Q&A

Q. How does Java implement `hashCode()` for `Integer`, `Double`, and `Long`?

A. For `Integer` it just returns the 32-bit value. For `Double` and `Long` it returns the *exclusive or* of the first 32 bits with the second 32 bits of the standard machine representation of the number. These choices may not seem to be very random, but they do serve the purpose of spreading out the values.

Q. When using array resizing, the size of the table is always a power of 2. Isn't that a potential problem, because it only uses the least significant bits of `hashCode()`?

A. Yes, particularly with the default implementations. One way to address this problem is to first distribute the key values using a prime larger than `M`, as in the following example:

```
private int hash(Key x)
{
   int t = x.hashCode() & 0x7fffffff;
   if (lgM < 26) t = t % primes[lgM+5];
   return t % M;
}
```

This code assumes that we maintain an instance variable `lgM` that is equal to lg M (by initializing to the appropriate value, incrementing when doubling, and decrementing when halving) and an array `primes[]` of the smallest prime greater than each power of 2 (see the table at right). The constant 5 is an arbitrary choice—we expect the first `%` to distribute the values equally among the values less than the prime and the second to map about five of those values to each value less than `M`. Note that the point is moot for large `M`.

k	δ_k	primes[k] $(2^k - \delta_k)$
5	1	31
6	3	61
7	1	127
8	5	251
9	3	509
10	3	1021
11	9	2039
12	3	4093
13	1	8191
14	3	16381
15	19	32749
16	15	65521
17	1	131071
18	5	262139
19	1	524287
20	3	1048573
21	9	2097143
22	3	4194301
23	15	8388593
24	3	16777213
25	39	33554393
26	5	67108859
27	39	134217689
28	57	268435399
29	3	536870909
30	35	1073741789
31	1	2147483647

Primes for hash table sizes

Q. I've forgotten. Why don't we implement `hash(x)` by returning `x.hashCode() % M`?

A. We need a result between `0` and `M-1`, but in Java, the `%` function may be negative.

Q. So, why not implement `hash(x)` by returning `Math.abs(x.hashcode()) % M`?