第7课 分治算法

分治算法概念

1.将一个复杂的问题分成两个或更多的相同或相似的子问题，再把子问题分成更小的子问题----“分”
2.最后子问题可以简单地直接求解----“治”
3.将所有子问题的解合并起来就是原问题的解----“合”

分治算法特征

1.该问题的规模缩小到一定的程度就可以容易地解决。
2.该问题可以分解为若干个规模较小的相同问题，即该问题具有最优子结构性质。
3.利用该问题分解出的子问题的解可以合并为该问题的解；
4.该问题所分解出的各个子问题是相互独立的，即子问题之间不包含公共的子子问题。

分治算法特征

第一条特征是绝大多数问题都可以满足的，因为问题的计算复杂性一般是随着问题规模的增加而增加；

第二条特征是应用分治法的前提，大多数问题也可以满足，此特征反映了递归思想的应用；

第三条特征是关键，能否利用分治法完全取决于问题是否具有第三条特征，如果具备了第一条和第二条特征，而不具备第三条特征，则可以考虑用贪心法或动态规划法。

第四条特征涉及到分治法的效率，如果各子问题是不独立的则分治法要做许多不必要的工作，重复地解公共的子问题，此时虽然可用分治法，但一般用动态规划法较好。

分治算法例子： 例1 对数组进行快速排序

```
#划分分区（非就地划分）
def partition(nums=list):
    pivot = nums[0]                              #挑选枢纽
    lo = [x for x in nums[1:] if x < pivot]     #所有小于pivot的元素
    hi = [x for x in nums[1:] if x >= pivot]    #所有大于pivot的元素
    return lo,pivot,hi
#快速排序
def quick_sort(nums=list):
    #被分解的Nums小于1则解决了
    if len(nums) <= 1:
        return nums
    #分解
    lo,pivot,hi = partition(nums)

    # 递归（树），分治，合并
    return quick_sort(lo) + [pivot] + quick_sort(hi)

lis = [7, 5, 0, 6, 3, 4, 1, 9, 8, 2]
print(quick_sort(lis)) #[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

例2：给定一个顺序表，编写一个求出其最大值的分治算法

```
#基本子算法（内置算法）
#虽然也可以处理大数组，这里用于解决分治问题规模小于或等于2的时候
def get_max(nums=list):
    return max(nums)
#分治法
def solve(nums):
    n = len(nums)
    if n <= 2:    #分治问题规模小于或等于2时解决
        return get_max(nums)
    # 分解（子问题规模为 n/2）
    left_list, right_list = nums[:n//2], nums[n//2:]
    # 递归（树），分治
    left_max, right_max = solve(left_list), solve(right_list)
    # 合并
    return get_max([left_max, right_max])

alist = [12,2,23,45,67,3,2,4,45,63,24,23]
# 求最大值
print(solve(alist))  # 67
```