`

图解算法三

快速排序——使用分而治之的策略

结论：

- 分而治之：一种著名的递归式问题解决思想
- 欧几里得算法：又名辗转相除法，一种常见且实用的算法思想
- **欧几里得算法**

> 又名辗转相除法，是一种求最大公约数的常用方法
>
> > 方式：以较大数除以较小数，再用出现的余数（第一余数）去除除数，然后再用出现的余数（第二余数）去除第一余数，如此往复，直至余为0，最后的除数就是最大公约数
> >
> > > A除B相当于说是用B除以A，即：B/A

注：比如说求（25,10）-> （25/10=2余5）-> （10/5=0余0）则最大公约数是整除全最后一个除数为5。

欧几里得算法用代码实现如下

```
    function Euclid(x, y) {
        let r = x > y ? x % y : y % x;
        let min = x > y ? y : x;
        if (r == 0) {
            return min;
        } else {
            return Euclid(min, r)
        }
    }
/**简化版本 */
    function Euclid(a, b) {
        return b === 0 ? a : Euclid(b, a % b);
    }
    Euclid(25, 10)//--5
```

使用递归函数进行数组之和/数组中最大值等获取

```
    function getSum(arr) {
        /*
        if(arr.length>1){
            //每次递归都移出数组第一项，*1避免字符串拼接而非运算
            return arr.shift()*1+getSum(arr)*1
        }else{
            return arr[0]
        }
```

```
        */
        //上方代码可简写成如下形式
        return arr.length == 1 ? arr[0] : arr.shift() + getSum(arr)
    }
    var arr = [2, 4, 5, 7, 5, 6, 2, 3]
    getSum(arr)

    function getMax(max, arr) {
        if (arr.length > 1) {
            //当前最大值大于数组当前首项则保持，否则重新赋值
            max = max > arr[0] ? max : arr[0];
            arr.shift()
            return getMax(max, arr)
        } else
            return max
    }
    var arr2 = [2, 4, 5, 7, 5, 6, 2, 3]
    getMax(0, arr2)
```

快速排序

1、算法的速度指的并非时间，而是操作数的增速。

2、谈论算法的速度时，说的是随着输入的增加，其运行时间将以什么样的速度增加。

3、算法的运行时间用大O表示法表示。

4、O(log n) 比 O(n)快，当需要搜索的元素越多时，前者比后者快得越多。

> 注：C语言标准库中得函数qsort实现的就是快速排序，快速排序也使用了D&C。

再谈大O表示法

- 其中，快速排序的独特之处在于，它的速度取决于选择的基准值
- 通常，用大O表示法是忽略会常量，常量对于一定大的数量级来说无关紧要
- 来查看下常用算法的大O表示法

算法	大O表示
二分查找	O(logn)
简单查找	O(n)
快速排序	O(n logn)
选择排序	O(n^2)
旅行商问题算法	O(n!)

> 还有一种合并排序（merge sort）的排序算法,运行时间为O（nlogn）

在快速排序中

- 快速排序由于性能依赖于选择的基准值

- 因此，快排有平均情况和最糟情况之说
- 情况一：
    - 若总是选择第一个元素作为基准值且处理的数组有序
    - 则此时的调用栈的高度为n,最糟情况调用栈的栈长为 O(n)
- 情况二：
    - 若总是选择中间的元素为基准值
    - 则此时调用栈的高度为n/2，最佳情况调用栈的栈长为 O(logn)

> 因此最佳情况下，调用栈高度为O(logn)，每层需要时间为O(n)该算法的运行时间为 **O(logn)*O(n)=O(nlogn)**

> 最糟糕情况下，调用栈高度为O(n)，每层需要时间为O(n)该算法的运行时间为 **$O(n)*O(n)=O(n^2)$**

注：最佳情况也是平均情况，快排的平均运行时间是 O(nlogn)

总结

> **由以上可知：**

- 实现随机排序时，请随机选择基准值元素，快排的平均运行时间为 O(nlogn)
- 比较简单查找和二分查找时，常亮可以忽略不计，当列表很长时，O(logn) 的速度比 O(n) 快的多。