如果聚集不算是问题，那么对应的公式就不难得到。我们假设有一个很大的散列表，并设每次探测都与前面的探测无关。对于随机冲突解决方法而言，这些假设是成立的，并且当 λ 不是非常接近于 1 时也是合理的。首先，我们导出在一次不成功查找中探测的期望次数，而这正是直到我们找到一个空单元的探测的期望次数。由于空单元所占的份额为 $1-\lambda$，因此我们预计要探测的单元数是 $1/(1-\lambda)$。一次成功查找的探测次数等于该特定元素插入时所需要的探测次数。当一个元素被插入时，可以看成进行一次不成功查找的结果。因此，我们可以使用一次不成功查找的开销来计算一次成功查找的平均开销。

需要指出的是，λ 从 0 到当前值之间变化，因此早期的插入操作开销较少，从而将平均开销拉低。例如，在上面的图 5-11 中，$\lambda=0.5$，访问 18 的开销是在 18 被插入时确定的，此时 $\lambda=0.2$。由于 18 是插入到一个相对空的散列表中，因此对它的访问应该比新近插入的元素(比如 69)的访问更容易。我们可以通过使用积分计算插入时间平均值的方法来估计平均值，如此得到：

$$I(\lambda)=\frac{1}{\lambda}\int_0^{\lambda}\frac{1}{1-x}\mathrm{d}x=\frac{1}{\lambda}\ln\frac{1}{1-\lambda}$$

这些公式显然优于线性探测那些相应的公式。聚集不仅是理论上的问题，而且实际上也发生在具体的实现中。图 5-12 把线性探测的性能(虚曲线)与从更随机的冲突解决方法中期望的性能作了比较。成功的查找用 S 标记，不成功查找和插入分别用 U 和 I 标记。

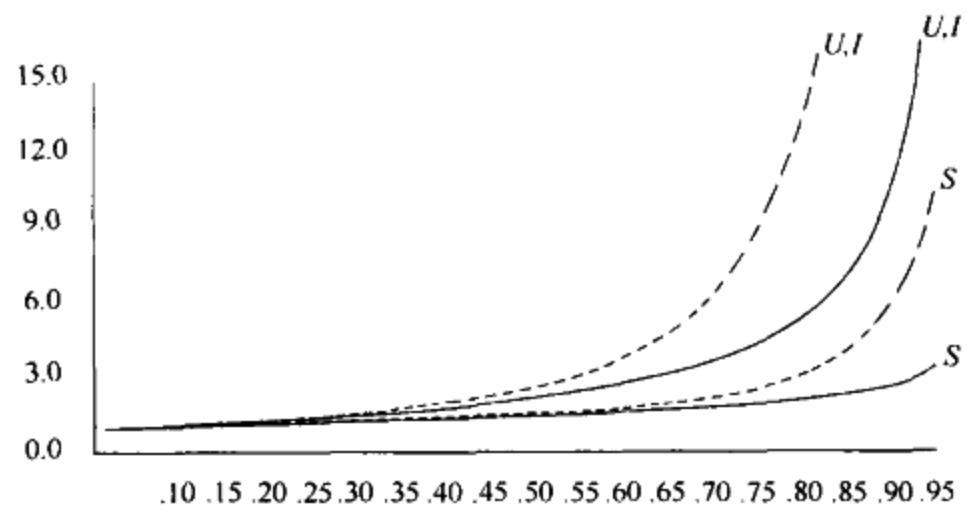

图 5-12　对线性探测(虚线)和随机方法的装填因子画出的探测次数
(S 为成功查找，U 为不成功查找，I 为插入)

如果 $\lambda=0.75$，那么上面的公式指出在线性探测中一次插入预计 8.5 次探测。如果 $\lambda=0.9$，则预计为 50 次探测，这就不切实际了。假如聚集不是问题，那么这可与相应的装填因子的 4 次和 10 次探测相比。从这些公式看到，如果表可以有多于一半被填满的话，那么线性探测就不是个好办法。然而，如果 $\lambda=0.5$，那么插入操作平均只需要 2.5 次探测，并且对于成功的查找平均只需要 1.5 次探测。

5.4.2　平方探测法

平方探测是消除线性探测中一次聚集问题的冲突解决方法。平方探测就是冲突函数为二次的探测方法。流行的选择是 $f(i)=i^2$。图 5-13 显示与前面线性探测例子相同的输入使用该冲突函数所得到的散列表。

当 49 与 89 冲突时，其下一个位置为下一个单元，该单元是空的，因此 49 就被放在那里。此后，58 在位置 8 处产生冲突，其后相邻的单元经探测得知发生了另外的冲突。下一个探测的单元在距位置 8 为 $2^2=4$ 远处，这个单元是个空单元。因此，关键字 58 就放在单元 2 处。对于关键字 69，处理的过程也一样。

对于线性探测，让散列表几乎填满元素并不是个好主意，因为此时表的性能会降低。对于平