Chapter 6, Hash tables:

- General introduction; separate chaining (last week Friday)
- Open addressing (Monday)
- Hash functions (**Today**)
- Practice open addressing (Thursday lab)
- Perfect hashing (week-after Monday)
- Hash table performance (week-after Wednesday)

Today:

- Hash function properties
- Integer hashes
- String hashes
- Experimental results

see Exercise 7.12. Suppose instead we use a *quadratic* formula:

$$probe(k, i) = (h(k) + a \cdot i + b \cdot i^2) \% m$$

But to be a legitimate probe sequence, this function must hit every possible slot in the table. In mathematical terms, it must be a permutation of the table positions, or a one-to-one correspondence from $[0, m)$ to itself. Otherwise, a search for a suitable place for a key would fail even though the table is not full. This makes it difficult to find a table size $m$ and coefficients $a$ and $b$, but one choice that works[3] is to let $m$ be a power of 2 and $a = b = \frac{1}{2}$. In the code below, note how we ensure that the table size is a power of two. This is why the `ProbeStrategy` interface includes methods for determining the size of the table.

[3] Proving that this is a permutation is often left as an exercise, as in Exercise 6.4.20 of Donald E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. 2nd ed. Addison Wesley Longman Publishing Co., Inc., 1998, p. 551. For a solution, see Fabian Giesen. *Triangular numbers mod $2^n$*. `https://fgiesen.wordpress.com/2015/02/22/triangular-numbers-mod-2n/`. 2015.

Hash functions should distribute the keys *uniformly* and *independently*.

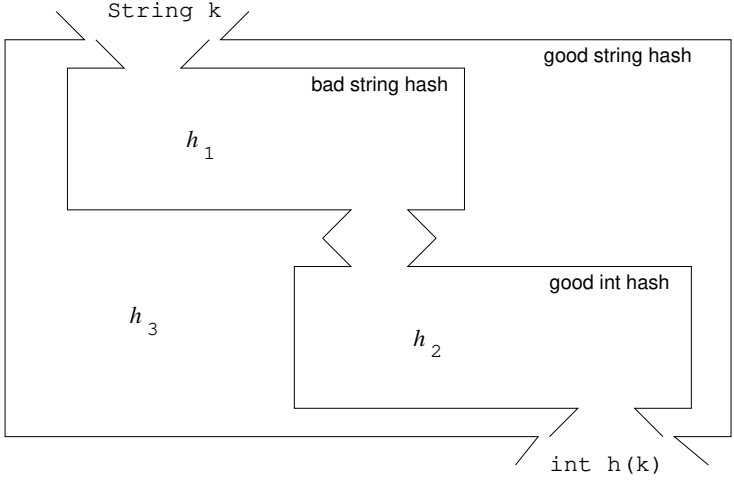
Uniformity:

$$P(h(k) = i) = \frac{1}{m}$$

Independence:

$$P(h(k_1) = i) \quad = \quad P(h(k_1) = i \mid h(k_2) = j)$$

Why do we talk about integer hashes?



Division method:

$$h(k) = k \mod m$$

Middle square method:

| | **Decimal** | **Binary** |
|---|---|---|
| **Original** | 37,914 | 0000 0000 0000 0000 1001 0100 0001 1010 |
| **Squared** | 1,437,471,396 | 0101 0101 1010 $\underbrace{1110\ 0001}_{\text{middle bits}}$ 0010 1010 0100 |
| **Middle 10 bits** | 225 | 0000 0000 0000 0000 0000 0000 1110 0001 |

Multiplicative method:

$$h(k) = \lfloor m(k \cdot a - \lfloor k \cdot a \rfloor) \rfloor$$

"Universal" hash (next time)

ASCII sum:

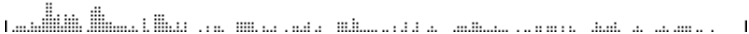
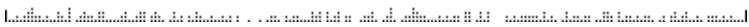
$$h(k) = \left( \sum_{i=0}^{n-1} s[i] \right)$$

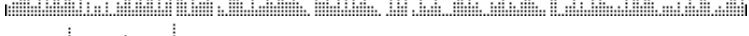
String polynomial:

$$h(k) = (k[0] \cdot b^{n-1} + k[1] \cdot b^{n-2} + \cdots + k[n-2] \cdot b + k[n-1]) \mod m$$

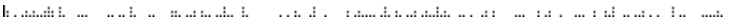
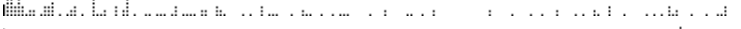
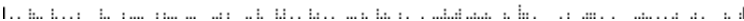
Carter-Wegman:

$$\begin{aligned} h(k) &= (h_0(k[0]) + h_1(k[1]) + \cdots + h_{n-1}(k[n-1])) \mod m \\ &= \left( \textstyle\sum_{i=0}^{n-1} h_i(k[i]) \right) \mod m \end{aligned}$$

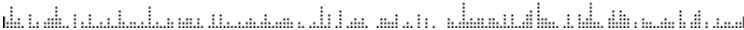
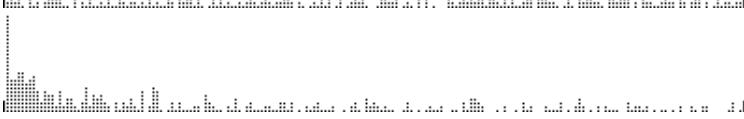
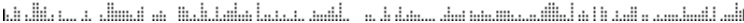
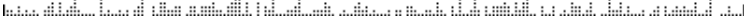
| Area codes ($n = 303$) | | Average penalty | Variance |
|---|---|---|---|
| Division |  | .673 | .808 |
| Mid square |  | 1.09 | 1.64 |
| Multiplicative |  | .508 | .478 |
| Fibonacci |  | .617 | .696 |
| Universal |  | .578 | .617 |

| Book ISBNs ($n = 718$) | | | |
|---|---|---|---|
| Division |  | .618 | 1.05 |
| Mid square |  | .812 | 1.48 |
| Multiplicative |  | .565 | .954 |
| Fibonacci |  | .544 | .873 |
| Universal |  | .667 | 1.15 |

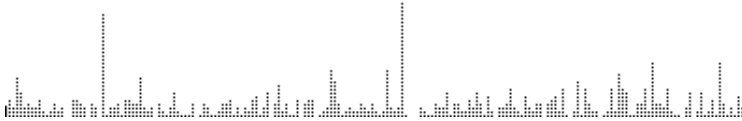
Randomly generated from $[0, 1000)$ ($n = 150$)

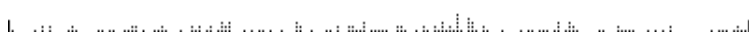
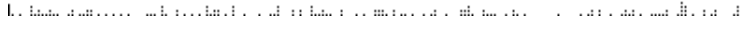
| | | Average penalty | Variance |
|---|---|---|---|
| Division |  | 1.36 | .958 |
| Mid square |  | 1.86 | 1.96 |
| Multiplicative |  | 1.34 | .919 |
| Fibonacci |  | 1.41 | 1.07 |
| Universal |  | 1.39 | 1.02 |

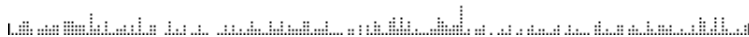
Randomly generated from $[0, 1000)$ ($n = 400$)

| | | | |
|---|---|---|---|
| Division |  | .518 | 1.16 |
| Mid square |  | 1.73 | 3.68 |
| Multiplicative |  | .405 | .930 |
| Fibonacci |  | .448 | .980 |
| Universal |  | .488 | 1.08 |

| Chemicals ($n = 663$) | | Average penalty | Variance |
|---|---|---|---|
| ASCII sum |  | .505 | 1.00 |
| String polynomial |  | .424 | .805 |
| Carter-Wegman |  | .800 | 1.63 |

| Books ($n = 718$) | | | |
|---|---|---|---|
| ASCII sum |  | .818 | 1.51 |
| String polynomial |  | .745 | 1.30 |
| Carter-Wegman |  | 2.06 | 4.08 |

| Randomly generated strings ($n = 150$) | | Average penalty | Variance |
|---|---|---|---|
| ASCII sum |  | 1.32 | .879 |
| String polynomial |  | 1.43 | 1.09 |
| Carter-Wegman |  | 1.41 | 1.05 |

| Randomly generated strings ($n = 400$) | | | |
|---|---|---|---|
| ASCII sum |  | .515 | 1.15 |
| String polynomial |  | .425 | .925 |
| Carter-Wegman |  | .540 | 1.20 |

**Coming up:**

*Do* **Optimal BST** *project (Due Mon, Nov 25)*
*Do* **Open addressing with linear probing** *project (due Monday, Apr 21)*

*Due* **Mon, Apr 21** *(end of day)*
*Read Sections 7.(4 & 5)*
*(No exercises or quiz)*

*Due* **Wed, Apr 23** *(end of day)*
*Re-read the last part of Section 7.3*
*Take quiz (hash table performance)*