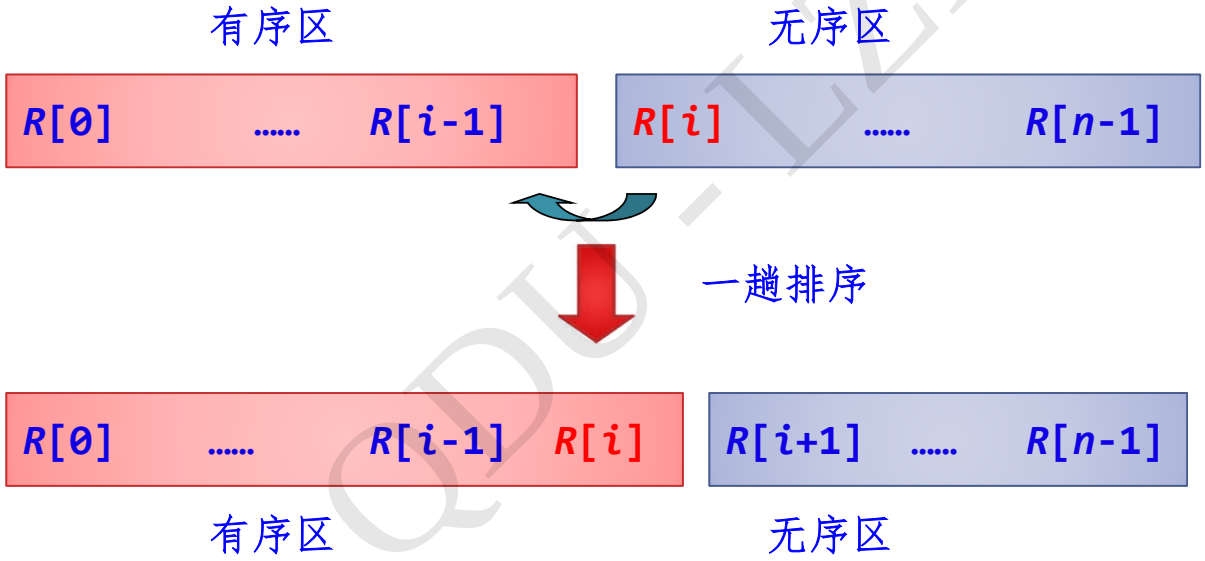
10.2 插入排序

- 插入排序的**基本思路**：每一趟将一个待排序的记录，按其关键字值的大小插入到已经排序的部分文件中适当位置上，直到全部插入完成。
- 主要的插入排序算法：直接插入排序、折半插入排序和希尔排序。

10.2.1 直接插入排序

直接插入排序是一种最简单的排序方法，其过程是依次将每个记录插入到一个有序的序列中去。



- 初始时，有序区只有一个元素R[0]
- i＝1~n-1，共经过n-1趟排序

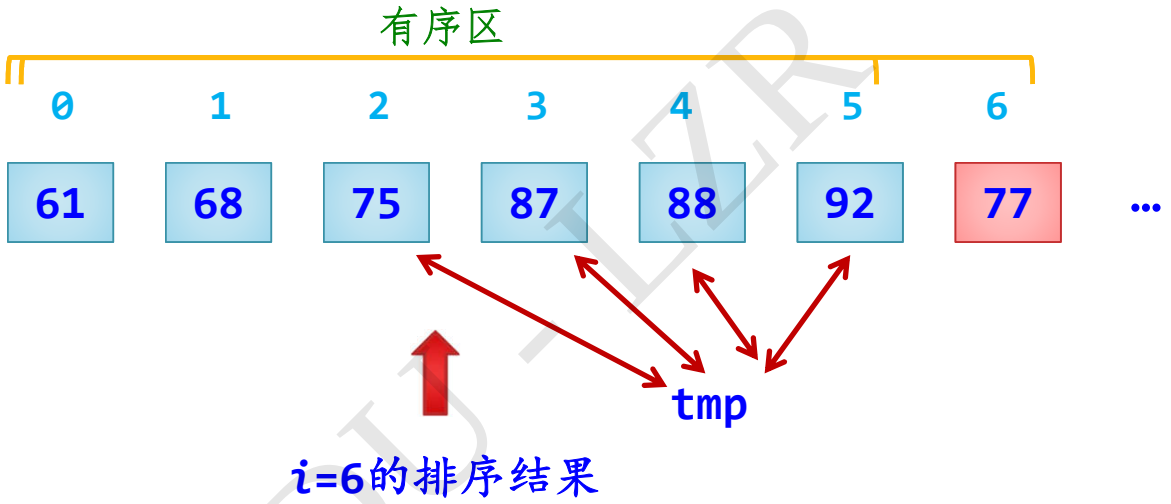
【示例-1】 已知有10个待排序的记录，它们的关键字序列为(75,87,68,92,88,61,77,96, 80,72)，给出用直接插入排序法进行排序的过程。

初始序列	75	87	68	92	88	61	77	96	80	72
i=1	75	87	68	92	88	61	77	96	80	72
i=2	68	75	87	92	88	61	77	96	80	72
i=3	68	75	87	92	88	61	77	96	80	72
i=4	68	75	87	88	92	61	77	96	80	72
i=5	61	68	75	87	88	92	77	96	80	72
i=6	61	68	75	77	87	88	92	96	80	72
i=7	61	68	75	77	87	88	92	96	80	72
i=8	61	68	75	77	80	87	88	92	96	72
i=9	61	68	72	75	77	80	87	88	92	96
最终结果	61	68	72	75	77	80	87	88	92	96

i=1的行表示i=1这一趟的排序结果

以i=6即插入77为例说明一趟的过程：

i=5的排序结果：



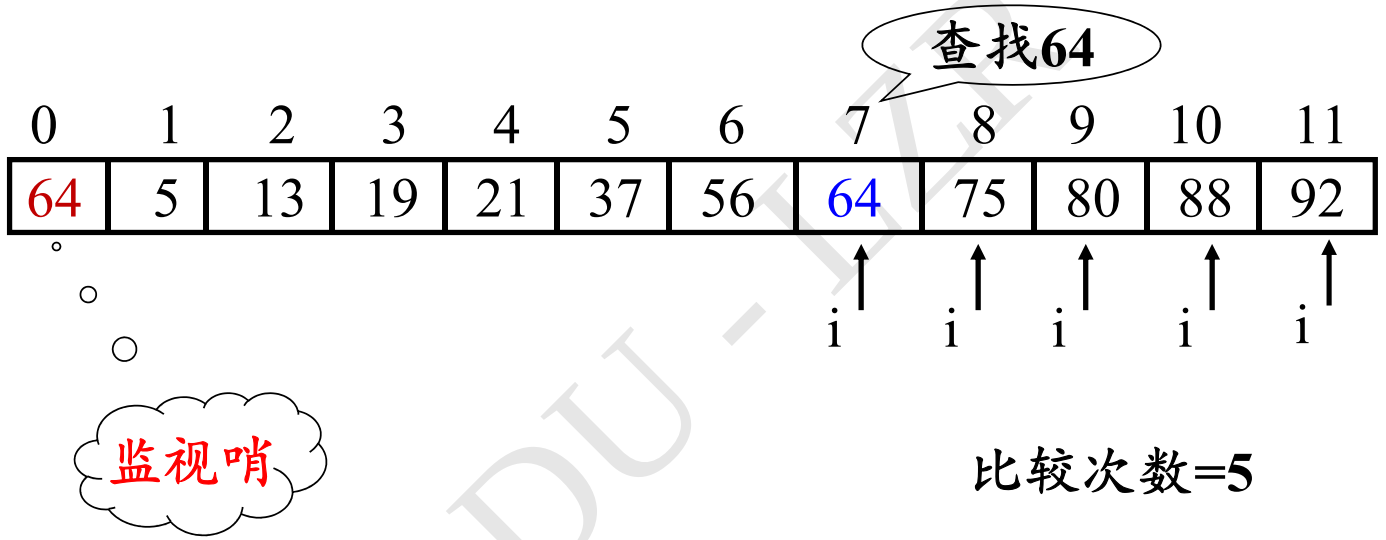
直接插入排序算法如下：

```
void InsertSort(RedType R[], int n)
{
   int i, j;
   RedType tmp;
   for (i=1;i<n;i++)              //从第二个元素即R[1]开始的
   {
      if (R[i-1].key>R[i].key)
      {  tmp=R[i];                //取出无序区的第一个元素R[i]
         j=i-1;                   //在R[0..i-1]中找R[i]的插入位置
         do
         {  R[j+1]=R[j];          //将关键字大于tmp.key的元素后移
            j--;                  //继续向前比较
         } while (j>=0 && R[j].key>tmp.key);
         R[j+1]=tmp;              //在j+1处插入R[i]
      }
   }
}
```

直接插入排序算法（设置哨兵）如下：

```
void InsertSort(SqList &L)
{
    // 对顺序表L作直接插入排序。算法10.1
    int i, j;
    for(i = 2; i <= L.length; ++i)
        // "<",需将L.r[i]插入有序子表
        if (LT(L.r[i].key, L.r[i - 1].key)) {
            L.r[0] = L.r[i];                    // 复制为哨兵
            for(j = i - 1; LT(L.r[0].key, L.r[j].key); --j)
                L.r[j + 1] = L.r[j];            // 记录后移
            L.r[j + 1] = L.r[0];                // 插入到正确位置
        }
}
```

Keys = (5　13　19　21　37　56　64　75　80　88　92)



算法9-1　顺序查找示例

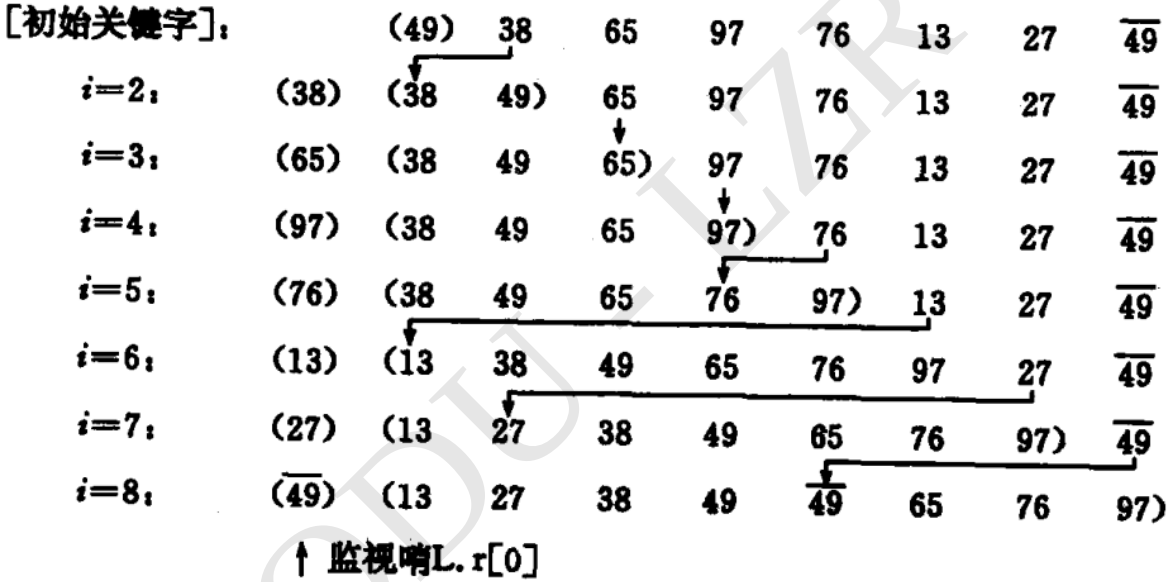


图 10.1　直接插入排序示例

直接插入排序算法分析：

最好的情况（关键字在记录序列中顺序有序）：

“比较”的次数：

$$\sum_{i=1}^{n-1} 1 = n-1$$

“移动”的次数：

0

最坏的情况（关键字在记录序列中逆序有序）：

“比较”的次数：

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

“移动”的次数：

$$\sum_{i=1}^{n-1} (i+2) = \frac{(n-1)(n+4)}{2}$$

总的平均比较和移动次数约为

$$\sum_{i=1}^{n-1} \left(\frac{i}{2} + \frac{i}{2} + 2\right) = \sum_{i=1}^{n-1} (i+2) = \frac{(n-1)(n+4)}{2} = O(n^2)$$

归纳起来，直接插入排序算法的性能如表所示。

时间复杂度			空间复杂度	稳定性
最好情况	最坏情况	平均情况		
$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	稳定

10.2.2 折半插入排序

- 有序区的有序性。
- 可以采用折半查找方法先在R[0..i-1]中找到插入位置，再通过移动记录进行插入。这样的插入排序称为折半插入排序或二分插入排序。

- 折半插入排序的元素移动次数与直接插入排序相同，不同的仅是变逐个移动为集中移动。
- 在**R[0..i-1]**中查找插入**R[i]**的位置，折半查找的平均关键字比较次数为$\log_2(i+1)$，平均移动元素的次数为**i/2+2**，所以平均时间复杂度为：

$$\sum_{i=1}^{n-1}(\log_2(i+1)+\frac{i}{2}+2)=O(n^2)$$

- 就平均性能而言，折半查找优于顺序查找，所以折半插入排序也优于直接插入排序。

END