Hash Functions

A good hash function satisfies the following properties:

1. Universal hash function
   \[ f(x) = \text{integer} \]

2. Distribution method

Assumption: Keys are random numbers (\( K \))

Example: Universal hashing

Assumption: Simple uniform hashing assumption

\[ f(x) = \text{integer} \]

where \( k < m \), \( k \geq 0 \), \( m > 0 \), \( x \in \{0, 1, \ldots, m-1\} \), \( P(x) \) is the probability that \( x \) is chosen.

\[ \sum_{x=0}^{m-1} P(x) = 1 \]

\[ P(0) = \frac{1}{m} \]

\[ \sum_{x=1}^{m-1} P(x) = \frac{m-1}{m} \]

Conclusion:

- If \( k \) is large enough, we can approximate the uniform distribution.
- The probability of any hash function \( f(x) \) being a good hash function is high.

\[ f(x) = k \]
Values of $m$ to avoid: $m$ power of 2, $m = \Phi (m)$.

If $m = 1$, then $\Phi (m) = 1$.

We have $\Phi (m) = n$, where $n = 2^k$, $k \geq 1$. If $k$ is an odd integer, then $\Phi (m) = n$.

Example: $\Phi (12) = \Phi (2) \cdot \Phi (3) = 2 \cdot 2 = 4$.

Example: $\Phi (12) = \Phi (2^2 \cdot 3) = \Phi (2^2) \cdot \Phi (3) = 2 \cdot 2 = 4$.

If $m = \Phi (m)$, then $m$ is a prime number.
Multiplication method: an example

Suppose $y = 1234.6 \div 2.0000 = 0.618039887$

Let: $y(k) = \lfloor y \mod (1/k) \rfloor$

Do exercise 12.3.4

Kathre Shigrefe upp $A$ $\approx$

* available in most hardware
* a simple list of $p$ positions, $p$ by $p$ multiplication instructions
* choose $m = n$ for a fixed number of words corresponding
* We can choose $m$ to make $n$-function easy to implement
* value of $m$ is not critical

Characteristics

2. then multiply value by $m$ and take $\lfloor y \mod (1/k) \rfloor$
1. multiply $b$ by $k$ and extract the $\lfloor / k \rfloor$

Two steps

Use: $y(k - 1) = \lfloor y \mod (1/k) \rfloor$, where $y \mod (1/k) = y

Multiplication method

Steps

0. Then multiply by \( k \) corresponding to taking the $p$
0. $0 \leq \{y \mod \lfloor 1/k \rfloor \}$ each of $y$ is $\leq k$

\( y(k) \) and $y(k) \div y(k) \) a word of

Multiplication method: an implementation (1)

Multiplication method: an implementation (II)
Universal hashing: phenomenon

The sequence is called universal hashing

No matter what keys are chosen by adversary

Independent of keys to be stored

Choose a hash function that is random

Key out?

Any fixed hash function is vulnerable

Avoidance method: universal hashing

Universal adversary chooses the keys to be hashed

Universal hashing: preimage

Theorem 12.5 If \( f \) is chosen from a universal collection of hash functions and is used to hash a key \( x \) into a bucket of size \( m \), then the expected number of hashing operations to find the bucket of \( x \) is exactly

\[
\frac{1}{n} \sum_{i=1}^{n} \frac{1}{i} \cdot \frac{1}{m}
\]

where \( n \) is the expected number of keys hashing into a bucket and is used to hash into a bucket of size \( m \).
Example of a universal class of hash functions

Implementation

- Choose \( m \), table size, to be a prime number.
- \( \ell = \lceil \log m \rceil \) is chosen randomly from \( (0, 1, \ldots, m - 1) \).
- Choose \( (r + 1) \alpha's \), each \( \alpha \), chosen randomly from \( (0, 1, \ldots, m - 1) \).
- \( \ell \) and \( \ell \) are fixed.
- When user gives the key \( x \), decompose \( x \) in \( r + 1 \) parts of \( x = (x_0, x_1, \ldots, x_r) \).
- Compute \( h_\ell (x) = \sum_{i=0}^{r} \alpha_i \mod m \).
- We can prove that \( H = \bigcup_{j=1}^{\ell} H_j \) is a universal class of hash functions.
- Since there are \( m^r \) possibilities for \( a \), there are \( m^r \) numbers in the class.

Number of collisions depends on parameters.

- Load factor: \( \ell \approx \frac{m}{\ell} \).
- \( \ell \approx \frac{m}{\ell} \).
- \( \ell \approx \frac{m}{\ell} \).

Open addressing

- Each bucket can contain only one element.
- All elements are stored in hash table.

Illustration of a universal hash function

- Choose \( \ell \) and \( \ell \) are fixed.
- \( \ell \) and \( \ell \) are fixed.
- \( \ell \) and \( \ell \) are fixed.
- Some of the \( \ell \) can be the same.
- Choose \( x \) and \( x \) are distinct keys.

Proof

- \( \ell \) is a universal class of hash functions.

\[ \{ \langle h, y \rangle \} = H \]
Assumption: elements are keys with no duplicate data.

Insertion in open addressing

Each slot contains a Key or NIL

\[ \text{Assumption: elements are keys with no duplicate data.} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]

\[ \text{Insertion in open addressing} \]