Hashing

- Design a data structure which allows
  1. Insertion of a record.
  2. Deletion of a record.
  3. Search of a record.
- A large number of records are there, say a million
- Linear data structures: require $O(n)$ time units for search, deletion.
- Trees: require at least $O(\log n)$ time units for every operation.

Hashing

- Every operation can be performed in $O(1)$ time
- Hashing is the answer. It has two components.
  1. A Hash function, and
  2. A Hash table.
- How it operates?
  - Takes a record key, and computes a value in O(1) time.
  - Inserts, extracts or deletes the record from entry from the table indexed by the computed value.

Hashing Requirements

- **Uniformity**: The hashing function should distribute every key equally likely in the range space.
- **Low cost**: Cost of executing hashing function should small.
- **Determinism**: For a given input same hash value must be generated by a hash function.

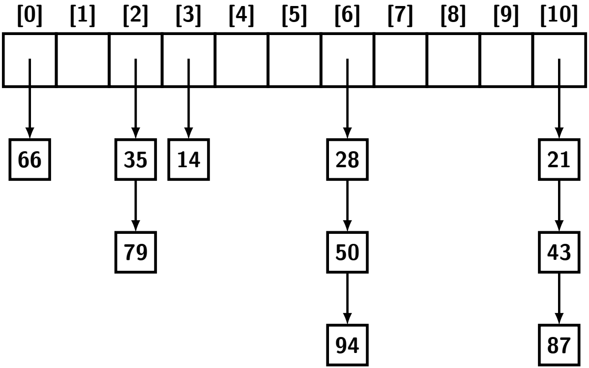
Types of Hashing

- Hash table basically stores an array of pointers to actual records.
- A NULL pointer means no record key mapped to the table entry.
- Two types of hashing:
  - **Open hashing** or **Separate chaining** and
  - **Closed hashing** or **Open addressing**.

Common Hash Functions

- Division method.
- Multiplication method.
- Mid square method.
- Folding method.

Division Method



Division Method

- $h(k) = k \mod m$.
- If $m = 2^p$, using hash function $\mod$ would map any k to its lower order p bits.
- In fact, any key of the form $k = (am + x)$ would map to $h(x)$, even if m is prime.

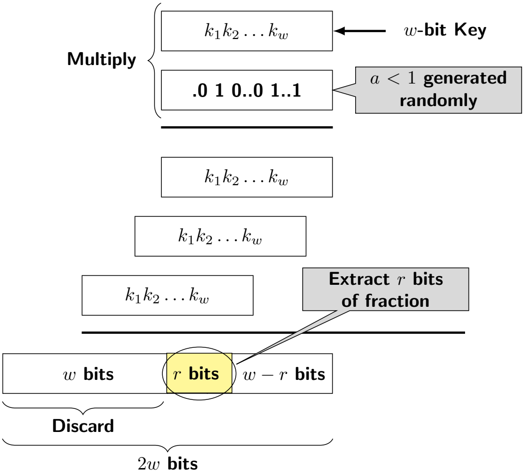
Division Method

Let the base of number system be b and $b \equiv 1 \pmod m$:

$$\begin{aligned} k \mod m &= \left(\sum_0^r b^i k_i\right) \mod m \\ &= \left(\sum_0^r (qm+1)^i k_i\right) \mod m \\ &= \sum_0^r k_i \mod m \end{aligned}$$

- Which means division function is bad.
- If m is prime not close to 2^p or 10^p ($b = 10$) then its ok in practice.

Multiplication Method



$$h(k) = \lfloor m.(k.a \mod 1) \rfloor$$

Multiplication Method

- This hash is random, because the middle bits of the result of multiplication depends on all bits of key.
- Optimal choice of a depends on keys.
- Consider the following example:
  - Let $m = 100$, $a = 1/3$.
  - For $k = 10$, $\lfloor 100 * (10 * 0.33...) \rfloor = 33$.
  - For $k = 11$, $\lfloor 100 * (11 * 0.33...) \rfloor = 66$
  - For $k = 12$, $\lfloor 100 * (12 * 0.33...) \rfloor = 99$
- Knuth claims a good choice is: $a \approx \left(\sqrt{5} - 1\right)/2$.

Comparison of Two Methods

m = 1000 $\qquad$ m = 1000

a = 0.618033988749895

key	$h(k) = \lfloor (m * (k * a \mod 1)) \rfloor$	$h(k) = k \mod m$
123456	4	456
123459	858	459
123496	725	496
123956	21	956
129456	208	456
193456	383	456
923456	195	456

Clearly, multiplication function distributes keys more evenly.

Mid-square Method

- Squares the key value and extracts same middle r values.
  - If $k = 1234$, then $k^2 = 1522756$.
  - Let table size = 100, we extract middle 2 digits $h(k) = 27$.
  - In above example we always choose 3rd and 4th digit from right.
- Like multiplication method middle r digits depend on most or all digits of the original key.

Folding Method

- Divide the key into a number of parts of equal lengths $k_1, k_2, \ldots k_p$.
- Only k_p may have less number of digits.
- Add up the parts, and ignore the last carry.
- Suppose we have 100 as table size and have following keys: 5678, 345 and 568901.
  - Parts of 5678: 56 and 78 $\implies$ 56+78=134, ignore carry, $h(5678) = 34$.
  - Parts of 345: 34 and 5 $\implies$ 34+5 = 39, so $h(345) = 39$.
  - Parts of 568901: 56, 89 and 01 $\implies$ 56+89+01 = 146, so ignore carry, $h(568901) = 46$.