字符串的哈希查找

〖Example〗 In Oxford English dictionary

name = **since**

attribute = **a list of meanings**

- **M[0] = after a date, event, etc.**
- M[1] = seeing that (expressing reason)
- ....

〖Example〗 In Oxford English dictionary

name = **since**

attribute = **a list of meanings**

- M[0] = after a date, event, etc.
- M[1] = seeing that (expressing reason)
- ....

〖Example〗 In a symbol table for a compiler

name = identifier (e.g. int)

attribute = a list of lines that use the identifier, and some other fields

Properties of f:

① $f(x)$必须能够计算任意关键字且冲突最少

② $f(x)$ 应该均匀分布，比如对任意 x和 i, 有如下概率 Probability($f(x) = i$) = $1 / b$. 这种函数函数叫（均匀分布哈希函数） uniform hash function.

Properties of *f*:

① *f*(*x*)必须能够计算任意关键字且冲突最少

② *f*(*x*) 应该均匀分布，比如对任意 *x*和 *i*, 有如下概率
Probability(*f*(*x*) = *i*) = 1 / *b*. 这种函数函数叫（均匀分布哈希函数） uniform hash function.

```
f ( x ) = x % TableSize ;  /* if x is an integer */
```

☹ What if *TableSize* = 10 and *x*' s are all end in zero?

☺ *TableSize* = prime number ---- good for random integer keys

如何获取字符串的x?

§2 Hash Function

$$f(x) = (\Sigma\, x[i])\ \%\ TableSize\ ;\quad /*\ \text{if } x \text{ is a string } */$$

〖Example〗 *TableSize* = 10,007 and string length of $x \le 8$.

If $x[i] \in [0, 127]$, then $f(x) \in [0,$ 1016 $]$ ☹

$f(x) = (\Sigma\, x[i])\ \%\ TableSize$; /* if x is a string */

〖Example〗 *TableSize* = 10,007 and string length of $x \le 8$.

If $x[i] \in [0, 127]$, then $f(x) \in [0,$ 1016 $]$ ☹

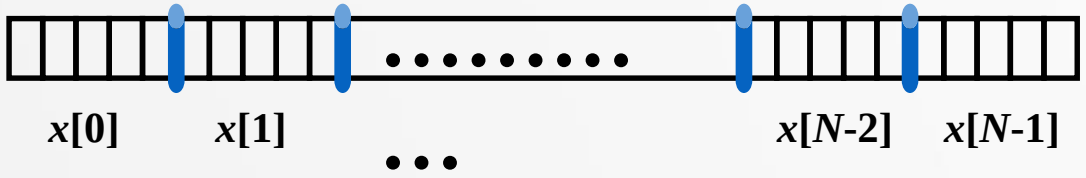
$f(x) = (x[0] + x[1]*27 + x[2]*27^2)\ \%\ TableSize$;

Total number of combinations = 26*272 = 18,954

☹ Actual number of combinations < 3000

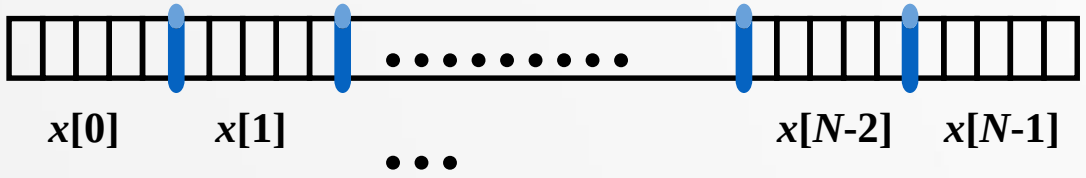
§ 2 Hash Function

$$f(x) = (\Sigma\, x[N-i-1] * 32^i)\ \%\ TableSize\ ;$$



§ 2 Hash Function

$$f(x) = (\Sigma\, x[N-i-1] * 32^i) \;\%\; TableSize\;;$$




```
Index Hash3( const char *x, int TableSize )
{
                unsigned  int  HashVal = 0;
/* 1*/          while( *x != '\0' )
/* 2*/              HashVal = ( HashVal << 5 ) + *x++;
/* 3*/          return HashVal % TableSize;
}
```

Faster than *27

Carefully select some characters from *x*.

☹ If *x* is too long (e.g. street address), the early characters will be left-shifted out of place.