11.4 散列表的性能分析

散列表的性能分析

- 平均查找长度（ASL）用来度量散列表查找效率：成功、不成功
- 关键词的比较次数，取决于产生冲突的多少

影响产生冲突多少有以下三个因素：

（1）散列函数是否均匀；不均匀冲突会更多

（2）处理冲突的方法；

（3）散列表的装填因子α。

分析：不同冲突处理方法、装填因子对效率的影响

1. 线性探测法的查找性能

可以证明，线性探测法的期望探测次数 满足下列公式：

$$p=\begin{cases}\dfrac{1}{2}\left[1+\dfrac{1}{(1-\alpha)^2}\right] & \text{（对插入和不成功查找而言）}\\ \dfrac{1}{2}\left(1+\dfrac{1}{1-\alpha}\right) & \text{（对成功查找而言）}\end{cases}$$

当**α= 0.5**时，

- 插入操作和不成功查找的期望 **ASLu = 0.5*(1+1/(1-0.5)^2) = 2.5** 次
- 成功查找的期望 **ASLs = 0.5*(1+1/(1-0.5)) = 1.5**次

H(key)	0	1	2	3	4	5	6	7	8	9	10	11	12
key	11	30		47				7	29	9	84	54	20
冲突次数	0	6		0				0	1	0	3	1	3

α= 9/13=0.69，于是

期望 **ASLu = 0.5*(1+1/(1-0.69)^2) = 5.70**次

期望 **ASLs = 0.5*(1+1/(1-0.69)) = 2.11**次（实际计算**ASLs =2.56**）

2. 平方探测法和双散列探测法的查找性能

可以证明，平方探测法和双散列探测法探测次数 满足下列公式：

$$p=\begin{cases}\dfrac{1}{1-\alpha} & \text{（对插入和不成功查找而言）}\\ -\dfrac{1}{\alpha}\ln(1-\alpha) & \text{（对成功查找而言）}\end{cases}$$

当α= 0.5时，

- 插入操作和不成功查找的期望 ASLu = 1/(1-0.5) = 2 次
- 成功查找的期望 ASLs = -1/0.5* ln(1-0.5) ≈ 1.39 次

H(key)	0	1	2	3	4	5	6	7	8	9	10
key	11	30	20	47			84	7	29	9	54
冲突次数	0	3	3	0			2	0	1	0	0

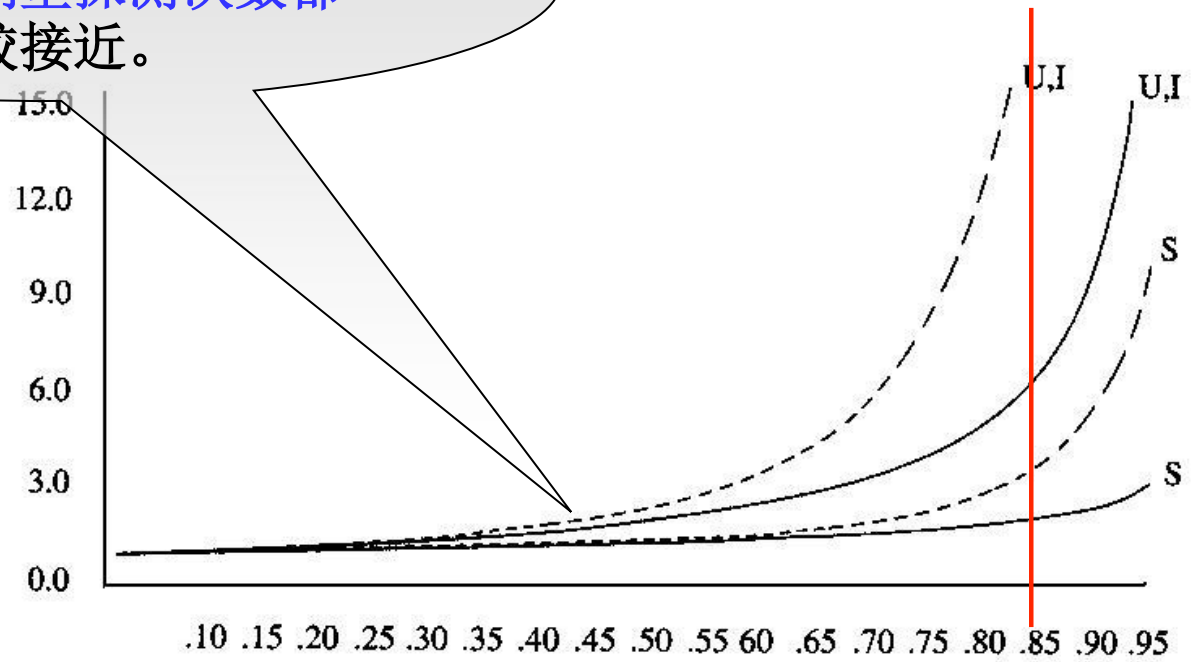
α= 9/11=0.82，于是

期望 ASLu = 1/(1-0.82) ≈ 5.56次

期望 ASLs = -1/0.5* ln(1-0.5) ≈ 2.09次（例中ASLs =2）。

❖ 期望探测次数与装填因子α的关系。

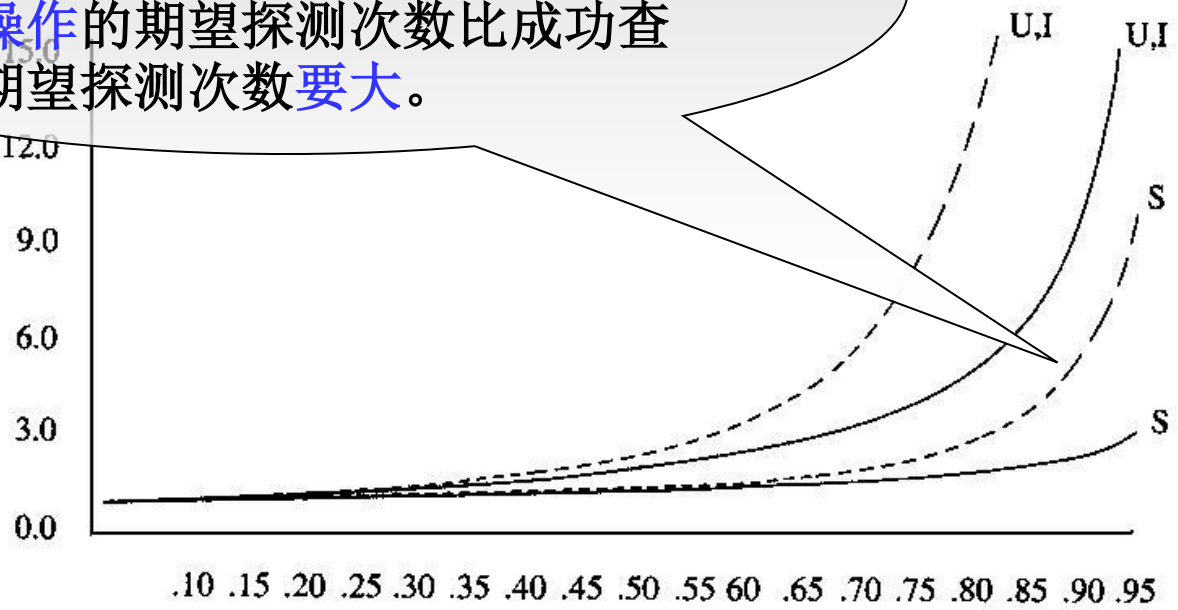
当装填因子α< 0.5的时候，各种探测法的期望探测次数都不大，也比较接近。



线性探测法（虚线）、双散列探测法（实线）
U表示不成功查找，I表示插入，S表示成功查找

❖ 期望探测次数与装填因子α的关系。

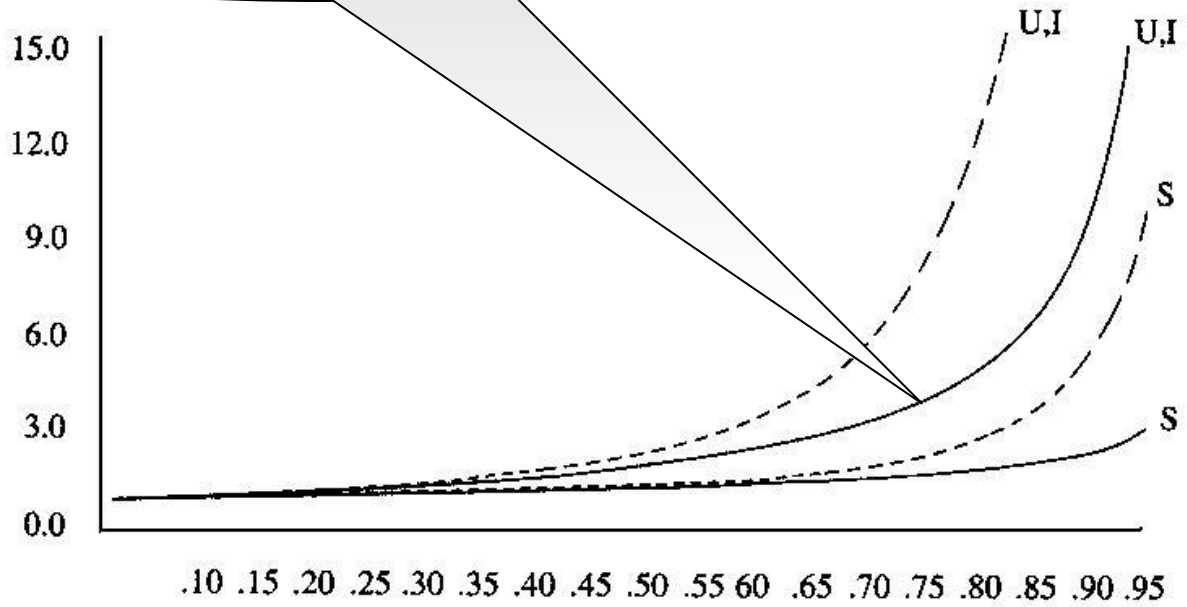
随着 α 的增大，线性探测法的期望探测次数增加较快，不成功查找和插入操作的期望探测次数比成功查找的期望探测次数要大。



线性探测法（虚线）、双散列探测法（实线）
U表示不成功查找，I表示插入，S表示成功查找

❖ 期望探测次数与装填因子α的关系。

合理的的最大装入因子α应该不超过0.85。



线性探测法（虚线）、双散列探测法（实线）

U表示不成功查找，I表示插入，S表示成功查找

建议装填因子不要超过0.85

3. 分离链接法的查找性能

所有地址链表的平均长度定义成装填因子α，α有可能超过1。
不难证明：其期望探测次数 p为：

$$p = \begin{cases} \alpha + e^{-\alpha} & \text{（对插入和不成功查找而言）} \\ 1 + \dfrac{\alpha}{2} & \text{（对成功查找而言）} \end{cases}$$

当α = 1时，

- 插入操作和不成功查找的期望 ASLu = $1+e^{-1} = 1.37$ 次，
- 成功查找的期望 ASLs = 1+1/2 = 1.5 次。

➢ 前面例子14个元素分布在11个单链表中，所以α= 14/11≈1.27， 故
期望 ASLu = $1.27+e^{-1.27} \approx$ 1.55次
期望 ASLs = 1+1.27/2 ≈ 1.64次（例中ASLs =1.36）。

选择合适的 h(key)，散列法的查找效率期望是常数*O(1)*，它几乎与关键字的空间的大小n无关！也适合于关键字直接比较计算量大的问题

它是以较小的α为前提。因此，散列方法是一个以空间换时间

散列方法的存储对关键字是随机的，不便于顺序查找关键字，也不适合于范围查找，或最大值最小值查找。

开放地址法：

散列表是一个数组，存储效率高，随机查找。

散列表有“聚集”现象

分离链法：

数组与链表结合


散列表是顺序存储和链式存储的结合，链表部分的存储效率和查找效率都比较低。当冲突较多时，链表需要从头到尾扫描，效率较低


关键字删除不需要“懒惰删除”法，从而没有存储“垃圾”。


太小的α可能导致空间浪费，大的α又将付出更多的时间代价。不均匀的链表长度导致时间效率的严重下降。