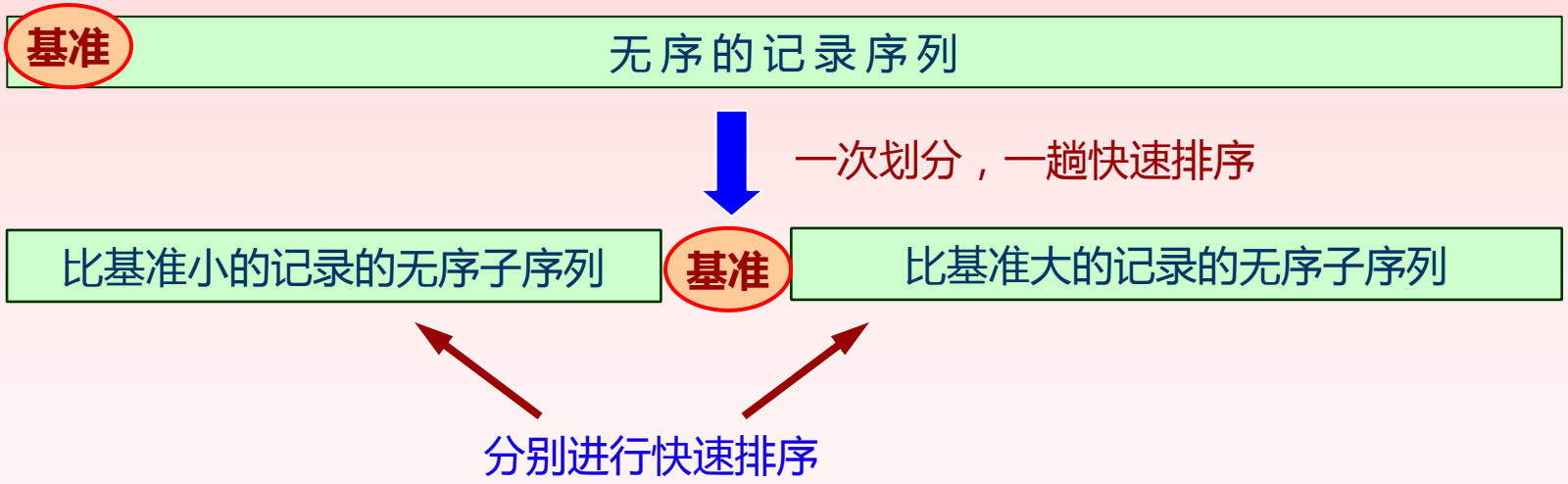
本节主题：

快速排序

快速排序的基本思想

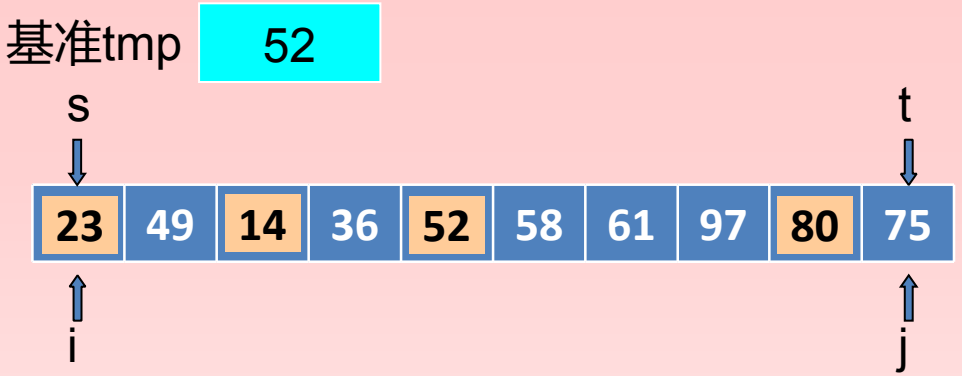
- 在待排序的n个记录中任取一个记录（通常取第一个记录）作为基准，把该记录放入适当位置后，数据序列被此记录划分成两部分，分别是比基准小和比基准大的记录。
- 再对基准两边的序列用同样的策略进行操作



快速排序的实现

```
//对R[s]至R[t]的元素进行快速排序
void QuickSort(RecType R[],int s,int t)
{
   int i=s,j=t;
   RecType tmp;
   if (s<t)
   {
      tmp=R[s];   //记录基准
      //进行一次划分
      R[i]=tmp;   //基准归位
      QuickSort(R,s,i-1);
      QuickSort(R,i+1,t);
   }
}
```

```
while (i!=j)
{
   while (j>i && R[j].key>=tmp.key)
      j--;
   R[i]=R[j];
   while (i<j && R[i].key<=tmp.key)
      i++;
   R[j]=R[i];
}
```



原始序列：

52, 49, 80, 36, 14, 58, 61, 97, 23, 75

一次划分后：

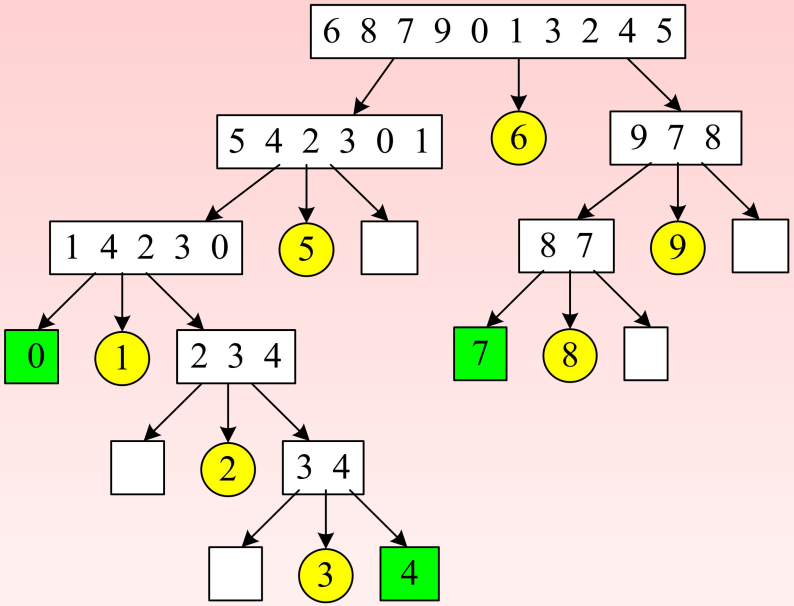
23, 49, 14, 36, (52), 58, 61, 97, 80, 75

对子序列分别继续划分

……

初始调用QuickSort(R, 0, n-1)

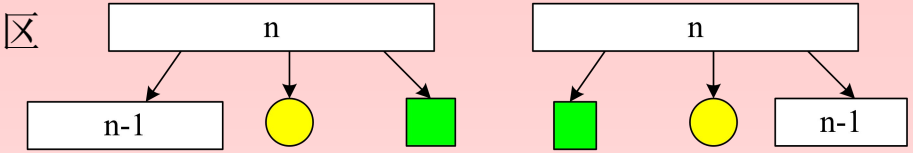
序列{6,8,7,9,0,1,3,2,4,5}的快速排序过程



快速排序的性能

- 最坏情况：每次划分的基准，是当前无序区中关键字最大（或最小）的元素
  - 需要n-1次划分
  - 第i次划分的长度为n-i+1，比较次数n-i

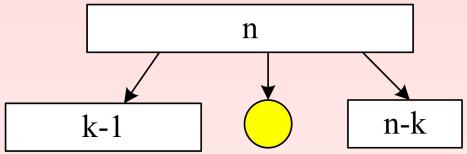
$$C_{max} = \sum_{i=1}^{n-1}(n-i) = \frac{n(n-1)}{2} = O(n^2)$$



- 平均情况

$$T_{avg}(n) = Cn + \frac{1}{n}\sum_{k=1}^{n}\left[T_{avg}(k-1) + T_{avg}(n-k)\right]$$

$$= O(n\log_2 n)$$



- 空间复杂度：$O(\log_2 n)$
  - 源自递归算法的栈空间消耗

- 非稳定的排序算法
  - 如：{5, 2, 4, 8, 7, 4}
  - 一次划分后：{4, 2, 4, 5, 7, 8}

基数的其他选择

```
void QuickSort1(RecType R[],int s,int t)
{
   int i=s,j=t;
   KeyType pivot;
   RecType tmp;
   pivot = R[(s+t)/2].key;
   if (s<t)
   {
      //进行一次划分
      QuickSort1(R,s,i-1);
      QuickSort1(R,j+1,t);
   }
}
```

```
while (i!=j)
{
   while (j>i && R[j].key>pivot)
      j--;
   while (i<j && R[i].key<pivot)
      i++;
   if(i<j)
   {
      tmp=R[i];
      R[i]=R[j];
      R[j]=tmp;
   }
}
```