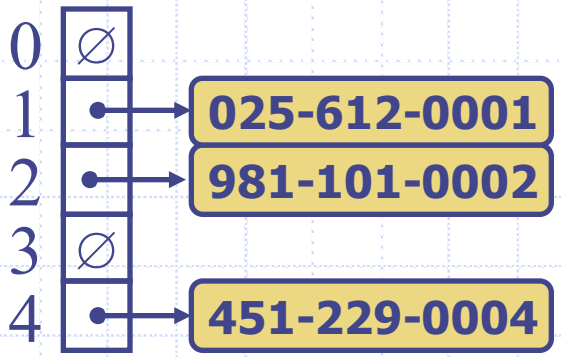
Hash Tables 2



How Hash Table Works

- A hash table for a set of key-value pairs, consists of two components,
  - Hash function ***h().***
  - An array (the container of the records) of size ***N***
- When implementing a hash table, the goal is to store a record (***k***, ***v***) at index ***i*** = ***h***(***k***) in that container array.
  - ***h(k) yields the index at which the record will be stored in the Array.***

Hash Functions and Hash Tables

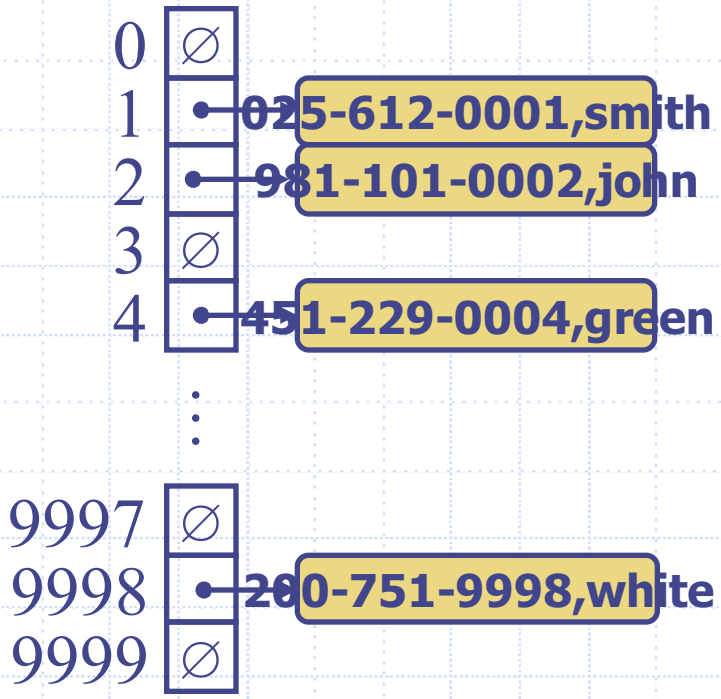
- A hash function h maps keys of a given type to integers in a fixed interval $[0, N-1]$
- Example:

  $$h(x) = x \bmod N$$

  is a hash function for integer keys
- The integer $h(x)$ is called the hash value of key x

Example

- We design a hash table for storing entries as (SSN, Name), where SSN (social security number) is a nine-digit positive integer
- Our hash table uses an array of size $N = 10{,}000$ and the hash function
  $h(x) = \text{last four digits of } x$



Hash Functions

- A hash function is usually specified as the composition of two functions:

  Hash code:

  h_1: keys $\rightarrow$ integers

  Compression function:

  h_2: integers $\rightarrow [0, N-1]$

- The hash code is applied first, and the compression function is applied next on the result, i.e.,

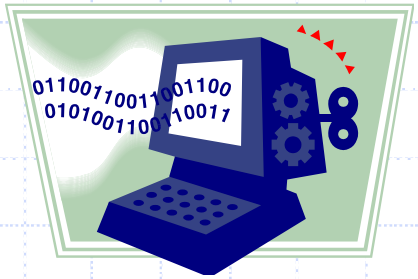
  $$h(x) = h_2(h_1(x))$$

- The goal of the hash function is to "disperse" the keys as evenly as possible into different array index locations.

Hash Functions

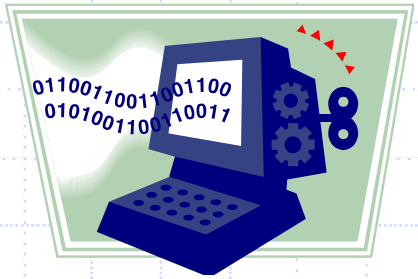
- The goal of the hash function is to "disperse" the keys as evenly as possible into different array index locations.
- It is highly possible that different keys will produce different hash values.
  - Still slightly possible that two keys have the same hash value.
- But, we are guaranteed that for the same key, hash function always produces the same hash value.

Hash Codes



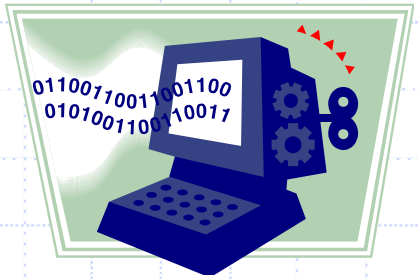
- Memory address:
  - We reinterpret the memory address of the key object as an integer (default hash code of all Java objects)
  - Good in general, except for numeric and string keys
- Integer cast:
  - We reinterpret the bits of the key as an integer
  - Suitable for keys of length less than or equal to the number of bits of the integer type (e.g., byte, short, int and float in Java)

Hash Codes



- Component sum:
  - We partition the bits of the key into components of fixed length (e.g., 16 or 32 bits) and we sum the components (ignoring overflows)
  - Suitable for numeric keys of fixed length greater than or equal to the number of bits of the integer type (e.g., long and double in Java)
  - We also apply Component Sum on strings.
    - E.g we have a string "abc"
    - In java or C, each character is represented by an int value with ASCII coding or Uni-code16 coding.
    - 'a' corresponds to 97, 'b' corresponds to 98 and 'c' is 99.
    - Then, ComponentSum("abc") = 97 + 98 + 99 = 294

Hash Codes



- Hash code function for an object **B** of any user-defined classes,
  - Step 1, Calling B.toString() returns a string **S** for B.
  - Step 2, Apply the Component Sum idea to the string **S**, which produces an integer value.

Compression Functions

- Division:
  - $h_2(y) = y \bmod N$
  - The size N of the hash table is usually chosen to be a prime
  - The reason has to do with number theory and is beyond the scope of this course