9.词典

（b）散列：原理

邓俊辉

deng@tsinghua.edu.cn

服务 ~ 电话

General inquiries

Tel: Toll Free: 1-800-IBM-4YOU
E-mail: askibm@vnet.ibm.com
www.ibm.com/us/en/

Shopping

Tel: Toll Free: 1-888-SHOP-IBM

Sales Center

1-855-2-LENOVO (1-855-253-6686)
Mon - Fri: 9am-9pm (EST)
Sat - Sun: 9am-6pm (EST)

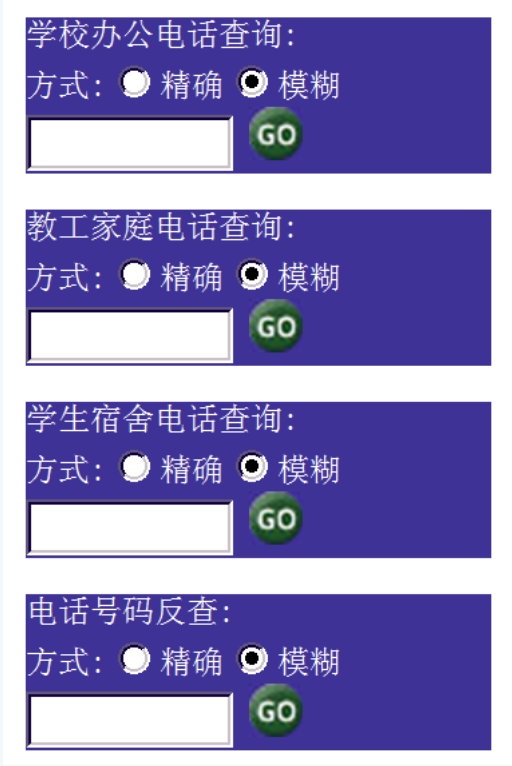
Customer Service

1-855-2-LENOVO (1-855-253-6686)
Mon - Fri: 9am-9pm (EST)
Sat - Sun: 9am-6pm (EST)

85001

- TV channel → s/n

CCTV-1 [1]	BJTV-1 [21]	CETV-1 [31]
CCTV-2 [2]	BJTV-2 [22]	CETV-2 [32]
CCTV-3 [3]	BJTV-3 [23]	CETV-2 [33]
... ...	... ...	... ...

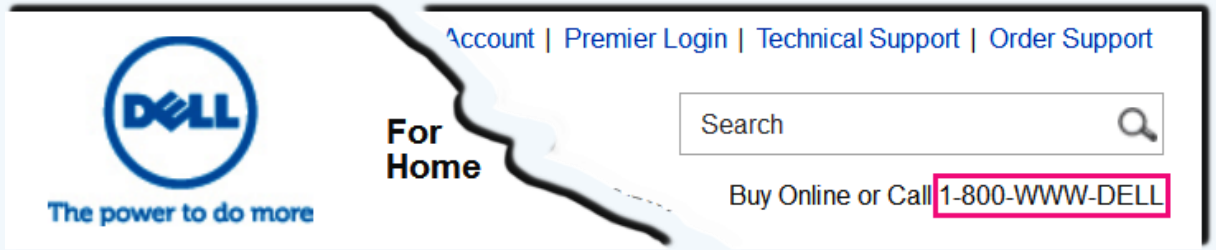


- http://85001.tsinghua.edu.cn/

```
owner = phonebook.get(number)
```

- 动态性：每天的电话簿都是不同的

```
phonebook.put(number, owner)
phonebook.remove(number)
```



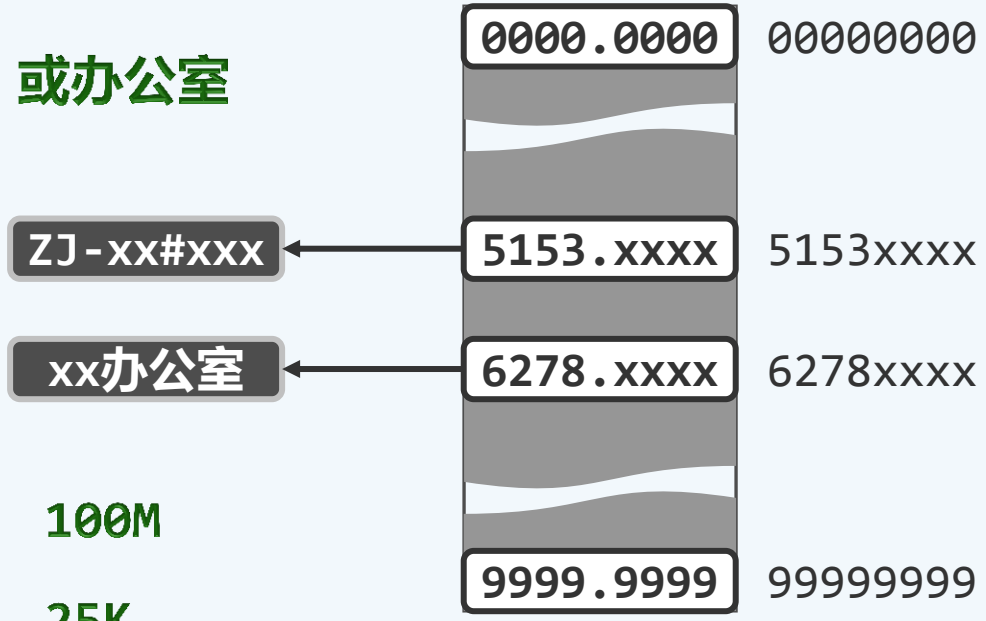
电话簿

- 需求： 为一所学校制作电话簿

  号码 ~ 个人（教员、学生、员工）或办公室

- 蛮力： 使用数组，按电话号码索引

  时间 = O(1)



- 以清华为例（2003）

  #可能的电话 = R = 10^8 = 100M

  #实有的电话 = N = 25,000 = 25K

- 问题

  空间 = O(R + N) = O(100M + 25K)

  效率 = 25K / 100M = 0.025%

  其它方面...

IP Dictionary

- IP → Domain name + Host info

  C-Term WRY.dat

  http://www.ip138.com

- IPv4 (32bit) → IPv6 (128bit)

  128-bit

  32-bit

  R = #可能的IP = 2^128 = 256 × 10^36

  N = #实际的IP = 2^32 = 4 × 10^9

  N / R = 0.000 000 000 000 000 000 000 000 000 000 015 625

- 事实：实际的词条数N << 可能的词条数R

  //相对于THU电话簿，此时的N与R相差更为悬殊

- 如何在保持查找速度的同时，降低存储消耗？

原理

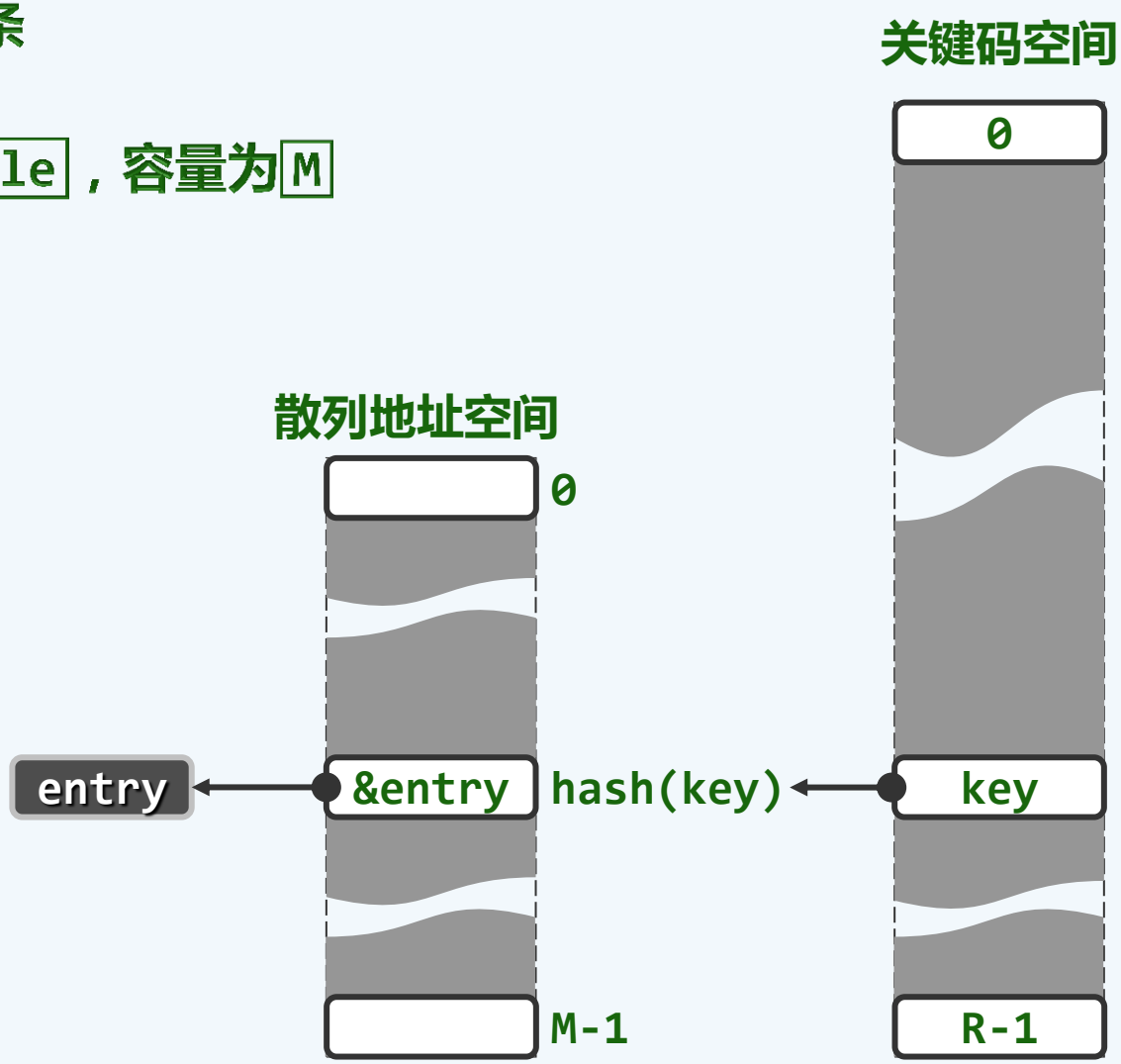
- 桶 bucket：直接存放或间接指向一个词条
- 桶数组 bucket array/散列表 hash table，容量为 M

  $N < M << R$

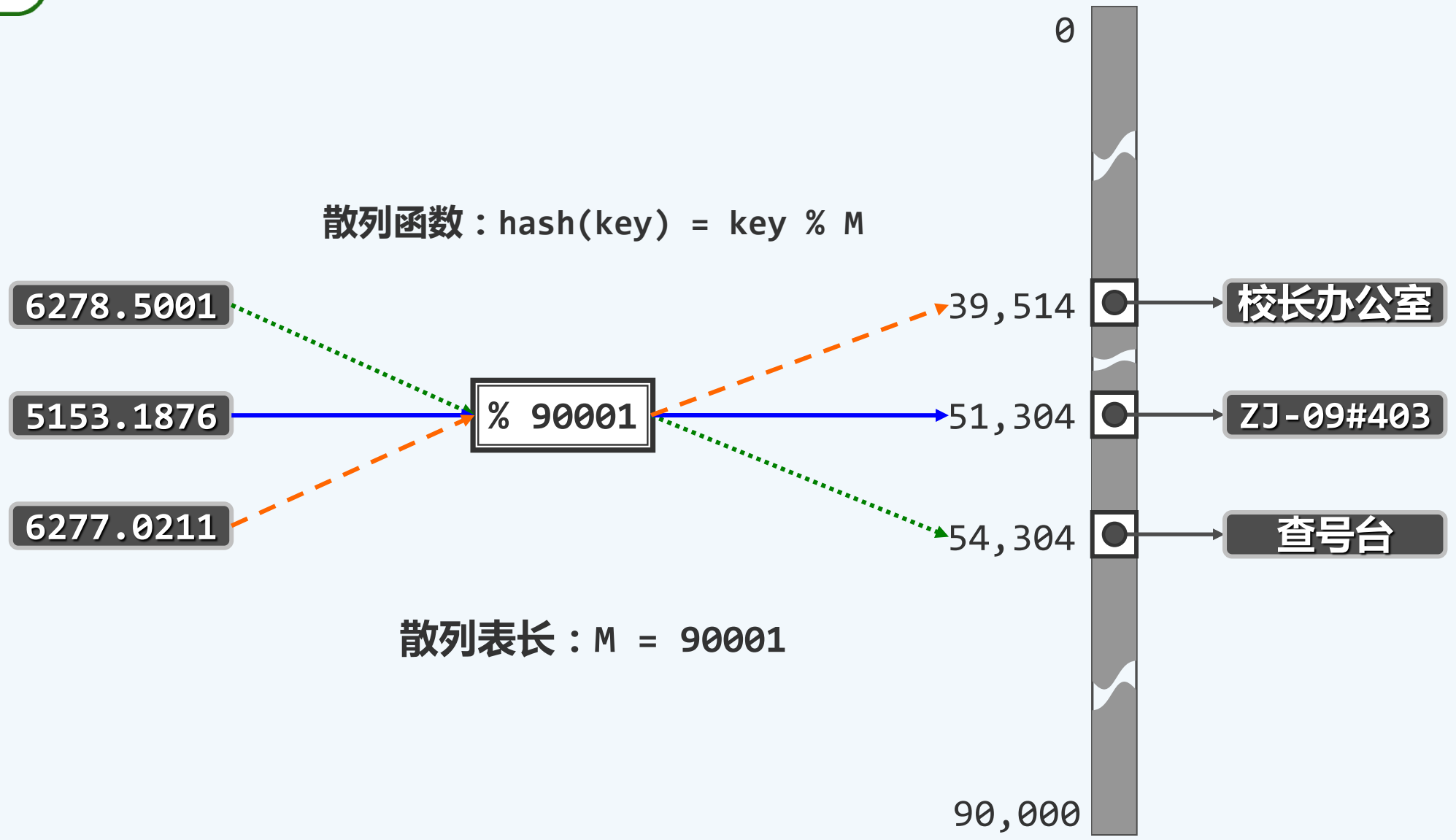
  空间 = $O(N + M) = O(N)$
- 定址/杂凑/散列：

  根据词条的key（未必可比较）

  直接确定散列表入口
- 散列函数：hash() : key ⟼ &entry
- 直接 = expected-$O(1)$ ≠ $O(1)$



实例


0
散列函数：hash(key) = key % M
6278.5001
5153.1876
6277.0211
% 90001
39,514
51,304
54,304
90,000
校长办公室
ZJ-09#403
查号台
散列表长：M = 90001

时空效率

- 定址：通过散列函数，将任一key映射至[0, M)

  除余法：hash(key) = key % M　　//例：M = 90001

  定址效率：基于取模运算——常数时间！　　//机器字长无限制？

- 无论散列表多大，定址、查询、插入和删除均只需expected-O(1)时间

- 装填因子load factor

  λ = N / M = #存放的词条 / |桶数组|　　//λ，选多大才合适？

- λ越大，空间利用率越高

  当然，λ不可能超过100%　　//否则，根据鸽巢原理...

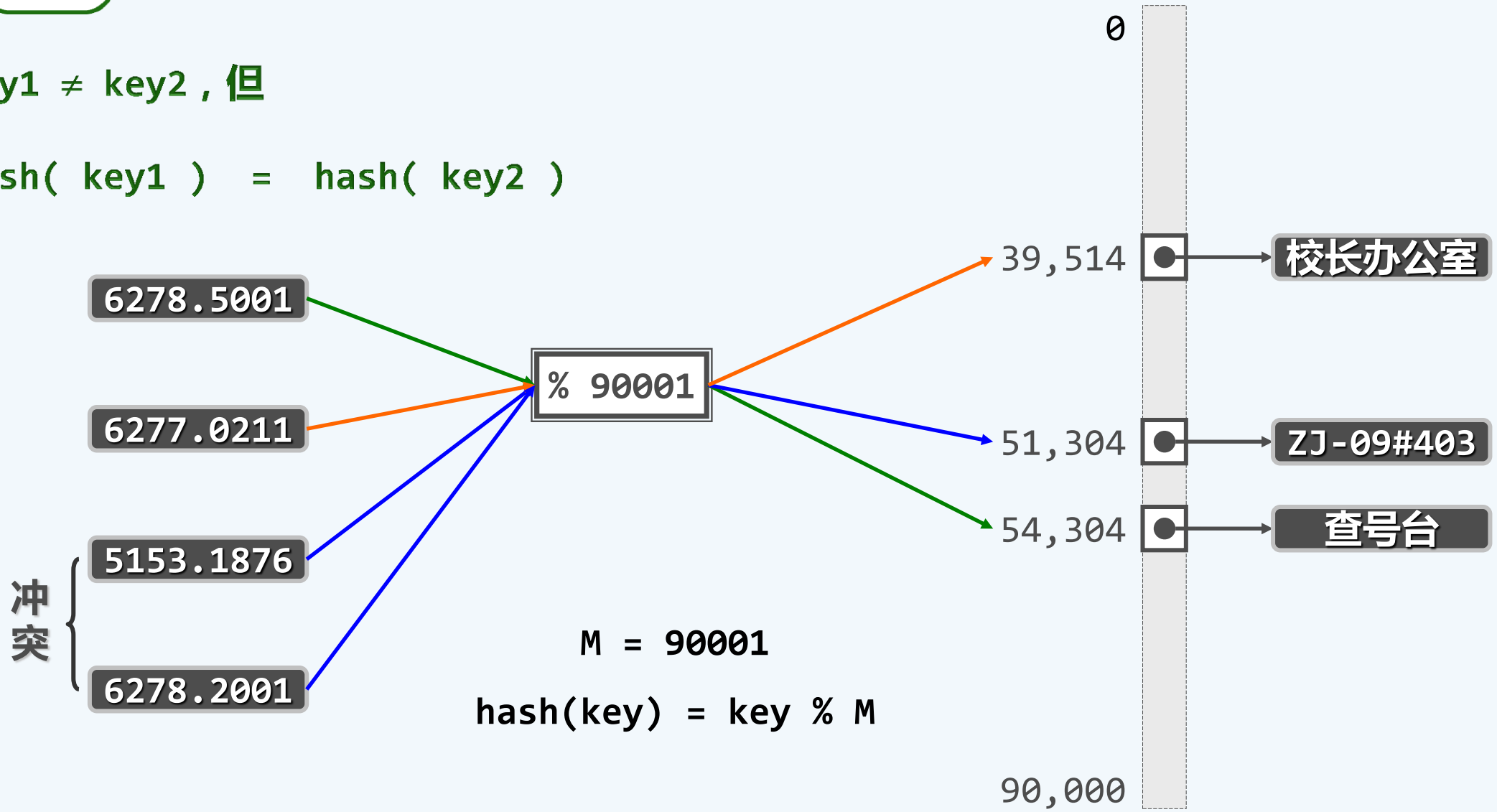
  反之，是否只要λ ≤ 1就行了？　　//比如，就取λ = 1

- 实际上，即便λ << 1，依然会有问题...

冲突

❖ key1 ≠ key2，但

hash(key1) = hash(key2)



完美散列

- 是否存在某种定址方法，能保证不出现冲突？

  亦即，散列函数等效于一个单射injection？

- 在关键码满足某些条件时，的确可以实现单射式散列，比如...

- 对已知且固定的关键码集（比如CD）

  可实现完美散列perfect hashing

  采用两级散列模式

  仅需O(n)空间

  关键码之间互不冲突

  即便在最坏情况下，查找时间也不过O(1)时间

- 不过，在一般情况下，完美散列无法保证存在...

生日悖论

❖ 将在座同学（对应的词条）按生日（月/日）做散列存储

散列表长固定为 M = 365，装填因子 = 在场人数N / 365

❖ 冲突（至少有两位同学生日相同）的可能性P365(n) = ？

//概率论与数理统计讲义第一章，清华大学数学系王晓峰

$P_{365}(21) = 44.4\%$, $P_{365}(22) = 47.6\%$, ..., $P_{365}(23) = 50.7\%$ // 23/365 = 6.3%

❖ 100人的集会：$1 - p_{365}(100) = 0.000,031\%$

自7岁起，不吃不喝、无休无息，每小时参加四次

到100岁，才有可能遇到一次没有冲突的集会

❖ 因此，在装填因子确定之后，散列策略的选取将至关重要，散列函数的设计也很有讲究...