Lecture 19: Hash Code & Implementations

generating ideas for hash tables:
→ how to define $h(x)$?

Terminology: a hashfunction h is a rule for assigning (numerical) values to object instances.

- $h(x)$ = hash code of x or hash value

Goal: Fix a rule for generating hash codes where values appear random
- collisions are unlikely.
  (x and y collide if $h(x) = h(y)$ but $x \neq y$).

In Java: Object class has a method

`hashCode()`: for any object, obj, calling obj.hashCode() returns an int that is a hash code for obj.

Guarantees:

(1) if obj.hashCode() involved multiple times, same execution, semantically same object gets same value.

(2) if o.equals(p), then o.hashCode() = p.hashCode()
but reverse is not necessarily true

Non-Guarantee:

- "as much as reasonably possible", hashCode() should avoid collisions

Conclusion: Defining hashCode() for a class is an artform.

↳ should NOT assume that default implementation is great... "trust no one".

Two issues: toward implementing hashTable

1. x.hashCode() → negative value (Index out of bounds)
2. hashCode may have patterns (not random) and lead to poor performance

Goal: Use hashCode() method to generate indices for a hash table of size n.

- assume $n = 2^d$

Ideal: If o.hashCode() ≠ p.hashCode()

then, $P(h(o) = h(p)) = \frac{1}{n} \left(= \frac{1}{2^d}\right)$

Simple approach: x = hashCode(obj) (some int)

determine $h(x) = |x| \% n$ ← return between 0 and n−1

↖ make positive

Problem Doesn't satisfy ideal: if $x = 0$, $y = n$

$\Rightarrow h(x) = 0$, $h(y) = 0$.

Diff. way of hashing

- choose z at random (int)
- define h as $h(x) = |z * x| \% n$

Better or worse than before?

- what if z is divisible by n?
→ everything hashes to 0!

Terrible... but...

A Tangent ⇒ int are 32 bit binary rep. of #s:

$$\underbrace{1\,0\,0\,1\,0\,0\,1\,1\,1\,0\ldots0\,1}_{32}$$

31 value bits
from 0 to $(2^{31}-1)$.

sign bit
1 → neg #
0 → pos #
size is fixed

eg. $(2^{31}-1)$: 0 1 1 1 1 1 1 ... 1

\+ 1 : 0 0 0 0 0 0 ... 1

"2s complement representation" 2^{31} : 1 0 0 0 0 0 0 0 ... 0 0 or -2^{31}

Recall: $n = 2^d$

Fact: If z is a random odd number and x is any odd number, then $(z * x) \% n$ is equally likely to be any value in $0, 1, \ldots, n-1$.

More generally: $x = \underbrace{0\,1\,0\,0\,1\ldots0}_{d}\underbrace{1\,0\,0\ldots0}_{32-d}$

then in binary rep of $z * x$ (as int),

$$z * x = \underbrace{*\,*\,*\ldots*}_{d}\underbrace{0\,0\ldots0}_{32-d}$$

all 0/1 values of * are equally likely.

Consequence: if $x \neq y$, and z is an odd number chosen at random, then the first d-bits of $z * x$ and $z * y$ are equals with probability of $\left(\frac{1}{2^d} = \frac{1}{n}\right)$. (randomness over choice of z)

Suggests: given x = obj.hashCode()
define $h(x)$ = first d bits of $z * x$.

In code: >>> = bit shift operator.

$$\begin{cases} y = b_{31}\, b_{30}\, b_{29} \ldots b_0 & \text{(bits of } y\text{)} \\ y >>> j = \underbrace{00\ldots0}_{j}\, b_{31}\, b_{30} \ldots b_j \end{cases}$$

first d bits of y is $y >>> (32-d)$