Lecture 27: Mappings

CS 62
Spring 2015
Kim Bruce & America Chambers

Week Ahead

- Please fill out survey
- This week's assignment long!
  - Read carefully == lots of pieces
  - Recommend work in pairs
  - Turn in code plus report answering questions
- Quiz on Friday
- Next week Prof. Chambers starts C++
  - Get "C++ for Java Programmers" by Weiss

Census Data

- Simple way: Load data into Census Data
  - Find data for rectangle in parallel or sequential
- Preprocess:
  - Load data as above (in parallel or sequential)
  - Create 2-d array of rectangles extending from upper left.
  - Calculate 2-d array sequentially
  - Can use to calculate any rectangle in constant time
- Run experiments for timing & write full report!
  - 8 of 20 points!!

Map<K,V>

- Collection of associations between a key and associated value, e.g. name & phone number
  - Though doesn't use Bailey's Association class
- As usual lots of implementations
- Also called dictionaries after example
  - Look up table!

```
public interface Map<K,V> {
    public int size();
    public boolean isEmpty();
    public boolean containsKey(Object k);
    public boolean containsValue(Object v);
    public V get(Object k);
    public V put(K k, V v);
    public V remove(Object k);
    public void putAll(Map<K,V> other);
    public void clear();
    public Set<K> keySet();
    public Collection<V> values();
    public Set<Map.Entry<K,V>> entrySet();
    public boolean equals(Object other);
    public int hashCode();
}
```

Map.Entry is essentially Association

Implementations

Data Structure	*Search*	*Insert*	*Delete*	*Space*
Linked List	O(n)	O(1)	O(n)	O(n)
Sorted Array	O(log n)	O(n)	O(n)	O(N)
Balanced BST	O(log n)	O(log n)	O(log n)	O(n)
Array[KeyRange]	O(1)	O(1)	O(1)	KeyRange

where n is # elts in table, N is # slots in array

Last row is array where keys are subscripts.

Hash Table

- Why is using keys as subscripts bad?
  - Restricts types of keys
  - keys often too sparse
  - Suppose use SS#'s as subscripts to table of students?
- *Instead provide function from keys to subscripts that is denser.*

Hash Functions

- Want H: EltType → Subscripts, where
  - H(elt) can be computed quickly
  - if e1 != e2 then H(e1) != H(e2)
    - H is one-to-one
- Called perfect hashing function
  - Hard to find unless know all keys in advance.
- Now adding, finding, removing all O(1)
- So important that hashCode function built-in to Java classes.

Hash Functions

- Look for reasonable function that scatters elements through array randomly so won't bump into each other.
  - Lose any ordering on keys
- Ideal is to find in time O(1).
- We want to:
  - Find good hashing functions
  - Figure out what to do if 2 elts sent to same locn
- *"A given hash function must always be tried on real data in order to find out whether it is effective or not."*

String-Valued Keys

- Convert from string to digits
  - Can use formula like Key(xy) = $2^8 *$ Ord(x) + Ord(y)
    - where ord(x) = ascii code (or unicode) for x
    - If use long ints then can get 4 letters into 1 number
    - Java uses for string s:
      `s[0]*31^(n-1) + s[1]*31^(n-2) + ... + s[n-1]`
  - Simple alternative, add together ord of all letters
    - Problem: words with same letters mapped to same place
    - E.g.: miles, slime, smile
- Similar w/other structured types
  - Combine hash of pieces, but not depend on structure

Cutting Down

- If hash code too large for table:
  - Choose digits from certain positions of key.
    - E.g., last 4 digits of SS#
  - Let H'(key) = H(key) mod TableSize
    - generally best if TableSize is prime.
  - Square the key and then select certain bits.
    - Usually the middle half of the bits is taken.
    - Multiplication ensures all digits used in computation
  - Folding:
    - Break key into pieces and add them up

Well-defined Hash Functions

- Require that if K1.equals(K2) returns true then H(K1) = H(K2)
  - Consider fractions 2/3, 4/6 represented in Fraction class w/instance vbles num, denom.
  - If H(2/3) ≠ H(4/6) then put into table in different places -- might not find if one in table and look up other.
- Hence, if redefine equals then must redefine hashCode so x.equals(y) =>
  x.hashCode() == y.hashCode()

Important!!

- How important?
  - Eclipse include automated way of generating equals and hashcode methods under “Source” menu.
- What if insert item into hash table and then change instance vble which affects hash code?
  - Like changing priority of elt in priority queue or key of elt in ordered structure!