Choosing a Hash Table Function

- Almost any function will do
  - But some functions are definitely better than others!

- Key criterion
  - Minimum number of collisions
    - Keeps chains short
    - Maintains $O(1)$ average length of chain
Uniform Hashing

- Ideal hash function
  - \( P(k) \) = probability that a key, \( k \), occurs
  - If there are \( m \) slots in our hash table,
  - A uniform hashing function, \( H(k) \), would ensure:

\[
\sum_{k \mid h(k) = 0} P(k) = \sum_{k \mid h(k) = 1} P(k) = \ldots \sum_{k \mid h(k) = m-1} P(k) = \frac{1}{m}
\]

(Read as sum over all \( k \) such that \( h(k) = 0 \))

- or, in plain English,
- the number of keys that map to each slot is equal

Hash Tables - A Uniform Hash Function

- If the keys, \( k \), are integers
  randomly distributed in \( [0, r) \),
- then

\[
h(k) = \left\lfloor \frac{mk}{r} \right\rfloor
\]

- is a uniform hash function

- Example
  - Add ASCII codes for characters mod, 255
    will give values in \( [0, 256) \) or \( [0, 255] \)
  - Replace + by \( \text{xor} \)
    - Same range, but without the mod operation
Hash Tables - Reducing the range to \([0, m)\)

- We’ve mapped the keys to a range of integers \(0 \leq k < r\)
- Now we must reduce this range to \([0, m)\)
  - where \(m\) is a reasonable size for the hash table

Strategies
- Division - use a mod function
- Multiplication
- Universal hashing

Using Division to Reduce hash range to \([0, m)\)

- Use a mod function
  - \(h(k) = k \mod m\)

- Choice of \(m\)?
  - Powers of 2 are generally not good!
  - \(h(k) = k \mod 2^n\)
    - selects last \(n\) bits of \(k\)

- Prime numbers close to \(2^n\) seem to be good choices
  - Eg: For ~4000 entry table, choose \(m = 4093\)

\[0110010111000011010\]
\[k \mod 2^n \text{ selects these bits} \]
\[0110010111000011010\]
Computing Hash Functions in Hardware

- XOR Functions are preferred
  - Avoids excessive hardware needed for multiplication, division, and long critical path required for carry chains in sums
- Bit shifting can help
  - Helps to randomize the high-order bits, since the ASCII values used to represent strings change mostly in the lower-order bits.
- Good to simulate function with real data to ensure that Hash function provides an even distribution

Hash Tables - Reducing the range to \([ 0, m )\)

- Universal Hash Function
  - Consider a set of functions, \( H \), which map keys \( x \) with \( r \) bits to \( H(x) \) with \( m \) bits
    - Input Range: \([0, r)\)
    - Output Range \([ 0, m )\)
  - \( H \) is a universal hash function, if for each pair of keys, \( x \) and \( y \), the number of functions for which
    \[ H(x) = H(y) \]
  is
    \[ |H|/m \]
Avoiding the Worst-case running times

- Using a well-known hash function can be bad
  - A determined "adversary" can always find a set of data that will defeat any hash function
    - Hash all keys to same slot causes $O(n)$ search
- For the paranoid ...
  - Select a hash function randomly at compile-time from a set of hash functions to reduce the probability of poor performance
- For the really paranoid ...
  - Pick a hash function randomly at run-time

Loadable Hash Generation Matrix
Example of Hash Matrix Generator

Example of how logic reduction reduces size of a circuit programmed at compile-time

- Logic reduction is **automatic**, and occurs during synthesis
- AND gates eliminated by identity property
Larger Example of Hash Matrix

- **Given**
  - 80-bit Input
    - Hash Function over 10 bytes
    - Length of our signatures
  - 12-bit Output
    - 4096 addresses
    - Size of our Block RAMs
  - Uniform and random distribution of \{0,1\} values in \(q\)
    - Universal hash function

- **Estimate number of**
  - AND gates
  - XOR gates
  - Logic depth

Example: Collision Frequency of Birthdays

- How many people does it take before the odds are > 50% that two people in a class of size \(n\) have the same birthday?

- **Model**
  - \(x =\) Birthday [Universe of all possible dates]
  - \(H(x) =\) DayNumber \((x)\)
    - There are 365 days in a normal year

- Are birthdays on the same day unlikely?
Distinct Birthdays

- Let $Q(n)$ = probability that $n$ people have distinct birthdays
  
  \[ Q(1) = 1 \]
  \[ Q(2) = Q(1) \cdot \frac{364}{365} \]
  - With two people, the 2nd has only 364 “free” birthday
  \[ Q(n) = Q(1) \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \ldots \cdot \frac{365-n+1}{365} \]
  - The 3rd has only 363, and so on:

Coincident Birthdays

- Probability of having two identical birthdays
- $P(n) = 1 - Q(n)$
- $P(23) = 0.507$
- With 23 entries, table is only $23/365 = 6.3\%$ full!