09-B2

9.词典

散列函数

更多

邓俊辉

deng@tsinghua.edu.cn

有意整齐与有意变化，皆是一方死法。

（伪）随机数法

❖ （伪）随机数发生器

循环：rand(x + 1) = [a × rand(x)] % M //M素数，a % M ≠ 0

a = 7^5 = 16,807 = 10000110100111_b

M = 2^{31} − 1 = 2,147,483,647 = 01111111 11111111 11111111 11111111_b

❖ （伪）随机数法

径取：hash(key) = rand(key) = [rand(0) × a^{key}] % M

种子：rand(0) = ?

❖ 把难题推给伪随机数发生器，但是...

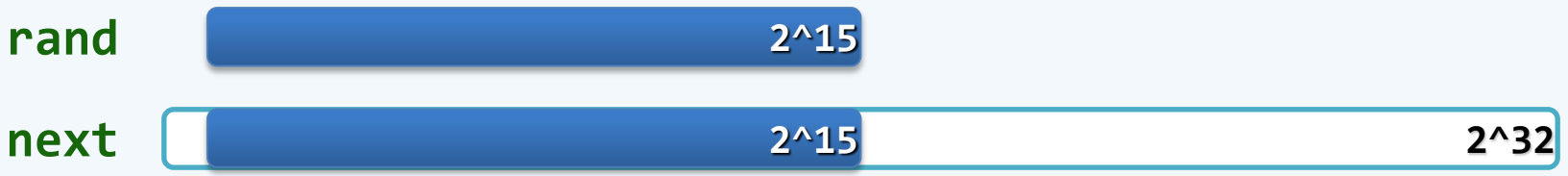
❖ （伪）随机数发生器的实现，因具体平台、不同历史版本而异

创建的散列表可移植性差——故需慎用此法！

（伪）随机数法

❖ unsigned long int next = 1; //The C Programming Language (2nd edn), p46

```
void srand(unsigned int seed) {  next = seed;  }
int rand(void) { //1103515245  =  3^5 * 5 * 7 * 129749
    next = next * 1103515245 + 12345;
    return (unsigned int)(next/65536) % 32768;
}
```



❖ 另类随机数（串）算法

```
int rand() { int uninitialized; return uninitialied; }
char* rand( t_size n ) { return ( char* ) malloc( n ); }
```

❖ 以上方法，可否用于散列？

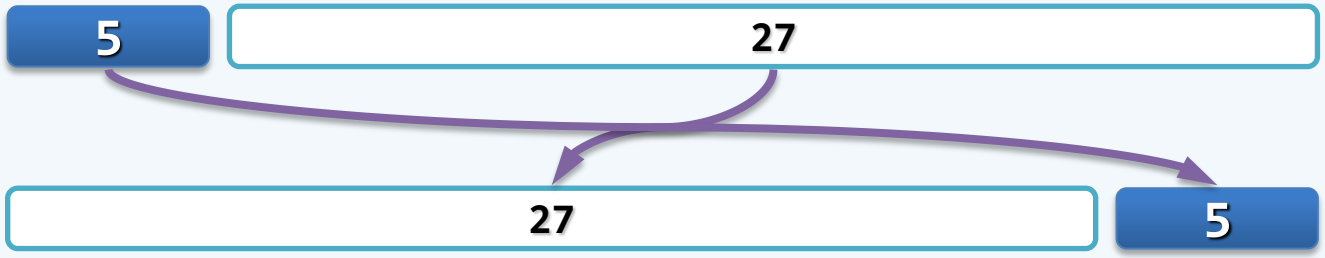
多项式法

❖ hash(s = $x_0\ x_1\ \ldots\ x_{n-1}$) = $x_0 \cdot a^{n-1} + x_1 \cdot a^{n-2} + \ldots + x_{n-2} \cdot a^1 + x_{n-1}$

= $(\ \ldots\ (\ (\ (\ x_0 * a\ + x_1\ )\ * a\ + x_2\ )\ * a\ + \ldots\ x_{n-2}\ )\ * a\ + x_{n-1}$

```
❖ static size_t hashCode( char s[] ) { //近似多项式，但更快捷
    unsigned int h = 0;
    for ( size_t n = strlen(s), i = 0; i < n; i++ )
       { h = (h << 5) | (h >> 27); h += ( int ) s[i]; }
    return ( size_t ) h;
  } //有必要如此复杂吗？
```

❖ 能否使用更简单的散列，比如...



多项式法

❖ 字符分别映射为数字：f(c) = CODE(UPPER(c)) - 64　　//['A', 'Z'] ~ [1, 26]

再简单相加：hash(s) = $\sum_{c \in s} f(c)$

hash ~ 8 + 1 + 19 + 8 = 36

❖ 字符相对次序信息丢失，将引发大量冲突

I am Lord Voldemort

Tom Marvolo Riddle

❖ 即便字符不同、数目不等...

He's Harry Potter

❖ Key to improving your programming skills

Learning Tsinghua Data Structures & Algorithms

Java:hashCode()

- hashCode()方法

  适用于Java中的所有对象

  将任意类的对象转换为（32位int型）整数

  对于非整型的key，先转换为整数（散列码），然后再做散列

- hashCode()：效果如何？效率如何？是如何实现的？

- object.hashCode() = object在内存中的地址

- 问题：相关的对象地址也相近，冲突概率高 //更糟糕的是...

- 散列码与对象的内容无关

  比如，完全相等的两个字符串对象，散列码居然不同

- 有何替代方案？