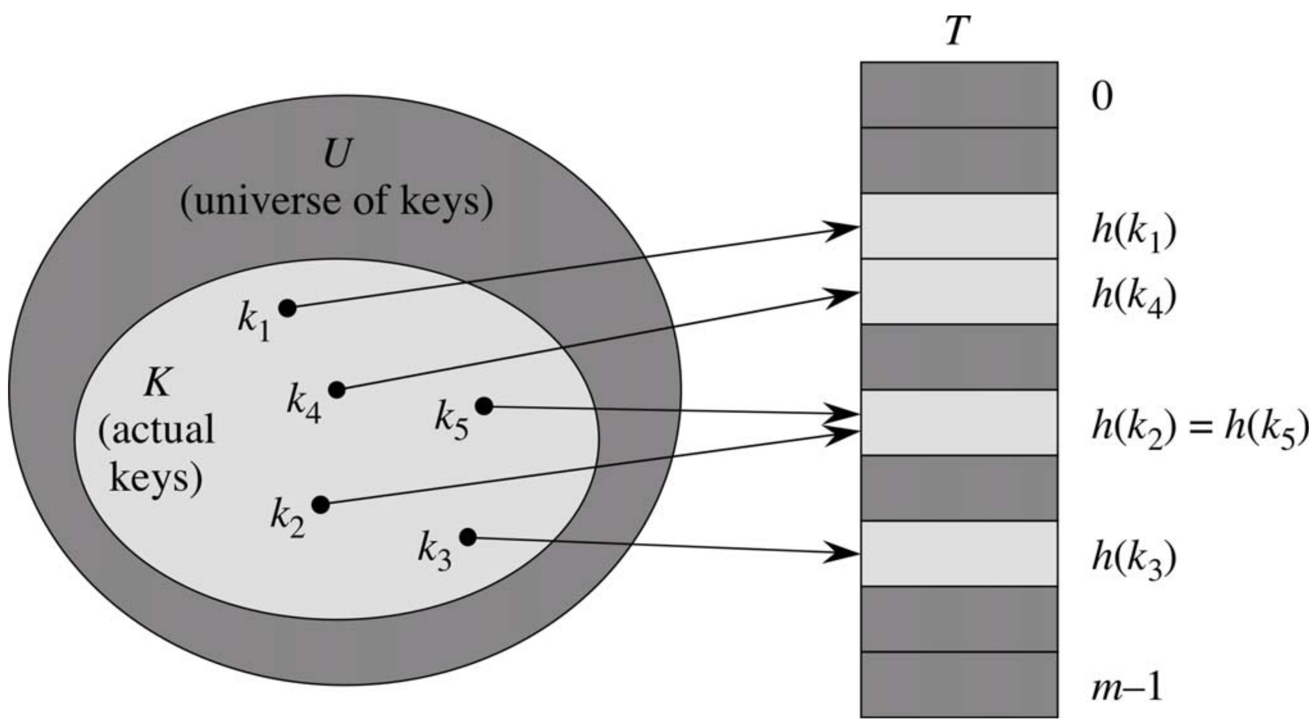

Figure : A Hash Table

Question What is h ? What is m ?

Example of Hash functions

1. Shift Folding
   Take parts of the key and add them together:
   A social security number (123-45-6789) can be divided into three parts and added
   $123 + 45 + 6789 = 6957$
   Then you can take the modulus of the table size
2. Boundry Folding
   The key is again divided into parts, but every other part is reversed
   $(123\text{-}45\text{-}6789) = 123 + 54 + 6789 = 6966$
   Again, then you can take the modulus of the table size
   Using bits that can actually reversing 456 is faster.

Example of Hash functions

1. Mid-Square Function
   Take the key, square it, and take the middle bits
   With this hash function in practice it is okay to have a power-of-two sized hashtable
2. Extraction
   Take only some of the bits or digits in a key
   Maybe all student id's start with 999, use the rest of the id as the key

Collision Resolution Strategies

1. Separate Chaining
2. Open Addressing (elements occupy the table itself)
    2.1 Linear Probing
    2.2 Quadratic Probing
    2.3 Double Hashing
3. Coalesced hashing

Separate Chaining: Ideas

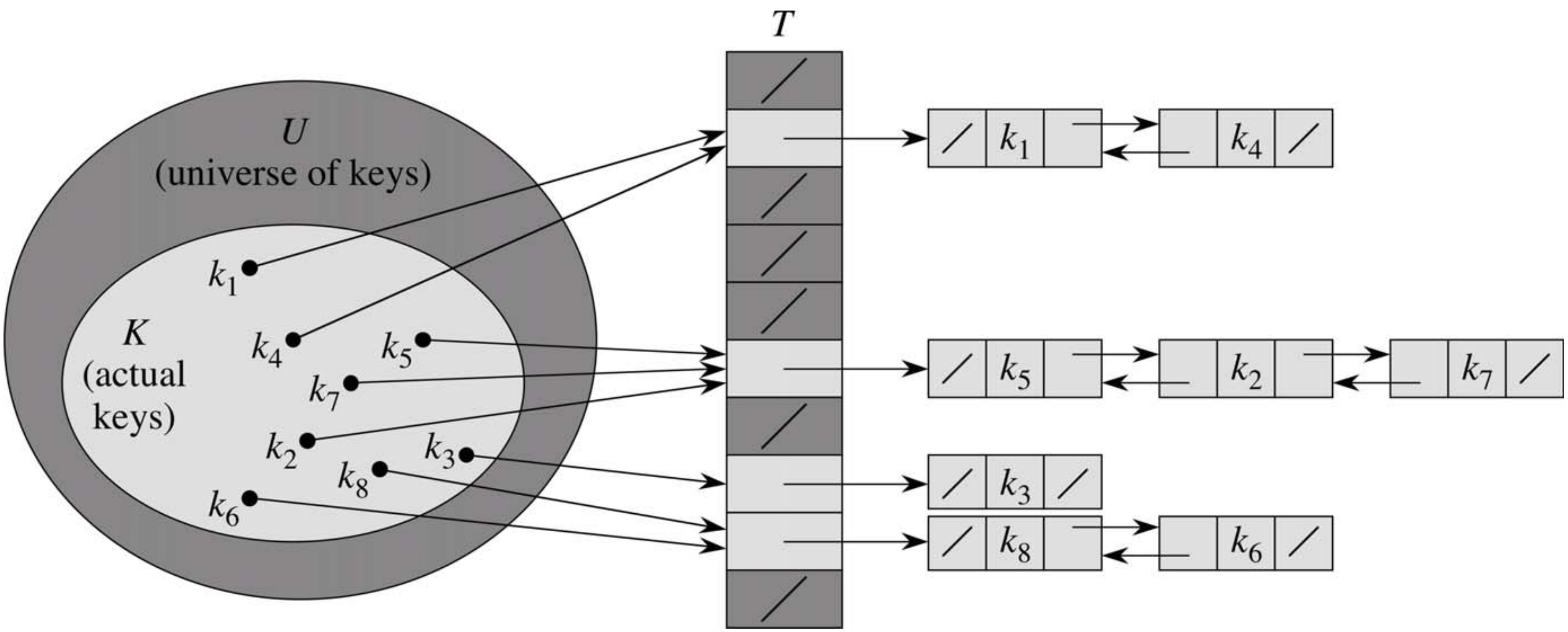

Figure : Build a Hash Table via Separate Chaining

Separate Chaining: Performance

1. Worst case is very bad (Why ?)
2. Interest: Average Case Performance
   Depends on how well the hash function h distributes the set of keys to be stored among the m slots, on the average.
3. When n (no. of keys) $= O(m)$ all dictionary operations is $O(1)$ run time *on average* (link lists used are double linked lists)

Open Addressing I

U: Universe of keys
m: Size of the hash table
h': Original hash function

1. The hash function is written as

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

   $h(k, i) =$ the address of i^{th}probe.
2. The probing sequence is $h(k, 0), \ldots, h(k, m-1)\}$.

Open Addressing II

1. Linear Probing: $h(k, i) = (h'(k) + i) \bmod m$
2. Quadratic Probing: $h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$
3. Double Probing: $h(k, i) = (h_1(k) + ih_2(k)) \bmod m$

Examples

We will use the examples from past lecture notes to illustrate how hashing works in the above cases.

Question

Question Start with an empty hash table of size 11, insert the following elements in the given order via the linear probing scheme. Draw the resulting hash table.

$$17, 50, 5, 20, 21, 6, 23, 9, 989.$$

Hashing II

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:

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

An implementation example

We will study an implementation of a hash table for integers with a universal hash function (as described in `Democode38.pdf`).

Question

Question Fill in the blanks:

In prior discussions, when we use universal hash functions of the form

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

the number p is a _______ number that are _______ than the values of any key k.

Hashing III

SYRACUSE UNIVERSITY ENGINEERING & COMPUTER SCIENCE

Perfect Hashing I

Perfect Hashing (Komlos and Szemeredi);
Use a two level hashing scheme (use universal hashing in each level)

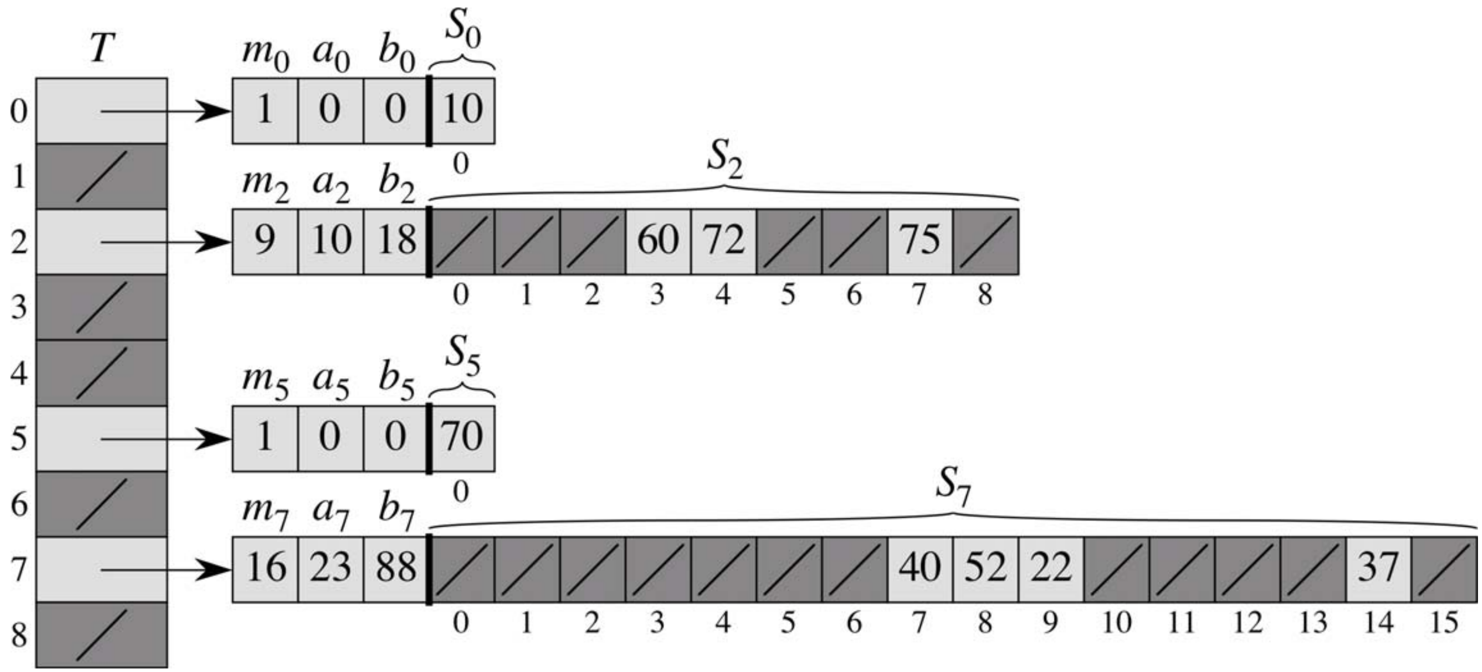

Figure : Example: perfect hashing. The size of hash table $S_j = m_j = n_j^2$

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)

An implementation example

We will study an implementation of a hash table for integers with a universal hash function (as described in `Democode40.pdf`).

Question

Question Fill in the blanks:

Perfect Hashing provides excellent ________ case performance when the set of keys are ________.

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

SYRACUSE UNIVERSITY
ENGINEERING
& COMPUTER
SCIENCE