

第十讲 排序（下）

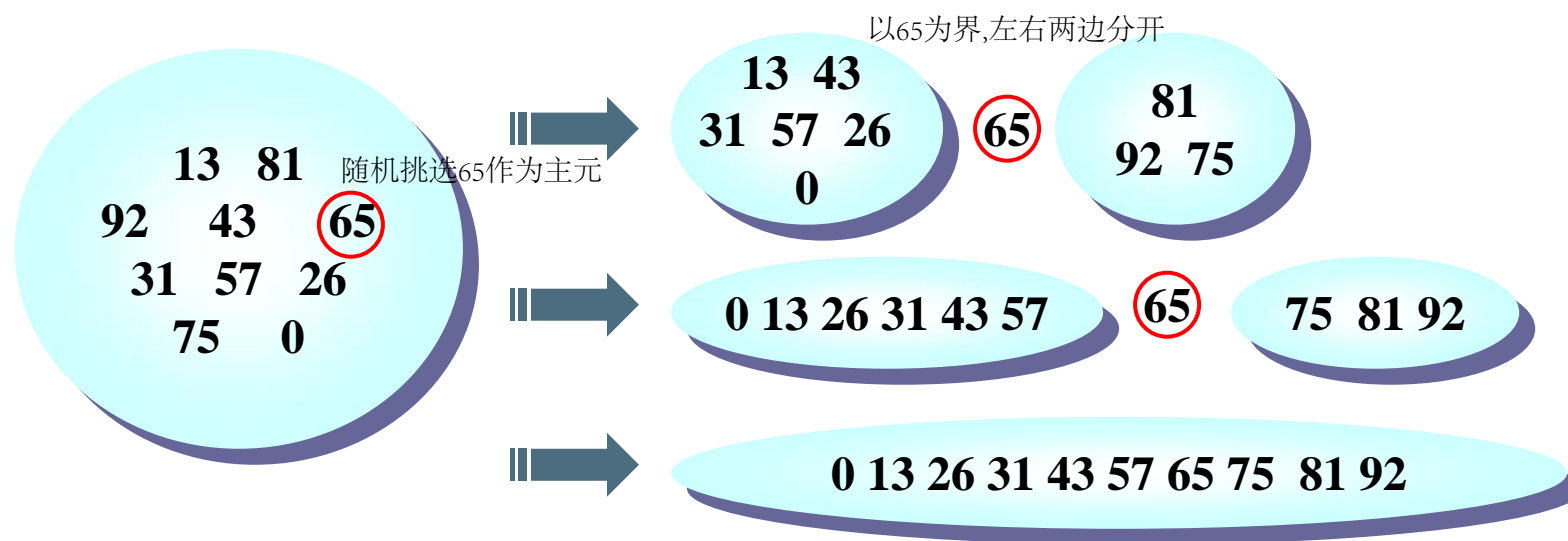
浙江大学 陈 越

10.1 快速排序

算法概述

快速排序不是稳定的算法。

■ 分而治之 --->想到递归



算法概述

什么是快速排序算法的最好情况？

■ 分而治之

挑选主元每次正好中分 $\longrightarrow T(N) = O(N \log N)$

```
void Quicksort( ElementType A[], int N )
{
    if ( N < 2 ) return;
    ? pivot = 从A[]中选一个主元;
    ? 将S = { A[] \ pivot } 分成2个独立子集:
        A1={ a∈S | a ≤ pivot } 和
        A2={ a∈S | a ≥ pivot };
    A[] = Quicksort(A1,N1) ∪
        {pivot} ∪
        Quicksort(A2,N2);
}
```

选主元

- 令 $\text{pivot} = A[0]$?

① 2 3 4 5 6 N-1 N
② 3 4 5 6 N-1 N
3 4 5 6 N-1 N



$$\begin{aligned} T(N) &= O(N) + T(N-1) \\ &= O(N) + O(N-1) + T(N-2) \\ &= O(N) + O(N-1) + \dots + O(1) \\ &= O(N^2) \end{aligned}$$

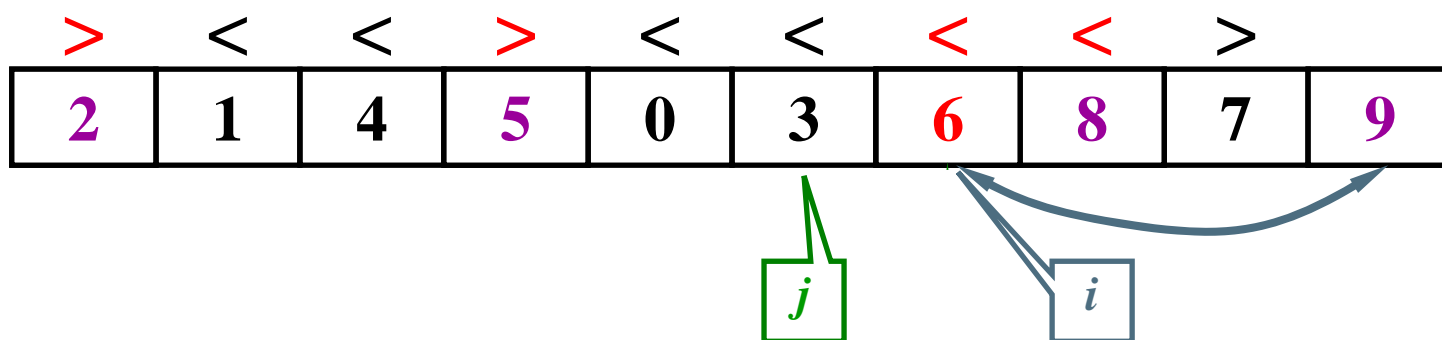
选主元

- 随机取 `pivot`? `rand()` 函数不便宜啊!
- 取头、中、尾的中位数
 - 例如 8、12、3的中位数就是8
 - 测试一下`pivot`不同的取法对运行速度有多大影响?

```
ElementType Median3( ElementType A[], int Left, int Right ) /*Left:头,Right:尾*/
{
    int Center = ( Left + Right ) / 2;
    if ( A[ Left ] > A[ Center ] )
        Swap( &A[ Left ], &A[ Center ] );
    if ( A[ Left ] > A[ Right ] )
        Swap( &A[ Left ], &A[ Right ] );
    if ( A[ Center ] > A[ Right ] )
        Swap( &A[ Center ], &A[ Right ] );
    /* A[ Left ] <= A[ Center ] <= A[ Right ] */
    Swap( &A[ Center ], &A[ Right-1 ] ); /* 将pivot藏到右边 */
    /* 只需要考虑 A[ Left+1 ] ... A[ Right-2 ] */
    return A[ Right-1 ]; /* 返回 pivot */
}
```

子集划分

快速排序之所以快的一个很重要的原因:它的主元被选中以后,在子集划分完成之后被一次性的放到了最终正确的位置上,以后再也不会移动



- 如果有元素正好等于 **pivot** 怎么办?
 - 停下来交换? ✓ $O(n \log n)$
 - 不理它, 继续移动指针? $O(n^2)$

小规模数据的处理

■ 快速排序的问题

- 用递归.....
- 对小规模的数据（例如 N 不到100）可能还不如插入排序快

■ 解决方案

- 当递归的数据规模充分小，则停止递归，直接调用简单排序（例如插入排序）
- 在程序中定义一个`Cutoff`的阈值 —— 课后去实践一下，比较不同的`Cutoff`对效率的影响

算法实现

```
void Quicksort( ElementType A[], int Left, int Right )
{
    if ( Cutoff <= Right-Left ) {
        Pivot = Median3( A, Left, Right );
        i = Left;  j = Right - 1;
        for( ; ; ) {
            while ( A[ ++i ] < Pivot ) { }
            while ( A[ --j ] > Pivot ) { }
            if ( i < j )
                Swap( &A[i], &A[j] );
            else break;
        }
        Swap( &A[i], &A[ Right-1 ] );
        Quicksort( A, Left, i-1 );
        Quicksort( A, i+1, Right );
    }
    else
        Insertion_Sort( A+Left, Right-Left+1 );
}
```

```
void Quick_Sort(ElementType A[],int N)
{
    Quicksort( A, 0, N-1 );
}
```