双向链表解决冲突

简单均匀散列

> m槽，每个元素散列位置相对独立

- 查找时间Θ(1+α)

> 无论是否是成功查找

- 冲突数期望C（2，n）/m

> n个关键字，m个槽

Hash函数

除法取模

> 寻找取模对象

- 尝试1：r=lgn，m=2^r

> 优点：每次h（k）取后r位，计算快
> 缺点：如果输入均为偶数，则只利用了一半空间

- 尝试2：如果key k和m有公因子d

> 则可能只能利用到1/d的空间

- 目标：寻找不靠近2^r的一个质数

乘法取浮点

> 假设输入最多w位

- Fix table size $m = 2^r$ for some $r \leq w$.
- Fix constant integer $0 < A < 2^w$.
- Hash function: $h(k) = (Ak \bmod 2^w) \gg (w - r)$

随机hash函数

> 任意hash 函数fixed和known之后，总能找到恶意输入bad keys，
> 使得映射相同或到一个极小区间，因此要随机化
> 没有input总是bad

开放寻址open addressing

> 所有元素在散列表中
> 无链表，无指针，节省空间给更多槽

> <h（k，0），h（k，1）.，h（k，m-1）>是<0，1，…、m-1>的一个排列，
>
> 散列表逐渐填满时，每个表位都可以被考虑为插入新关键字的槽

Insert

HashInsert(T,k):

```
i=0
repeat
  j=h(k,i)
  if (T[j]==NIL)
    T[j]=k
    return j
  else i=i+1
until (i==m)
return "overflow"
```

T[j]==NIL or T[j]==DEL

search

HashSearch(T,k): No change!

```
i=0
repeat
  j=h(k,i)
  if (T[j]==k)
    return j
  i=i+1
until (i==m or T[j]==NIL)
return NIL
```

remove

HashRemove(T,k):

```
pos=HashSearch(T,k)
if (pos!=NIL)
  T[pos]=DEL
return pos
```

线性探查

```
h(k,i)=(h'(k)+i) mod m  (i=0~m-1)
```

> 容易实现，k+探查次数-->得到index
>
> 连续非空时效率低

二次探查

```
h(k,i)=(h'(k)+c*i+cc*i^2) mod m  (i=0~m-1)
```

> 跳着找

双重散列

```
h(k,i)=(h_1(k)+i*h_2(k)) mod m
```

> h_1和h_2为辅助散列函数
> 最优方法，接近理想均匀散列性能

- 装载因子α=n/m（<1）
- 不成功查找次数--找到空

> 次数对应时间复杂度
> 除了最后一次探查，每次都要检查一个被占用但失败的槽

$\mathbb{E}[X] \leq 1/(1-\alpha)$. Here, $\alpha = n/m < 1$.

- [ ] 证明:

> 第i次探查后，还剩m-i+1个待探位置，至多还有n-i+1个待探值

- $\Pr[A_i | A_1 \cap A_2 \cap \ldots \cap A_{i-1}] = (n-(i-1))/(m-(i-1)) \leq n/m$
- $\Pr[X \geq i] = \Pr[A_1 \cap A_2 \cap \ldots \cap A_{i-1}]$
  $= \Pr[A_1] \cdot \Pr[A_2|A_1] \cdot \Pr[A_3|A_1 \cap A_2] \cdots \Pr[A_{i-1}|A_1 \cap A_2 \cap \ldots \cap A_{i-2}]$
  $\leq (n/m)^{i-1} = \alpha^{i-1}$
- $\mathbb{E}[X] = \sum_{i=1}^{\infty} \Pr[X \geq i] \leq \sum_{i=1}^{\infty} \alpha^{i-1} = 1 + \alpha + \alpha^2 + \ldots = \frac{1}{1-\alpha}$

- 插入探查次数1/(1-α)
- 成功查找次数

$\mathbb{E}[X] \leq \frac{1}{\alpha}\ln\left(\frac{1}{1-\alpha}\right)$. Here, $\alpha = n/m < 1$.

完全散列