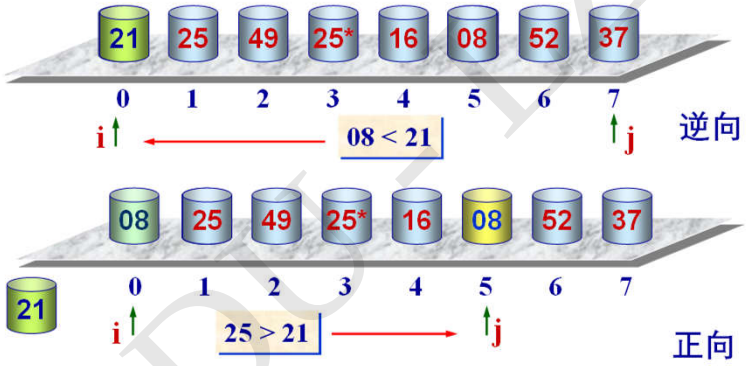
10.3.2 快速排序

- 快速排序的设计思想：
  - （1）找一个记录(例如取第一个记录)，以它的关键字作为“枢轴”（基准）；
  - （2）凡其关键字小于枢轴的记录均移动至该记录之前；
  - （3）凡其关键字大于枢轴的记录均移动至该记录之后。

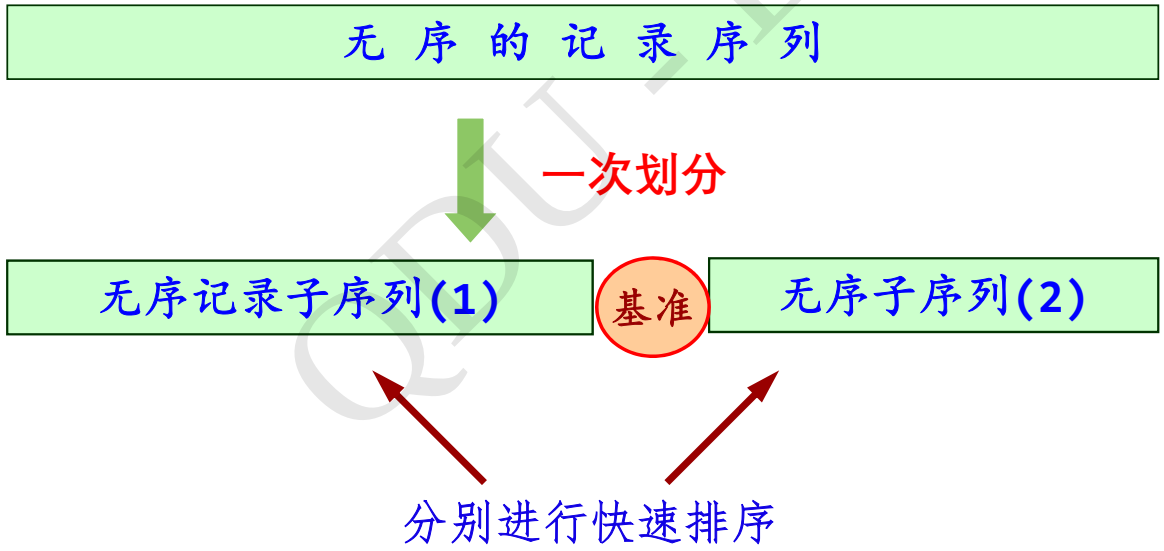
    即对无序的记录序列进行“一次划分”，
  - （4）之后分别对分割所得两个子序列“递归”进行快速排序。

快速排序演示



10.3.2 快速排序

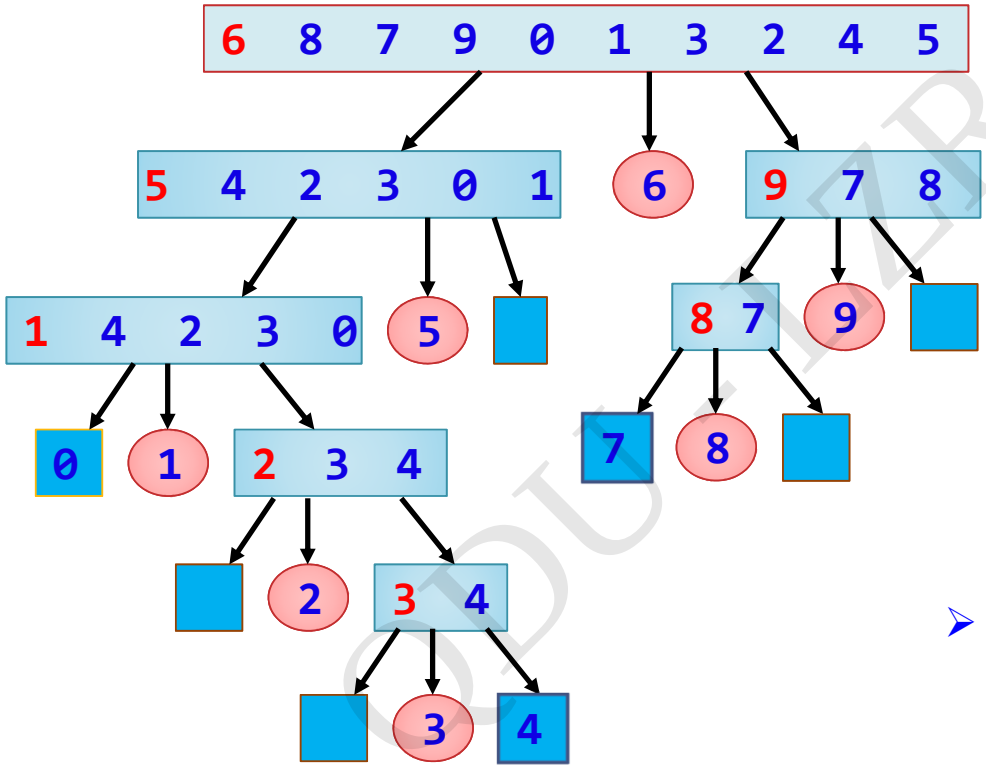
首先对无序的记录序列进行“一次划分”，之后分别对分割所得两个子序列“递归”进行快速排序。



- 每趟使表的第一个元素（基准）放入适当位置，将表一分为二，对子表按递归方式继续这种划分。
- 直至划分的子表长为0或者1。

【示例-2】设待排序的表有10个记录，其关键字分别为（6，8，7，9，0，1，3，2，4，5）。说明采用快速排序方法进行排序的过程。

快速排序递归树的演示



- 将递归树看成一棵3叉树，每个分支结点对应一次递归调用。这里递归次数：7
- 左右分区处理的顺序无关

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

【示例-3】采用递归方式对顺序表进行快速排序，下列关于递归次数的叙述中，正确的是(　　)。

A. 递归次数与初始数据的排列次序无关

B. 每次划分后，先处理较长的分区可以减少递归次数

C. 每次划分后，先处理较短的分区可以减少递归次数

D. 递归次数与每次划分后得到的分区处理顺序无关

说明：本题为2010年全国考研题。

解：快速排序的过程是以一个元素为基准将整个数据表一分为二，基准归位，左、右两个子表（分区）都是无序的，然后分别对左、右子表进行递归快速排序，无论先处理哪个子表都可以，也不影响递归树的结果和递归调用的次数。

本题答案为D。

快速排序算法如下：

```
int Partition(SqList &L, int low, int high)
{
    // 交换顺序表L中子表r[low..high]的记录，枢轴记录到位，并返回其
    // 所在位置，此时在它之前(后)的记录均不大(小)于它。算法10.6(b)
    KeyType pivotkey;
    L.r[0] = L.r[low];              // 用子表的第一个记录作枢轴记录
    pivotkey = L.r[low].key;        // 枢轴记录关键字
    while(low < high) {
        // 从表的两端交替地向中间扫描
        while(low < high && L.r[high].key >= pivotkey)
            --high;
        L.r[low] = L.r[high];       // 将比枢轴记录小的记录移到低端
        while(low < high && L.r[low].key <= pivotkey)
            ++low;
        L.r[high] = L.r[low];       // 将比枢轴记录大的记录移到高端
    }
    L.r[low] = L.r[0];              // 枢轴记录到位
    return low;                     // 返回枢轴位置
}
```

快速排序算法如下：

```
void QSort(SqList &L, int low, int high)
{
    // 对顺序表L中的子序列L.r[low..high]作快速排序。算法10.7
    int pivotloc;
    if(low < high) {  // 长度大于1
        // 将L.r[low..high]一分为二
        pivotloc = Partition(L, low, high);
       // 对低子表递归排序，pivotloc是枢轴位置
       QSort(L, low, pivotloc - 1);
       QSort(L, pivotloc + 1, high);        // 对高子表递归排序
    }
}

void QuickSort(SqList &L)
{
    // 对顺序表L作快速排序。算法10.8
    QSort(L, 1, L.length);
}
```

快速排序时间性能分析

假设一次划分所得枢轴位置 i=k，则对n 个记录进行快速排序所需时间：

$$T(n) = T_{pass}(n) + T(k-1) + T(n-k)$$

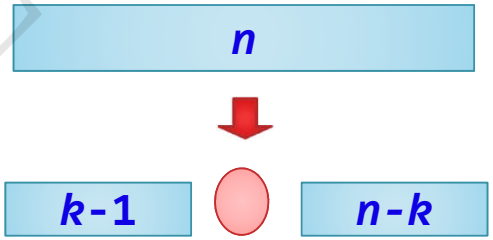
✓ 其中 $T_{pass}(n)$为对 n 个记录进行一次划分所需时间。

■ 若待排序列中记录的关键字是随机分布的，则 k 取 1 至 n 中任意一值的可能性相同。

由此可得快速排序所需时间的平均值为：

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

1次划分的时间



则可得结果：

$T_{avg}(n)=cn\log_2 n$。

结论： 快速排序的时间复杂度为$O(n\log_2 n)$

平均所需栈空间为$O(\log_2 n)$。

归纳起来，快速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		
$O(n\log_2 n)$	$O(n^2)$	$O(n\log_2 n)$	$O(\log_2 n)$	不稳定

END