

## 25 第十讲 排序 (下)

笔记本: 浙江大学《数据结构》

创建时间: 2025/5/25 11:41

更新时间: 2025/5/25 19:23

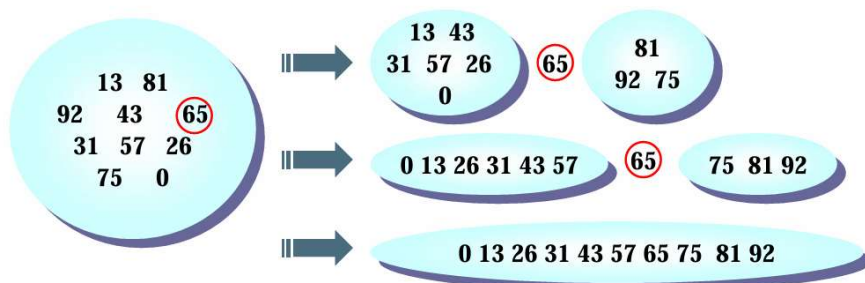
作者: panhengye@163.com

URL: <https://www.doubao.com/chat/6911774733615874>

## 快速排序

### 算法概述

#### ■ 分而治之



### 伪代码描述

```
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); }
```

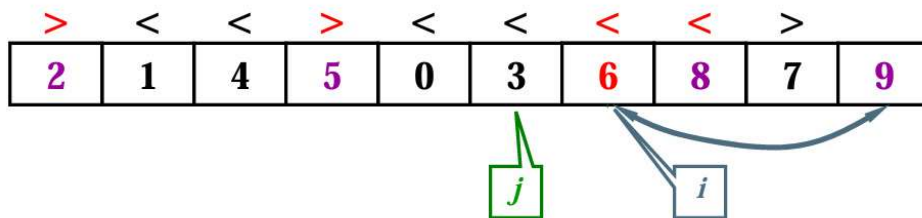
### 选主元

主要思路:

- 取头、中、尾的中位数
- 为了提高效率, 把center放在n-1的位置, 可以少处理元素

```
ElementType Median3( ElementType A[], int Left, int Right ) {    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，就是因为每一次选定主元，会被一次性放在正确位置上

- 当某个数字等于主元怎么办？继续执行交换，这样能确保主元位置较为居中
- 小规模数据的处理：
  - 当数据规模足够小的时候，直接调用简单排序（例如插入排序）
  - 定义一个阈值（Cutoff）

## 伪代码描述

```
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 );
}
```


## 表排序

每一个元素都是一个结构体，因此移动会带来很大的开销

## 算法概述

间接排序

- 定义一个指针数组作为“表”（table）
- 对指针进行排序，不挪动元素位置



A	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
key	f	d	c	a	g	b	h	e
table	3	5	2	1	7	0	4	6

如果仅要求按顺序输出，则输出：

$A[\text{table}[0]]$ ,  $A[\text{table}[1]]$ , ...,  $A[\text{table}[N-1]]$

## 物理排序

N个数字的排列由若干个独立的环组成



## 多关键字的排序



一副扑克牌是按2种关键字排序的

$K^0$  [花色]

♣ < ♦ < ♥ < ♠

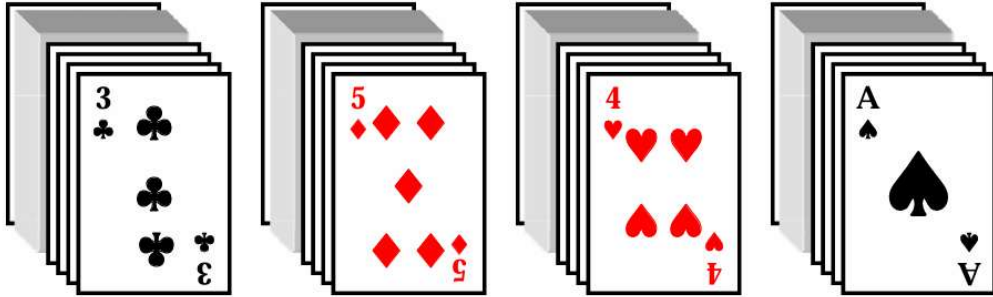
$K^1$  [面值]

2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < 10 < J < Q < K < A

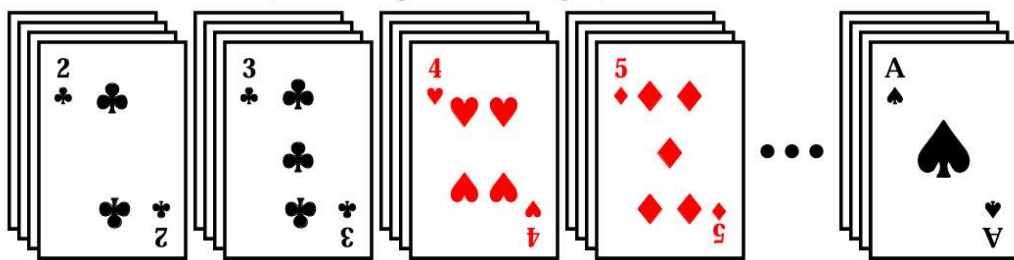
有序结果:

2♣ ... A♣ 2♦ ... A♦ 2♥ ... A♥ 2♠ ... A♠

主位优先 (4个桶)



次位优先 (13个桶)



如果要变成有序结果:

- 先次位优先
- 再主位优先

这样两次分类即可, 根本不用排序

## 排序算法的比较

排序方法	平均时间复杂度	最坏情况下时间复杂度	额外空间复杂度	稳定性
简单选择排序	$O(N^2)$	$O(N^2)$	$O(1)$	不稳定
冒泡排序	$O(N^2)$	$O(N^2)$	$O(1)$	稳定
直接插入排序	$O(N^2)$	$O(N^2)$	$O(1)$	稳定
希尔排序	$O(N^d)$	$O(N^2)$	$O(1)$	不稳定
堆排序	$O(N\log N)$	$O(N\log N)$	$O(1)$	不稳定
快速排序	$O(N\log N)$	$O(N^2)$	$O(\log N)$	不稳定
归并排序	$O(N\log N)$	$O(N\log N)$	$O(N)$	稳定
基数排序	$O(P(N+B))$	$O(P(N+B))$	$O(N+B)$	稳定