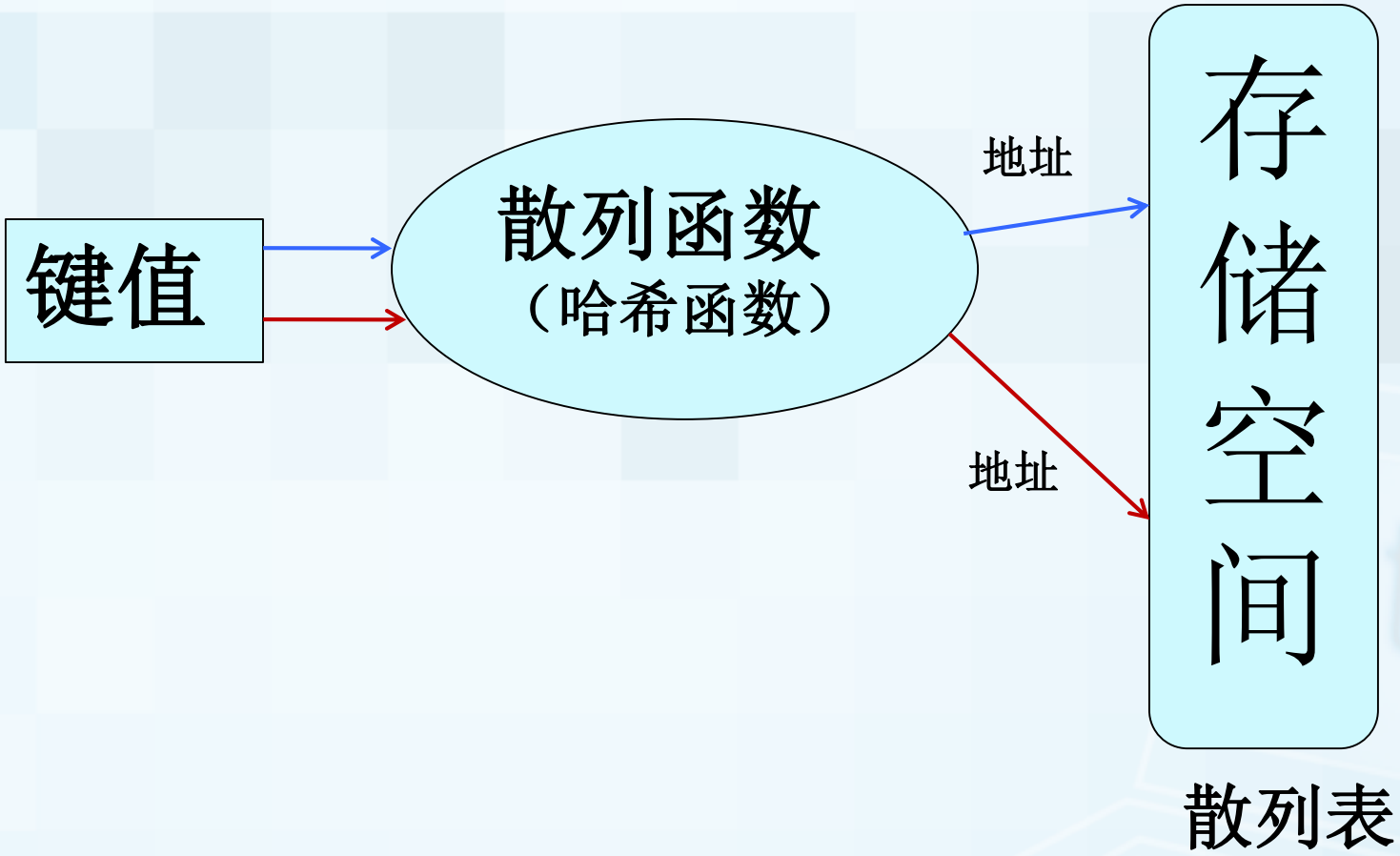
7.3 散列法

- 散列法（哈希表）的基本思想：



7.3 散列法

- **散列法（哈希表）的基本思想：**
  - 把**键**分布在一个称为**散列表**的一维数组H[0..m-1]中。
  - 可以利用**散列函数**来计算每个**键**的值，该函数为每个**键**指定一个称为**散列地址**的值。

- **散列函数举例：**
  - 如果**键**是一个**非负整数**，则h(K)=K *mod* m
  - 如果**键**是某个字母表中的**字母**，则可以把该字母在字母表中的**位置**指定为键，记为ord(K)

7.3 散列法

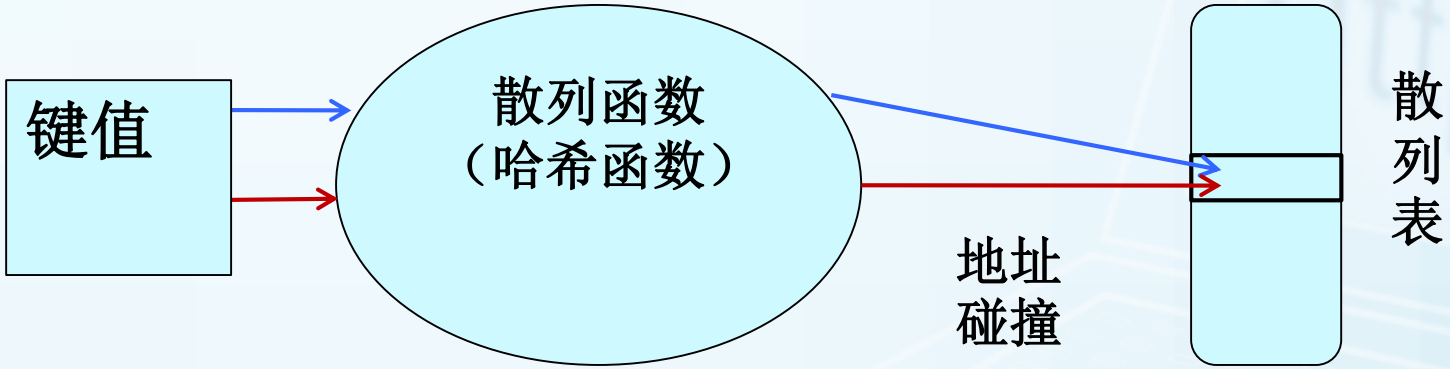
- **散列函数举例：**
  - 如果**键**是一个**字符串：K=** $c_0c_1...c_{s-1}$，则可定义
    - $h(K) = (\sum ord(c_i)) \bmod m$

或者更好的做法是 （**C 为大于每个ord(c_i)的常量**）：

```
h←0;
for i←0 to s-1 do
    h←(h*C+ord(ci)) mod m
```

7.3 散列法

- **散列函数**必须满足**两个要求**：
  - 需要把**键**在**散列表**的单元中**尽可能的均匀分布**
  - 必须是**容易计算**的
- **碰撞**
  - 当散列表的长度m小于键的数量n时，会有两个或多个键被映射到同一个单元中
  - 一般，即使m相对于n足够大，碰撞还是会发生



7.3 散列法

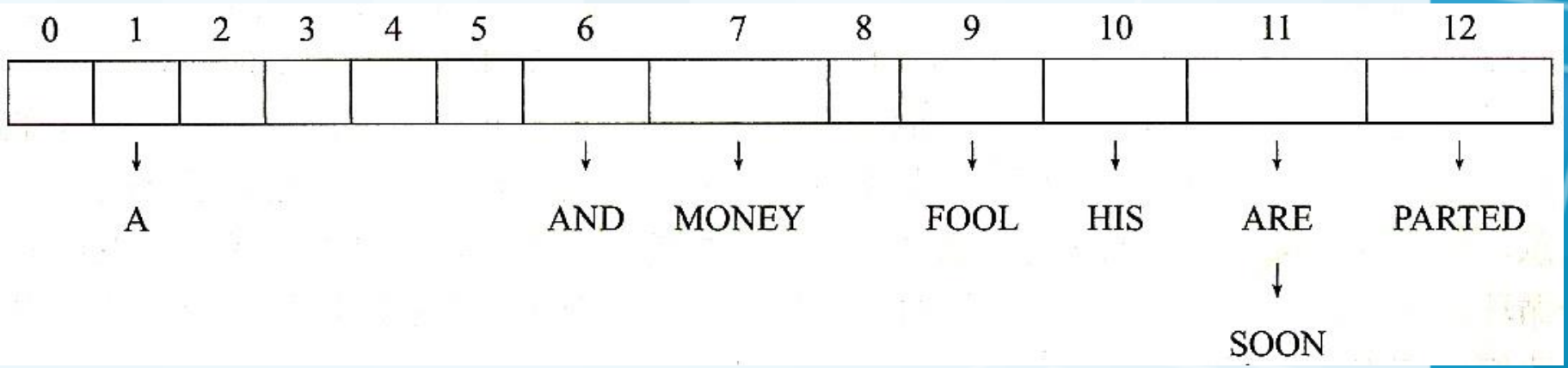
- **散列法**的两个版本
  - **开散列**（分离链）
  - **闭散列**（开式寻址）

7.3 散列法-开散列/分离链

- 在**开散列**中，**键**被存放于以散列表单元开头的**链表**中。

A,FOOL,AND,HIS,MONEY,ARE,SOON,PARTED

键	A	FOOL	AND	HIS	MONEY	ARE	SOON	PARTED
散列地址	1	9	6	10	7	11	11	12



7.3 散列法-开散列/分离链

- 一般来说，**查找的效率**取决于**链表的长度**，而这个长度有取决于**字典和散列表的长度**以及**散列函数的质量**。

- 如果该**散列函数**大致均匀地将**n个键**分布在散列表的**m个单元**中，每个链表的长度大约相当于n/m，其 α =n/m称为散列表的负载因子。
- **成功查找**中平均需检查的指针个数S=1+ α /2
- **不成功查找**中平均需检查的指针个数U= α

- 通常情况下我们希望**负载因子和1不要相差太大**。

7.3 散列法-闭散列/开式寻址

- 在闭散列中，所有键都存储在散列表本身。
- 碰撞发生时，
  - 如果下一个单元格空，则放在下一个单元格;
  - 如果不空，则继续找到下一个空的单元格，如果到了表尾，则返回到表首继续。

7.3 散列法-闭散列/开式寻址

键	A	FOOL	AND	HIS	MONEY	ARE	SOON	PARTED
散列地址	1	9	6	10	7	11	11	12

0	1	2	3	4	5	6	7	8	9	10	11	12
	A											
	A								**FOOL**			
	A					**AND**			FOOL			
	A					AND			FOOL	**HIS**		
	A					AND	**MONEY**		FOOL	HIS		
	A					AND	MONEY		FOOL	HIS	**ARE**	
	A					AND	MONEY		FOOL	HIS	ARE	**SOON**
PARTED	A					AND	MONEY		FOOL	HIS	ARE	SOON

7.3 散列法-闭散列/开式寻址

- **闭散列**的**查找**和**插入**操作是**简单**而直接的，但是**删除操作**则会带来不利的后果。

- **思考：**
  - 删除操作会带来什么问题？
  - 如何处理更好？

7.3 散列法-闭散列/开式寻址

- 比起**开散列**，**闭散列**的数学分析是复杂得多的问题。

- 对于**负载因子**为α的**散列表**，**成功查找**和**不成功查找**必须要访问的次数分别为：
  - $S \approx (1+1/(1-\alpha))/2$　　$U \approx (1+1/(1-\alpha)^2)/2$
  - 散列表的规模越大，该近似值越精确

- 课堂练习

对于输入 30， 20， 56， 75， 31， 19，
设散列表大小 m =11，
散列函数为 h(K)=K mod 11，

– 构造出它们的开散列表
– 构造出它们的闭散列表