本节主题：

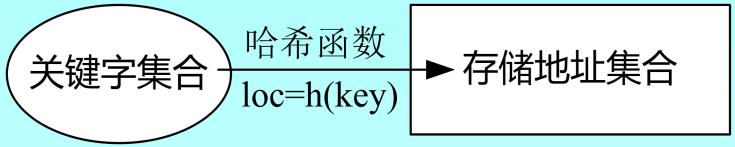
哈希表——散列结构

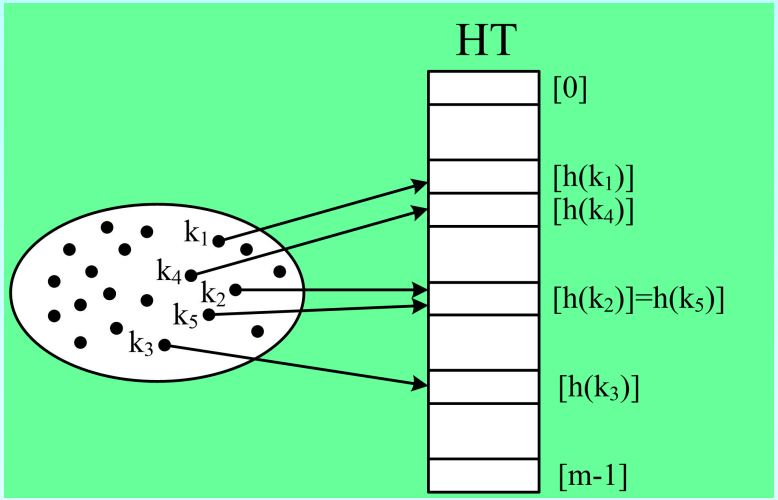
哈希表又称散列表，四大存储结构中的一种

顺序表存储结构　　链接表存储结构　　索引表存储结构　　散列表存储结构

哈希表存储的基本思路

- 哈希函数，散列函数
  - **一个函数h(key)**，用于根据关键字key，计算得到一个内存单元的地址(或称下标)loc
- 哈希表（Hash Table），散列表
  - 要存储的对象个数为n，设置一个长度为m(m≥n)的连续内存单元，将关键字k_i(0≤i≤n-1)存储到地址为h(k_i)的内存单元中。
  - h(k_i)称为**哈希地址/散列地址**
  - 把如此构造的线性表存储结构称为**哈希表/散列表**
- 哈希函数构造方法
  - 直接定址法
  - 除留余数法
  - 数字分析法
  - ……



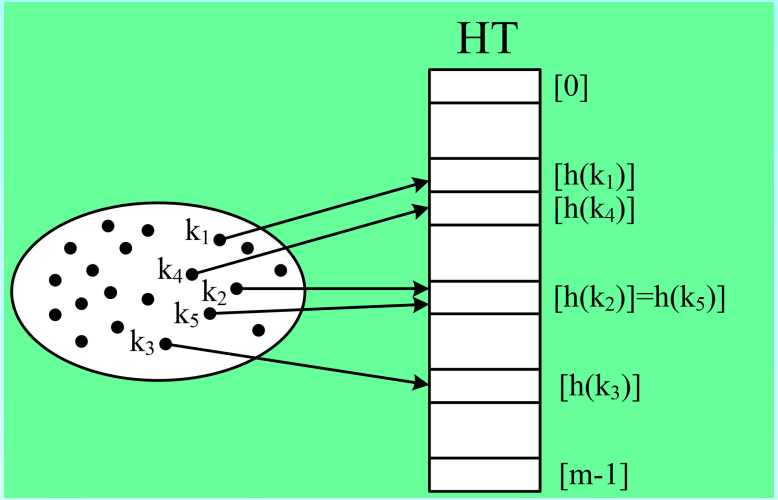


用哈希函数h将关键字映射到散列表中

哈希冲突

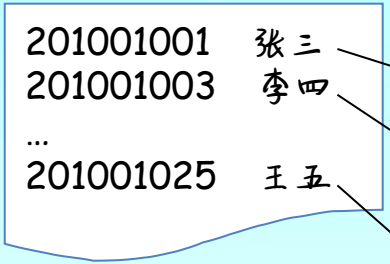
- 术语
  - 对于两个关键字k_i和k_j，$i \neq j$，且$k_i \neq k_j$，但有$h(k_i)=h(k_j)$，这种现象叫做哈希冲突。
  - 这种具有不同关键字而具有相同哈希地址的对象称做“同义词”，引起的冲突称作同义词冲突。
- 冲突中的两难
  - 同义词冲突很难避免，除非关键字的变化区间小于等于哈希地址的变化区间，而这种情况当关键字取值不连续时是非常浪费存储空间的。

    例：取h(i)=i
  - 通常，关键字的取值区间远大于哈希地址的变化区间。
- 冲突解决方法
  - 开放定址法
  - 拉链法
  - ……



哈希函数构造法(1): 直接定址法

- 构造哈希函数的目标
  - 使得到的哈希地址，尽可能均匀地分布在n个连续内存单元地址上，同时使计算过程尽可能简单，以达到尽可能高的时间效率。
- 直接定址法思想
  - 以关键字k本身或关键字加上某个数值常量c作为哈希地址的方法。
- 直接定址法的哈希函数：
  - h(k)=k+c
- 例
  - h(学号)=学号-201001001
- 评价
  - 这种哈希函数计算简单，并且不可能有冲突发生。
  - 适用关键字的分布基本连续的场合
  - 若关键字分布不连续，将造成内存单元的大量浪费。



	学号	其他
[0]	201001001	...
[1]	###	###
[2]	201001003	...
[...]		...
[24]	201001025	...
[...]		...

哈希函数构造法(2): 除留余数法

- 除留余数法思想
  - 用关键字k除以p(某个不大于哈希表长度m的数)， 所得的余数作为哈希地址
- 除留余数法的哈希函数

  h(k)=k mod p （mod为求余运算，p≤m）

  p最好是质数（素数）
- 例

数据表

区号	城市名	说明
010	Beijing	首都
021	Shanghai	直辖市
027	Wuhan	湖北省省会
029	Xian	陕西省省会
025	Nanjing	江苏省省会

h(区号)=VAL(区号) mod 7

区号	010	021	027	029	025
VAL(区号)	10	21	27	29	25
H(key)	3	0	6	1	4

城市哈希表

地址	区号	城市名	说明
0	021	Shanghai	直辖市
1	029	Xian	陕西省省会
2			
3	010	Beijing	首都
4	025	Nanjing	江苏省省会
5			
6	027	Wuhan	湖北省省会

例 除留余数法

- 问题
  - 假设哈希表长度m=13，采用除留余数法哈希函数建立如下关键字集合的哈希表：{16,74,60,43,54,90,46,31,29,88,77}。
- 解：

n=11，m=13

确定除留余数法的哈希函数为：h(k)=k mod p

p的选择：应为小于等于m的素数，假设p取值13。则有：

h(16)=3，h(74)=9，h(60)=8，h(43)=4，

h(54)=2，h(90)=12，h(46)=7，h(31)=5，

h(29)=3，h(88)=10，h(77)=12。

注意：存在冲突。

哈希函数构造法(3): 数字分析法

- 数字分析法思想
  - 提取关键字中取值较均匀的数字位作为哈希地址的方法
- 做法
  - 适合于所有关键字值都已知的情况，并需要对关键字中每一位的取值分布情况进行分析。
- 例如
  - 有一组关键字，如右
  - 分析：每个关键字从左到右的第1、2、3位和第6位取值较集中，不宜作为哈希函数，剩余的第4、5、7和8位取值较分散，可根据实际需要取其中的若干位作为哈希地址。
  - 方案：取最后两位作为哈希地址

学号	其他
92317602	…
92326875	
92739628	
92343634	
92706816	
92774638	
92381262	
92394220	

	学号	其他
…	…	…
[2]	92317602	…
…	…	…
[16]	92706816	…
…	…	…
[20]	92394220	…
…	…	…
[28]	92739628	…
…	…	…

哈希冲突解决

- 冲突与哪些因素有关？
  - 与装填因子有关

$$\text{装填因子 } \alpha = \frac{\text{哈希表中已存入的元素数 } n}{\text{哈希地址空间大小 } m}$$

α 越小，冲突的可能性就越小；
α 越大(最大可取1)，冲突的可能性就越大

α 越小，存储空间的利用率就越低
反之，存储空间的利用率也就越高

对策：
既兼顾减少冲突的发生，又兼顾提高存储空间的利用率
α 一般控制在0.6～0.9的范围内

  - 与所采用的哈希函数有关
    - 若哈希函数选择得当，就可使哈希地址尽可能均匀地分布在哈希地址空间上，从而减少冲突的发生；
    - 若哈希函数选择不当，就可能使哈希地址集中于某些区域，从而加大冲突的发生。
  - 与解决冲突的哈希冲突函数有关
    - 哈希冲突函数选择的好坏也将减少或增加发生冲突的可能性。

哈希冲突解决方法(1): 开放定址法

- 思想
  - 以发生冲突的哈希地址为自变量，通过某种哈希冲突函数得到一个新的空闲的哈希地址

- 线性探测法
  - 线性探测法是从发生冲突的地址（设为d）开始，依次探测d的下一个地址，直到找到一个空闲单元为止（当m≥n时一定能找到一个空闲单元）
  - 当到达下标为m-1的哈希表表尾时，下一个探测的地址是表首地址0
  - 线性探测法的数学递推描述公式为：

    $d_0=h(k)$

    $d_i=(d_{i-1}+1) \bmod m \quad (1\le i\le m-1)$

- 平方探测法
  - 设发生冲突的地址为d，则平方探测法的探测序列为：$d\pm1^2$，$d\pm2^2$，...。
  - 平方探测法的数学描述公式为：
  - $d_0=h(k)$
  - $d_i=(d_0\pm i^2) \bmod m \quad (1\le i\le m-1)$
  - 评价
    - 平方探测法是一种较好的处理冲突的方法，可以避免出现堆积问题。
    - 它的缺点是不能探测到哈希表上的所有单元，但至少能探测到一半单元。

例 除留余数法定址，线性探测法解决冲突

- 问题
  - 假设哈希表长度m=13，采用除留余数法哈希函数建立如下关键字集合的哈希表：
    {16,74,60,43,54,90,46,31,29,88,77}。
- 解：

n=11，m=13

确定除留余数法的哈希函数为：h(k)=k mod p

p的选择：应为小于等于m的素数，假设p取值13，有：

h(16)=3，h(74)=9，h(60)=8，h(43)=4，
h(54)=2，h(90)=12，h(46)=7，h(31)=5，
h(29)=3，h(88)=10，h(77)=12

$d_i=(d_{i-1}+1) \bmod m$

哈希冲突解决方法(2): 拉链法

- 思想
  - 拉链法是把所有的同义词用单链表链接起来的方法。
- 例
  - 采用除留余数法哈希函数建立如下关键字集合的哈希表：{16,74,60,43,54,90,46,31,29,88,77}
  - 用拉链法解决冲突
- 优点
  - 简单，无堆积现象
  - 动态分配存储灵活
  - 节省空间（数据规模大时）
  - 删除节点方便
- 缺点
  - 指针需要额外空间（数据少时）