本节主题：

插入排序

插入排序

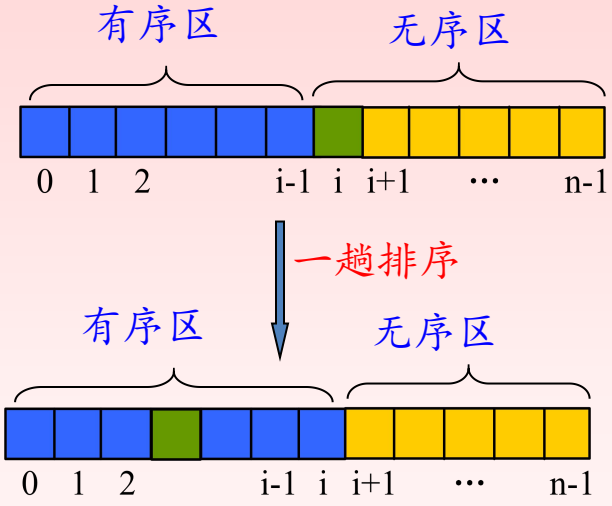
- 基本思想
  - 每次将一个待排序的记录，按其关键字大小插入到前面已经排好序的子表中的适当位置，直到全部记录插入完成为止。
- 介绍两种插入排序方法

（1）直接插入排序（含二分插入排序）

（2）希尔排序

直接插入排序

- 假设待排序的记录存放在数组R[0..n-1]中
- 排序过程的某一中间时刻，R被划分成两个子区间R[0..i-1]和R[i..n-1]，其中
  - 已排好序的有序区
  - 当前未排序的部分，称其为无序区
- 直接插入排序的基本操作是将当前无序区的第1个记录R[i]插入到有序区R[0..i-1]中适当的位置上，使R[0..i]变为新的有序区。
- 直接插入排序每次使有序区增加1个记录，故常称为增量法。



示例及算法实现

- 例：直接插入排序过程：

初始关键字　49,38,65,97,76,13,27,49

第1趟　(38,49),65,97,76,13,27,49

第2趟　(38,49,65),97,76,13,27,49

第3趟　(38,49,65,97),76,13,27,49

第4趟　(38,49,65,76,97),13,27,49

第5趟　(13,38,49,65,76,97),27,49

第6趟　(13,27,38,49,65,76,97),49

第7趟　(13,27,38,49,49,65,76,97)

```
//对R[0..n-1]按递增有序进行直接插入排序
void InsertSort(RecType R[],int n)
{
   int i,j;
   RecType tmp;
   for (i=1; i<n; i++)
   {
      tmp=R[i];
      j=i-1;
      while (j>=0 && tmp.key<R[j].key)
      {
         R[j+1]=R[j];
         j--;
      }
      R[j+1]=tmp;
   }
}
```

当R[i].key=R[j].key且i>j时，直接排序算法将R[i]插入在R[j]的后面，使R[i]和R[j]的相对位置保持不变，所以**直接插入排序是一种稳定的排序方法**。

直接插入排序的性能

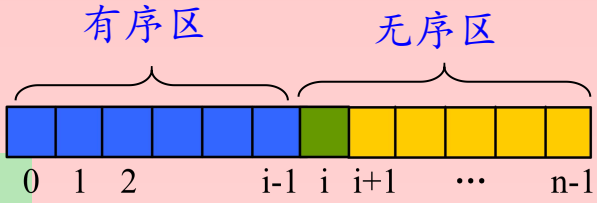
- 最好的情况
  - 关键字在记录序列中顺序有序
  - “比较”的次数：$\sum_{i=1}^{n-1} 1 = n-1$
  - “移动”的次数：$2(n-1)$
- 最坏的情况
  - 关键字在记录序列中逆序有序
  - “比较”的次数：$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$
  - “移动”的次数：$\sum_{i=1}^{n-1} (i+2) = \frac{(n-1)(n+4)}{2}$
- 总的平均比较和移动次数约为

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

```
//对R[0..n-1]按递增有序进行直接插入排序
void InsertSort(RecType R[],int n)
{
   int i,j,k;
   RecType tmp;
   for (i=1; i<n; i++)
   {
      tmp=R[i];
      j=i-1;
      while (j>=0 && tmp.key<R[j].key)
      {
         R[j+1]=R[j];
         j--;
      }
      R[j+1]=tmp;
   }
}
```

- 直接插入排序是一种稳定的排序方法。

折半插入排序




```
void InsertSort1(RecType R[],int n)
{
   int i,j,low,high,mid;
   RecType tmp;
   for (i=1; i<n; i++)
   {
      tmp=R[i];
      low=0;
      high=i-1;
      //用折半查找确定插入位置
      //顺序移动实施插入
   }
}
```

```
while (low<=high)
{
   mid=(low+high)/2;
   if (tmp.key<R[mid].key)
      high=mid-1;
   else
      low=mid+1;
}
```

```
for (j=i-1; j>=high+1; j--)
    R[j+1]=R[j];
R[high+1]=tmp;
```

- 折半插入排序的元素移动次数与直接插入排序相同，不同的仅是变分散移动为集合移动。
- 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$\log_2(i+1)-1$，平均移动元素的次数为$i/2+2$，所以平均时间复杂度为：

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

- 平均性能优于直接插入排序。
- 是非稳定排序方法