# 分治策略
# 的设计思想

# 分治策略的基本思想

分治策略（ **Divide and Conquer** ）

1. 将原始问题划分或者归结为规模较小的子问题
1. 递归或迭代求解每个子问题
2. 将子问题的解综合得到原问题的解

注意：

1. 子问题与原始问题性质完全一样
2. 子问题之间可彼此独立地求解
3. 递归停止时子问题可直接求解

# 二分检索

**算法 Binary Search ($T, l, r, x$)**

**输入：数组 $T$，下标从 $l$ 到 $r$；数 $x$**

**输出：$j$ // 若$x$在$T$ 中, $j$ 为下标; 否则为 0**

1. $l \leftarrow 1$; $r \leftarrow n$
2. **while** $l \leq r$ **do**
3. $\quad m \leftarrow \lfloor (l+r)/2 \rfloor$ // $m$为中间位置
4. $\quad$**if** $T[m]=x$ **then return** $m$ // $x$是中位数
5. $\quad$**else** $\boxed{\text{if } T[m]>x \text{ then } r \leftarrow m-1}$
6. $\quad\quad\quad\boxed{\text{else } l \leftarrow m+1}$
7. **return** 0

# 二分检索算法设计思想

- 通过 $x$ 与中位数的比较，将原问题归结为规模减半的子问题，如果 $x$ 小于中位数，则子问题由小于 $x$ 的数构成，否则子问题由大于 $x$ 的数构成.
- 对子问题进行二分检索.
- 当子问题规模为 1 时，直接比较 $x$与 $T[m]$，若相等则返回 $m$，否则返回 0.

是否能够递归实现?

# 二分检索时间复杂度分析

二分检索问题最坏情况下时间复杂度

$$W(n) = W(\lfloor n/2 \rfloor)+1$$

$$W(1) = 1$$

可以解出

$$W(n)=\lfloor \log n \rfloor + 1$$

# 二分归并排序

算法 **Merge Sort** $(A, p, r)$

输入：数组 $A[p..r]$

输出：元素按从小到大排序的数组 $A$

1. **if** $p < r$
2. **then** $q \leftarrow \lfloor (p+r)/2 \rfloor$ 对半划分
3. **Merge Sort** $(A, p, q)$ 子问题1
4. **Merge Sort** $(A, q+1, r)$ 子问题 2
5. **Merge** $(A, p, q, r)$ 综合解

# 二分归并排序设计思想

- 划分将原问题归结为规模为 $n/2$ 的 2 个子问题
- 继续划分，将原问题归结为规模为 $n/4$ 的 4 个子问题. 继续…，当子问题规模为1 时，划分结束.
- 从规模 1到 $n/2$，陆续归并被排好序的两个子数组. 每归并一次，数组规模扩大一倍，直到原始数组.

# 二分归并排序时间复杂度分析

假设$n$为2的幂，二分归并排序最坏情况下时间复杂度

$$W(n) = 2W(n/2) + n-1$$

$$W(1) = 0$$

可以解出

$$W(n) = n\log n - n + 1$$

# Hanoi塔的递归算法

算法 **Hanoi** $(A, C, n)$ // $n$个盘子$A$到$C$

1. **if** $n$=1 **then move** $(A, C)$ //1个盘子$A$到$C$
2. **else Hanoi** $(A, B, n-1)$
3. **move** $(A, C)$
4. **Hanoi** $(B, C, n-1)$

设 $n$个盘子的移动次数为 $T(n)$

$$T(n) = 2\,T(n-1) + 1,$$
$$T(1) = 1,$$
$$T(n)=2^n-1$$

# 算法设计思想

- 将原问题归结为规模为 $n$-1 的2个子问题.
- 继续归约，将原问题归结为规模为 $n$-2 的 4 个子问题. 继续…， 当子问题规模为1 时，归约过程截止.
- 从规模 1到 $n$-1，陆续组合两个子问题的解. 直到规模为$n$.

# 小结

通过几个例子展示分治算法的特点：

- 将原问题归约为规模小的子问题，子问题与原问题具有相同的性质.
- 子问题规模足够小时可直接求解.
- 算法可以递归也可以迭代实现.
- 算法的分析方法：递推方程.