CS367 Announcements
Tues, July 30th, 2013

- P3 due Wed July, 31st 11:59pm

Last Time

- Hashing

Today

- Hashing (cont.)

Recall Hashing

Goal: O(1) lookup, insert and delete ops

Idea: store items in array and compute index from key-hash function

Concerns

- table size
- hash function
- collisions

Properties of a good hash function

- quick/easy to compute
- spreads data evenly across the hashtable
- minimizes collisions
- uses all of the unique part of key (that distinguish different data values)

Typically, a hash function has 2 steps:

1. convert search key to an integer **hash code**

2. compress hash code into a **hash index** that's valid for the hashtable

Collision Handling using Open Addressing

Linear Probing

Quadratic Probing

Collision Handling using Open Addressing (cont.)

Double Hashing

Collision Handling using Buckets

Java API Support for Hashing

`hashCode` **method**

- method of `Object` class
- returns an `int`
- default hash code is BAD - computed from object's memory address

Guidelines for overriding `hashCode`**:**

Java API Support for Hashing (cont.)

`Hashtable<K,V>` **class**

- in `java.util` package
- implements `Map<K,V>` interface
- constructors allow you to set
  - initial capacity (default = 11)
  - load factor (default = 0.75)

`HashMap<K,V>` **class**

- in `java.util` package
- essentially the same as `Hashtable` (except `Hashtable` is synchronized)

TreeMap **vs.** HashMap