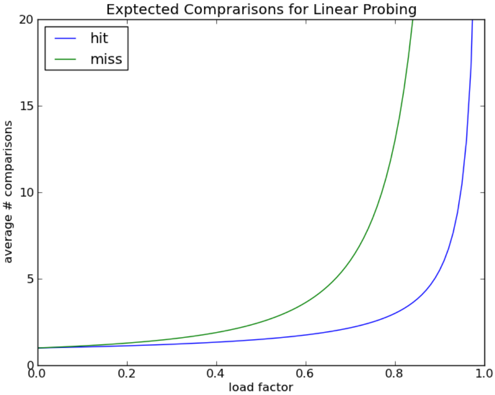
CS240

Nathan Sprague

November 30, 2012

(From Last Week)

- Expected comparisons when simple linear probing is used:



Bit-Manipulation Based Hashing

- If the size of a hash table is a power of two, division method can use bitwise AND:
  - 00001001 = 9
  - 00000111 = 8-1
  - These are equivalent:
    - 9 % 8 = 1
    - 00001001 & 00000111 = 00000001 = 1
- Why is that relevant?
- Let's time it...

C#/.net

- Prime-number-sized tables.
- Collision resolution: Double hashing.
- Default load factor: .72

(http://msdn.microsoft.com/library/ms379571.aspx)

Ruby

- Prime-number-sized tables.
- Collision resolution: chaining
- Default load factor: 5

Hash implementation in st.c. Ruby source can be downloaded from: http://www.ruby-lang.org/en/

Python Dictionary Implementation

- Power-of-two table sizes.
- Hash function: grab lower order bits (no effort to avoid collisions)
- Collision resolution: fancy double hashing:
    - Original hash *j* is modified according to:
    - j = ((5*j) + 1 + PERTURB)
    - PERTURB is initialized to the original hash, then bit-shifted after every collision.
    - (All of this can be done with bit-level operators.)
- Default load factor: 2/3

Implementation in: Python-2.7.2/Objects/dictobject.c. Source can be downloaded from www.python.org.

Java HashMap Implementation

- Power-of-two table sizes.
- Hash Function: “Bit Scrambling” then use low order bits.
- Collision resolution: Chaining
- Default load factor: .75

```java
static int hash(int h) {
    h ^= (h >>> 20) ^ (h >>> 12);
    return h ^ (h >>> 7) ^ (h >>> 4);
}

static int indexFor(int h, int length) {
    return h & (length-1);
}
```

(code from HashMap.java, OpenJDK, v.7 GPLV2: http://download.java.net/openjdk/jdk7/)