Properties of a hash function

- Must provide as distinct indices for distinct keys as possible. That is the number of collisions should be kept as minimum as possible.
- Fast computable.
- Distributes the keys evenly (unbiasness) in the container or table for holding the keys.
- Each key has equal probability to be mapped to any position.
- Usually range of keys not known a priori.
- However, the actual number of keys expected to be stored at any time is significantly smaller than the range.
- Therefore, even with a small table size, it is possible to perform all the required dictionary operations, provided the keys are distributed evenly.

Mod function

- Simplest function is $h(x) = x \bmod M$, where M is the table size.
- But what is characteristics of M?
- An even number? Fine if the sample key values do not have non even distribution of even and odd values.
- Power of 2? Then only k least significant bits of key value participate in generating hash value.
- If $r \equiv 1 \bmod M$, where b is the radix for number x then M is not a good choice.
- In this case $r = qM + 1$ where q is an integer.
- Therefore

$$\begin{aligned} x \bmod M &= (\textstyle\sum_1^k (qM+1)^i x_i) \bmod \mathrm{M} \\ &= (\textstyle\sum_1^k x_i) \bmod M \end{aligned}$$

Mod function

- Since, x mod M becomes equal to sum of individual digits of x modulo M.
- Therefore, all mumbers which are permutation of same set of digits give same hash value.
- Similarly, we can prove that if $r^k \equiv 1 \bmod M$ then $h(x)$ becomes equal to modulo M sum of the blocks of k characters each from x.
- Which means numbers which consists of permuation of same set of k blocks of characters give same hash value.
- In general if $r^k \equiv \pm a \bmod M$ for small a then it results in combination of individual characters or groups of characters that can produce a number of collisions.
- Therefore, the best choice for M is a prime number.

Middle of square

- Key is squared and a few middle bits extracted.
- Middle square function assumes table size $M = 2^k$, for $k \geq 1$.
- The hash function is
$$h(x) = \left\lfloor \frac{M}{W}(x^2 \bmod W \right\rfloor$$
where $W = 2^w$, w being a word size.
- Obviously, M is less than W.

Middle of square

- The fraction $\dfrac{M}{W} = 2^{k-w}$.
- Therefore, right shifting $x^2 \bmod W$ by $w-k$ bits it the multiplication in hash function is possible.
- Essential idea of right shifting is to extract bits from middle by ignoring $w-k$ least significant bits of x^2.
- Right shifting fills the leading bits by 0s, therefore, the hash value is determined by k least significant bits, or equivalently varies from 0 to M.

Multiplication method

- The function is given by

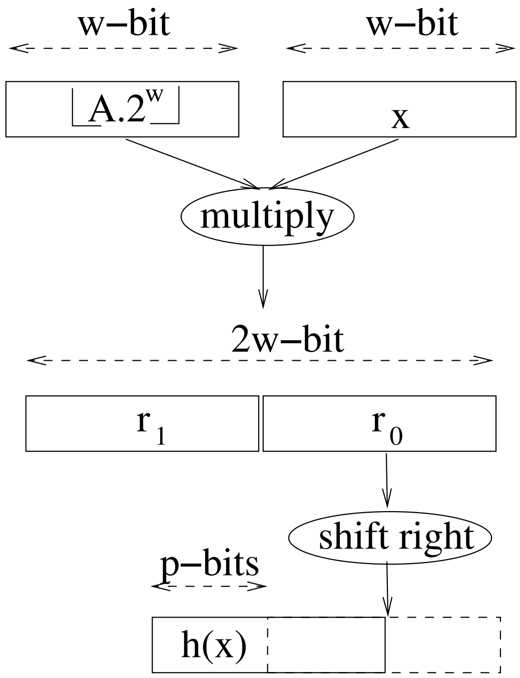
$$h(x) = \lfloor M \times (ax - \lfloor ax \rfloor) \rfloor$$

  where a is a constant between 0 and 1.
- Since M is multiplied by a fraction the hash value is always between 0 and $M - 1$.
- Choice of M is not critical, though we let it as 2^p for some integer p.
- It helps in implementation.

Multiplication method: implementation

- To get the fractional part of product,
  1. multiply the constant a by 2^w where, w is the word size.
  2. multiply $a.2^w$ by x which is another w-bit number.
  3. let the product be $r_1 2^w + r_0$, a $2w$-bit number.
  4. now multiply the product by $M = 2^p$, which extracts p most significant bits of r_0.

Multiplication method: implementation



Multiplication method: implementation

- Since two of three multiplications are multiplication by a power of 2 shift operations suffice.
- Since word size is w-bit the only r_0 which forms least significant w-bits of the $2w$-bit product $r_1 2^w + r_0$ is actually stored. This involves a single multiplication step.
- Knuth recommends the value of $a = (\sqrt{5} - 1)/2$, the golden ratio with value approximately equal to 0.6180339887...

Multiplication method: example

- If word size is 8-bit then multiplying 2^8 with 0.6180339887, we obtain a value = 158.21670111.
- Multiplying 158 by a 8-bit number, say $x = 205$, results in 16-bit number 32309, whose 8-least significant bits provide a value=34.
- Assuming table size to be 32, multiply 34 by 32 and extract 5 bits as 17.
- Therefore, 205 mapped to position 17 of hash table.

Universal hash function

- Universal hash function is not just a single function, but a collection of hash function.
- The idea is if a single hash function is used then an adversary can just supply n keys such that all these hash into same slot of the table.
- Then all the dictionary operations we wish to do will need $\Theta(n)$ time in average per operation.
- This not a good scenario. We must be able to get around this problem. How?
- Universal hash function provides an answer to this problem.

Universal hash function

- The approach is to use a hash function at random from a set of possible hash functions.
- All the hash functions should be designed with care as explained earlier.
- Since each execution employs a different hash function the algorithm behave differently from one execution to the other.
- The poor performance can result if we choose a hash function from the set which maps large number of keys to same table slot.
- But the probability of this happening is small and is same for any set of keys of same size.

Universal hash function

- H is a collection of hash functions that map keys into range $\{0, 1, \ldots, M-1\}$.
- The collection is called universal provided the number of hash functions $h \in H$ for which $h(x) = h(y)$ for pair of keys $x \neq y$ is precisely $|H|/M$
- That is by choosing h randomly from H the probability of a clash occurring is same as a clash occurring if $h(x)$ and $h(y)$ are chosen at random from $\{0, 1, \ldots M-1\}$.

Analysis of universal hashing

- Define an independent (why?) random variable R_{xy} as follows:

$$R_{xy} = \begin{cases} 1, & \text{if } x \text{ and } y \text{ collide using } h \\ 0, & \text{otherwise} \end{cases}$$

- Since probability of a single collision is $1/M$ we have

$$\begin{aligned} E(R_{xy}) &= 0.Pr[h(x) \neq h(y)] + 1.Pr[h(x) = h(y)] \\ &= Pr[h(x) = h(y)] \\ &= 1/M \end{aligned}$$

- Let R_x denote the total number of collisions involving a single key x.

$$\begin{aligned} E(R_x) &= E(\textstyle\sum_{i=1}^{n-1} R_{xy_i}) \\ &= \textstyle\sum_{i=1}^{n-1} E(R_{xy_i}) \\ &= \frac{n-1}{M} \end{aligned}$$

Designing universal hash function

- Choose M as a prime number.
- Decompose each key x into $n+1$ fixed width bit strings or characters i.e., $x = < x_0, x_1, \ldots x_n >$ such that $|x_i| < M$ for each i.
- Let $a = \{a_0, a_1, \ldots, a_n\}$ denote a sequence of $n+1$ elements chosen randomly from set $\{0, 1, \ldots, M-1\}$.
- Define $h_a(x) = \sum_{i=0}^{n} a_i x_i \bmod M$.
- Define $H = \bigcup_a \{h_a\}$.

Proof for validity

- Consider a pair of distinct keys x and y and assume that $x_0 \neq y_0$.
- We can provide same argument for $x_i \neq y_i$ for $0 < i < n+1$.
- For any fixed values of $a_1, a_2, \ldots, a_n$ there is exactly one value a_0 that satisfies $h(x) = h(y)$.
- This a_0 is a solution to

$$a_0(x_0 - y_0) \equiv -\sum_{i=1}^{n} a_i(x_i - y_i)(\text{mod } M)$$

- There is a result that if for $M > 1$ and $\gcd(M, a_0)$=1 then the equation

$$ax \equiv b(\text{mod } M)$$

has a unique solution.

Proof for validity

- We have chosen M to be prime, therefore, gcd(M, a_0)=1.
- So, a_0 is a unique solution to the above equation.
- We have proved the result for just one fixed combination of values $a_1, a_2, \ldots, a_n$.
- There are M^r possible combination of values for $a_1, a_2, \ldots, a_n$.
- Therefore keys x and y can collide with a probability of $M^n/M^{n+1} = 1/M$.
- This proves that the hash function stated above is universal.