Hashing - 1

Hash Functions
Sections 5.1 and 5.2
Introduction

- Data items stored in an array of some fixed size.

- Search performed using some part of the data item – called the **key**.

- Hash function
  - Maps keys to integers (which represent table indices)
    - $\text{Hash(Key)} = \text{Integer}$

  - Evenly distributed index values
    - Even if the input data is not evenly distributed
Simple Hash Functions

Assumptions:
- $K$: an unsigned 32-bit integer
- $M$: the number of buckets (the number of entries in a *hash table*)

Goal:
- If a bit is changed in $K$, all bits are equally likely to change for $\text{Hash}(K)$
A Simple Hash Function...
A Simple Hash Function...

What if

\( \text{Hash}(K) == K \)
A Simple Hash Function...

- What if
  - $\text{Hash}(K) == K$
- What is wrong?
A Simple Hash Function...

- What if
  - $\text{Hash}(K) == K$
- What is wrong?

- Values of $K$ may not be evenly distributed
  - But $\text{Hash}(K)$ needs to be evenly distributed
Another Simple Function
Another Simple Function

If

Hash(K) == K % M
Another Simple Function

- If
  - Hash(K) == K % M

- What is wrong?
Another Simple Function

If

Hash(K) == K % M

What is wrong?

Suppose

M = 10,
K = 2, 20, 34, 42, 76
Another Simple Function

- If
  - $\text{Hash}(K) \equiv K \mod M$
- What is wrong?
- Suppose
  - $M = 10$,  
  - $K = 2, 20, 34, 42, 76$
- Then $K \mod M = 2, 0, 4, 2, 6, \ldots$
  - Since 10 is even, all even $K$ are hashed to even numbers ...
Yet Another Simple Function
Yet Another Simple Function

If

\[ \text{Hash}(K) = K \mod P \], with \( P \) a prime number
Yet Another Simple Function

- If
  - $\text{Hash}(K) = K \mod P$, with $P$ a prime number
- Suppose
  - $P = 11$
  - $K = 10, 20, 30, 40$
Yet Another Simple Function

- If
  - \( \text{Hash}(K) = K \mod P \), with \( P \) a prime number

- Suppose
  - \( P = 11 \)
  - \( K = 10, 20, 30, 40 \)
  - \( K \mod P = 10, 9, 8, 7 \)
Yet Another Simple Function

- If
  - $\text{Hash}(K) = K \mod P$, with $P$ a prime number

- Suppose
  - $P = 11$
  - $K = 10, 20, 30, 40$
  - $K \mod P = 10, 9, 8, 7$

- More uniform distribution...
Hashing a Sequence of Keys

- $K = \{ K_1, K_2, \ldots, K_n \}$
- E.g., $\text{Hash} \left( \text{“test”} \right) = 98157$
- Design Principles
  - Use the entire key
  - Use the ordering information
Use the Entire Key

```c
unsigned int Hash(const char *Key) {
    unsigned int hash = 0;
    for (unsigned int j = 0; j < K; j++) {
        hash = hash ^ Key[j] // bitwise xor
    }
    return hash;
}
```

**Problem:** $\text{Hash(“ab”)} = \text{Hash(“ba”)}$
unsigned int Hash(const char *Key) {
    unsigned int hash = 0;
    for (unsigned int j = 0; j < K; j++) {
        hash = hash ^ Key[j];
        hash = hash ^ (j%32);
    }
    return hash;
}
Better Hash Function

```c
unsigned int Hash(const String& S)
{
    unsigned int i;
    long unsigned int bigval = S.Element(0); // S[0]

    for (i = 1; i < S.Size(); ++i)
        bigval = ((bigval & 65535) * 18000) // low16 * magic_number
                + (bigval >> 16) // high16
                + S[i];

    bigval = ((bigval & 65535) * 18000) + (bigval >> 16); // bigval = low16 * magic_number + high16
    return bigval & 65535; // return low16
}
```