内部资料，请勿外传。

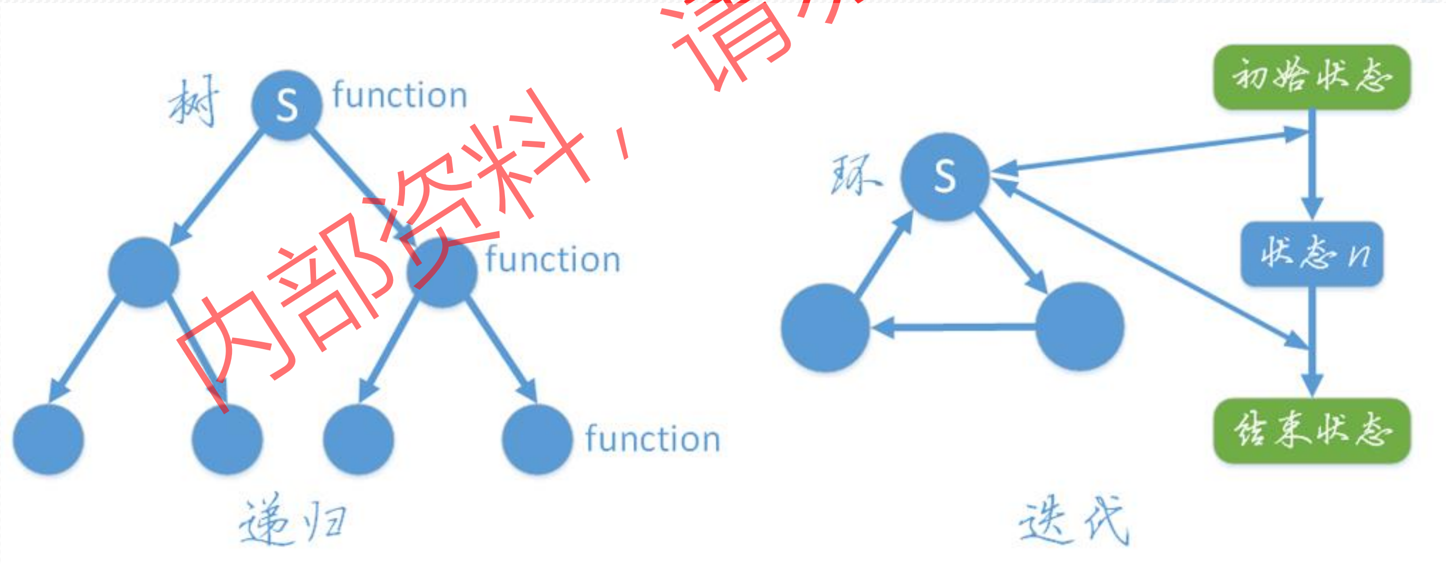
分治算法

张里博

lbzhang@swu.edu.cn

递归与迭代

- 递归是一个树结构，包含“递推”（拆分和重复调用） 和 “回归”的过程，当“递推”到达截止条件时开始“回归”。
- 迭代是一个环结构，从初始状态开始，每次迭代都遍历这个环，并更新状态，多次迭代直到到达结束状态



递归：重复调用函数自身实现循环，即A重复调用A；

迭代：每一次迭代的结果会作为下一次迭代的初始值，即A重复调用B；

内部资料，请勿外传。

递归与迭代

已知：1!=1; n!=n* (n-1)!

分别采用递归和迭代计算n!（n≥1）

内部资料，请勿外传

factorial_rec(n)

```
1.  if  n==1  then
2.      return 1;
3.  else
4.      return n*factorial_rec(n-1);
5.  end
```

factorial_ite(n)

```
1.  product←1;
2.  for i ← 2 to n do
3.      product← product*i;
4.  end
5.  return product;
```

递归三板斧：

1. 起名字；2. 截止条件；3. 递归调用

迭代：

自底向上，从小到大，逐步逼近

目录

部资料，请勿外传。

PART 01

分治策略的基本思想

内部资料，请勿外传。

◆分治策略（Divide and Conquer）思想：

凡治众如治寡，分数是也。 ----孙子兵法

- 将一个难以直接解决的大问题分割拆分成若干规模更小的问题，以便分而治之，逐个击破。

- 将求解一个规模很大的问题转化为**若干同类型的小规模问题**，如此不断分拆，直至问题的规模**足够小**，可以直接求解，然后**自底向上**得到原问题的解。

内部资料，请勿外传。

◆分治策略需要满足的条件：

- 1.子问题与原始问题的性质一样；
- 2.子问题之间可彼此独立地求解；
- 3.递归或迭代停止时（最小规模）子问题可直接求解。

内部资料，请勿外传。

Divide-and-Conquer*(P)*

1. *if* $|P| \leq c$ *then*
2. *return S(P);* //解决最小规模的问题
3. *else*
4. *divide P into* $P_1, P_2, \ldots, P_k$; //分解问题
5. *for i* $\leftarrow$ *1 to k do*
6. $y_i \leftarrow$ *Divide-and-Conquer*(P_i) ; //递归求解子问题
7. *end*
8. *return Merge*$(y_1, y_2, \ldots, y_k)$; //归并子问题的解
9. *end*

内部资料，请勿外传。

分治策略的基本思想

◆分治算法时间复杂度的递推方程：

$$\begin{cases} W(n) = W(|P_1|) + W(|P_2|) + \ldots + W(|P_k|) + f(n) \\ W(c) = C \end{cases}$$

- $P_1, P_2, \ldots, P_k$为划分后产生的子问题；
- $f(n)$为划分子问题（划分）及将子问题的解综合到原问题解（归并）的总工作量；
- 规模为c的最小规模子问题的工作量为C；

拆分为多少个？每个问题的规模为多少？
没有明确的答案，要根据实践经验和具体场景来决定。
一般问题规模大致相等，算法效率比较高。

内部资料，请勿外传。

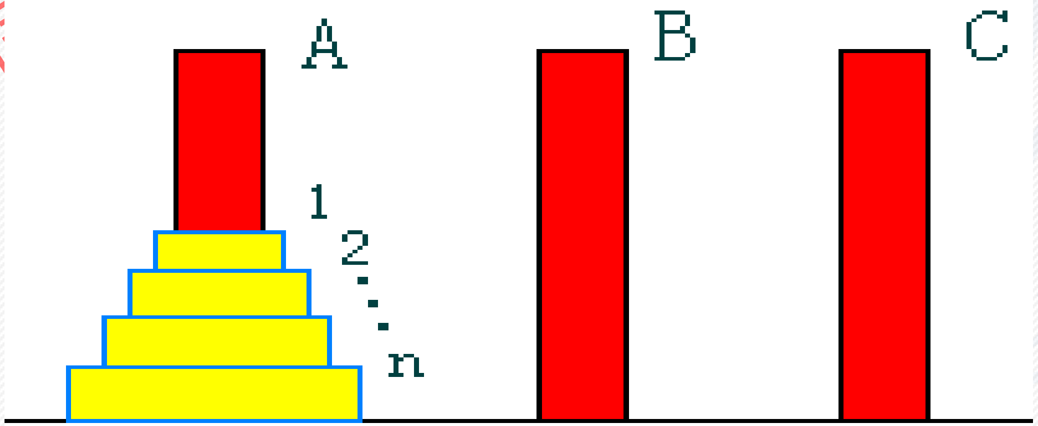
PART 1.2

Hanoi塔问题

内部资料，请勿外传。

Hanoi塔问题

- 已知：有A， B， C是3个塔座，在塔座A上有一叠共n个圆盘，从小到大叠在一起，分别编号为1,2,……,n；
- 求：将塔座A上的这一叠圆盘移到塔座B上，并仍按原顺序叠置。



- 在移动圆盘时应遵守的规则：

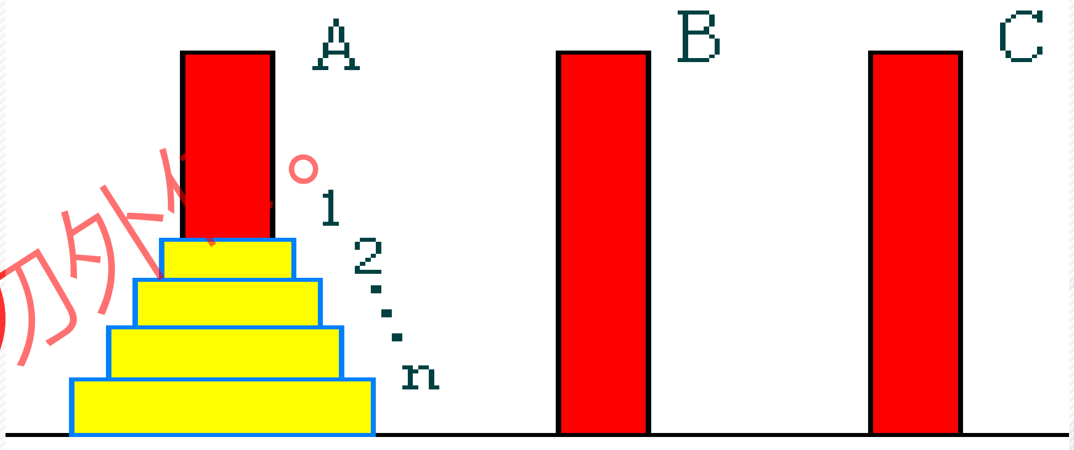
➢ 规则1：每次只能移动1个圆盘；

➢ 规则2：任何时刻都不允许大的圆盘在较小的圆盘之上；

内部资料，请勿外传。

Hanoi塔问题

当n=1时，只要将圆盘从塔座A直接移至塔座B上即可；



- 当n＞1时

➤首先，将n-1个较小的圆盘依照规则从塔座A移至塔座C；
➤然后，将剩下的最大圆盘从塔座A移至塔座B；
➤最后，再将n-1个较小的圆盘依照规则从塔座C移至塔座B。

- 综上，n个圆盘的移动问题可拆分为2次n-1个圆盘的移动问题，这又可以递归地用上述方法来做。

$$T(n) = 2\,T(n-1) + 1，\ T(1) = 1;$$

内部资料，请勿外传。

算法的正确性

- 要求（分治算法需满足的条件）：

1.子问题与原始问题的性质完全一样；

➢ 圆盘移动满足的规则一致，只有数量不同

2.子问题之间可彼此独立地求解；

➢ 从A挪到C，从C挪到B

3.递归或迭代停止时（最小规模）子问题可直接求解。

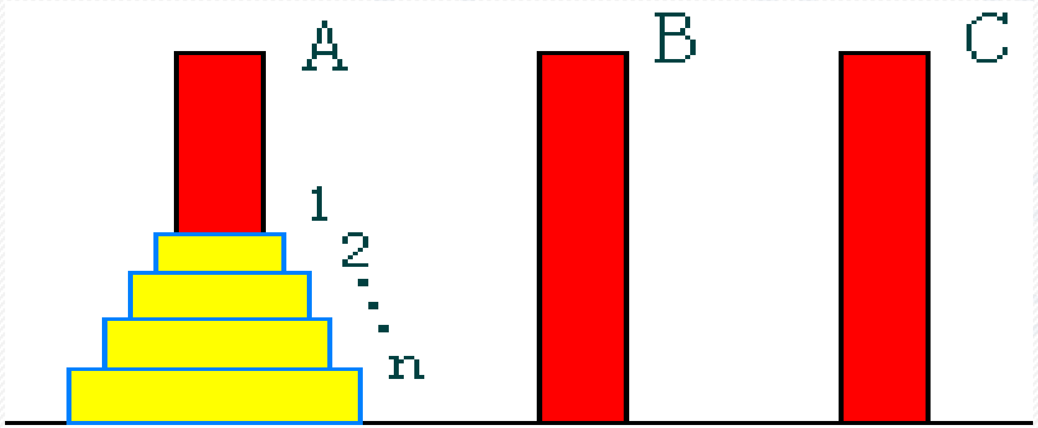
➢ 1个圆盘可以直接挪动

内部资料，请勿外传。

Hanoi塔问题

算法1.3 Hanoi(A,B,n) // 将A的n个盘子移到B

```
if n==1 then
    move (A,B);            //将A的1个盘子移到B
else
    Hanoi(A,C,n–1);
    move(A,B);
    Hanoi(C,B,n–1);
end
```



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

内部资料，请勿外传。

PART 1.3

二分查找算法

内部资料，请勿外传。

二分查找算法

- Although the basic idea of binary search is comparatively straightforward, the details can be surprisingly tricky.

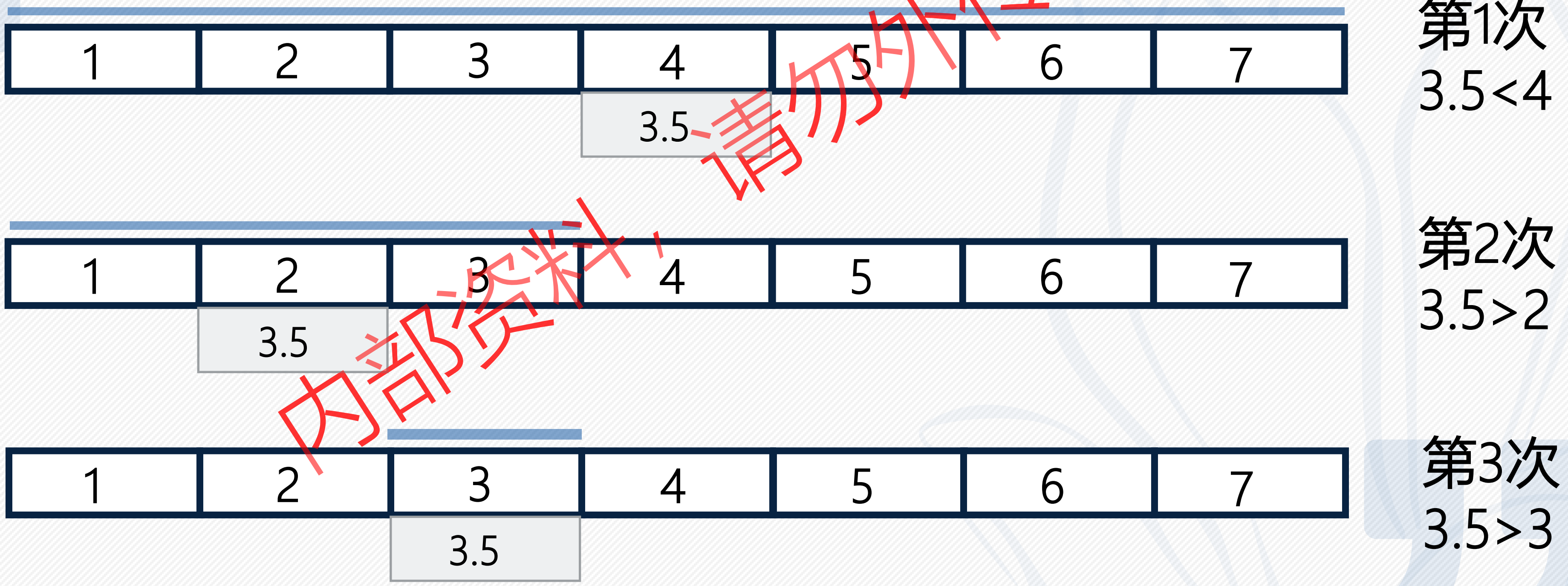
➢ 问题：在一个有序数组（元素从小到大排列）中查找是否存在元素的值等于x

➢ 算法思想：每次找到中间元素（中位数），比较其与x大小关系；如不等，则舍弃一半元素。

Knuth：第一个二分查找算法1946年出现，第一个完全正确的二分检索算法1962年出现

内部资料，请勿外传。

二分检索运行实例：x=3.5



内部资料，请勿外传。

自然语言描述：

- 1. 寻找中间元素位置mid=⌊(left + right)/ 2⌋（向下取整）；
- 2. 比较mid位置元素与x的大小，如果相等就返回；如果元素大于x，则继续在数组中1到right = mid – 1区域查找；如果小于x，则继续数组中left = mid + 1至最后一个元素区域查找；
- 3. 重复1和2，直到区间缩小到只有一个元素；
- 4. 如果该元素等于x，则返回下标；否则，查找失败，返回0。

内部资料，请勿外传。

算法的正确性

- 要求（分治算法需满足的条件）：

1.子问题与原始问题的性质完全一样；
- ➢ 规模减半的有序数组中查找x

2.子问题之间可彼此独立地求解；
- ➢ 每次只拆分出一个子问题

3.递归或迭代停止时（最小规模）子问题可直接求解。
- ➢ 一个元素与x直接比较就能判断

内部资料，请勿外传。

二分查找算法

$$\begin{cases} W(n) = W\left(\left\lfloor \dfrac{n}{2} \right\rfloor\right) + 1 = W\left(\dfrac{n}{2}\right) + 1 \\ W(1) = 1 \end{cases}$$

算法2.1 BinarySearch(T, x)

输入：已排好序的数组 $T[1\cdots n]$；数 x

输出：如果 x 在 T 中，输出下标 j；否则输出0

```
l←1; r←n;
while l≤r do
    m←⌊(l+r)/2⌋;
    if T[m]==x  then
        return m;      // x恰好等于中位元素
    elif T[m]>x  then
        r ← m–1;
    else
        l←m+1;
    end
end
return 0;
```

首先进行顶层设计，之后关注底层实现

```
l←1; r←n;
while l≤r do
    m←⌊(l+r)/2⌋;
    if T[m]==x  then
        break;
    elif T[m]>x  then
        r ← m–1;
    else
        l←m+1;
    end
end
if T[m]!=x  then
    m←0;
end
return m;
```

规模减半的子问题

a_k是奇数: $a_{k+1} = (a_k-1)/2$;

$a_k>2$的偶数: $a_{k+1} = a_k/2$;

或者 $a_{k+1} = a_k/2 - 1$;

内部资料，请勿外传。

假设数组中共有$n=2^k-1$个元素

x在数组T中，最多需要比较多少次？

x不在数组T中，需要比较多少次？

- 已知总共有$n = 2^k$-1个元素，最坏情况下需要比较多少次？
- 每次比较前有：

$$a_1 = 2^k - 1$$

$$a_2 = \frac{a_1 - 1}{2} = 2^{k-1} - 1$$

$$a_3 = \frac{a_2 - 1}{2} = 2^{k-2} - 1$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$a_k = \frac{a_{k-1} - 1}{2} = 2^1 - 1 = 1$$

a_k是奇数: $a_{k+1} = (a_k-1)/2$;

$a_k>2$的偶数:$a_{k+1} = a_k/2$;

或者$a_{k+1} = a_k/2 - 1$;

内部资料，请勿外传。

二分查找算法

X与数组T的关系，有2n+1种情况

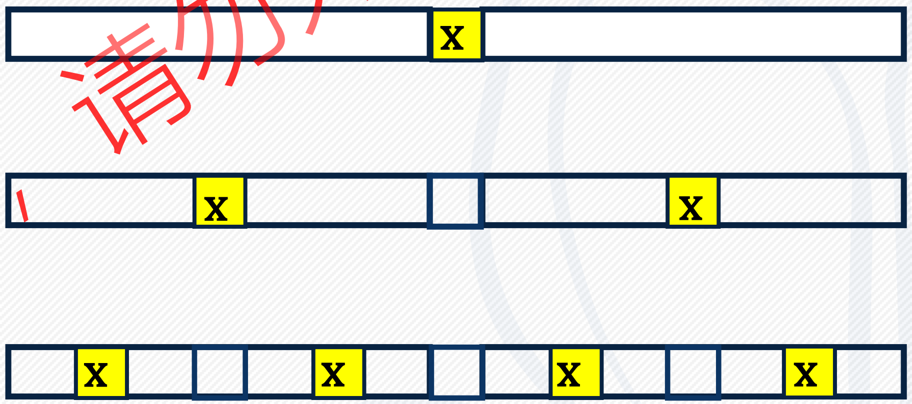
x=T[1]
x=T[2]
x=T[3]
…
x=T[n-1]
x=T[n]

x在T中

X<T[1]
T[1]<x<T[2]
…
T[n-1]<x<T[n]
T[n]<x

x不在T中

比较t次的输入的情况



比较1次：1个

比较2次：2个

比较3次：4个

当**t=1,2,…,k-1**，比较t次的输入总共有 $\mathbf{2^{t-1}}$个；
当**t=** k，比较k次的输入有 $\mathbf{2^{k-1}+n+1}$ 个。

内部资料，请勿外传。

平均时间复杂度

假设 $n=2^k-1$，且各种输入x发生的概率相等

$$A(n)=\frac{1}{2n+1}\left[\sum_{t=1}^{k-1}t2^{t-1}+k\left(2^{k-1}+n+1\right)\right]$$

内部资料，请勿外传。

$$\sum_{t=1}^{k}t2^{t-1}=\sum_{t=1}^{k}t\left(2^{t}-2^{t-1}\right)$$

拆项

$$=\sum_{t=1}^{k}t2^{t}-\sum_{t=1}^{k}t2^{t-1}=\sum_{t=1}^{k}t2^{t}-\sum_{t=0}^{k-1}(t+1)2^{t}$$

变限

$$=\sum_{t=1}^{k}t2^{t}-\sum_{t=0}^{k-1}t2^{t}-\sum_{t=0}^{k-1}2^{t}$$

拆项

$$=k2^{k}-\left(2^{k}-1\right)=(k-1)2^{k}+1$$

含弘光大 继往开来

平均时间复杂度

假设 $n=2^k-1$，且各种输入x发生的概率相等

$$A(n)=\frac{1}{2n+1}\left[\sum_{t=1}^{k-1}t2^{t-1}+k\left(2^{k-1}+n+1\right)\right]$$

$$=\frac{1}{2n+1}\left[\sum_{t=1}^{k}t2^{t-1}+k(n+1)\right]$$

$$=\frac{1}{2n+1}\left[(k-1)2^k+1+k(n+1)\right]$$

$$=\frac{k2^k-2^k+1+k2^k}{2^{k+1}-1}=O(k)=O(\log n)$$

内部资料，请勿外传。

二分查找算法

a_k是奇数: $a_{k+1}= (a_k-1)/2$;
$a_k>2$的偶数:$a_{k+1}= a_k/2$;
或者$a_{k+1}= a_k/2 - 1$;

- 假设二分查找法循环了k次即停止，且第k次查找的区间长度为1，问原数组最多有多少个元素？
- 每次比较前有：

$$a_k = 1$$

$$a_{k\text{-}1} = 2a_k + 2 = 2a_k + 2$$

$$a_{k\text{-}2} = 2a_{k\text{-}1} + 2 = 2(2a_k + 2) + 2 = 2^2 a_k + 2^2 + 2$$

$$a_{k\text{-}3} = 2a_{k\text{-}2} + 2 = 2(2^2 a_k + 2^2 + 2^1) + 2 = 2^3 a_k + 2^3 + 2^2 + 2^1$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$a_1 = 2^{k-1} a_k + 2^{k-1} + \cdots + 2^1 = 2^{k-1} + \frac{1-2^{k\text{-}1+1}}{1-2} \text{-}1 = 2^{k\text{-}1} + 2^k - 2 = n$$

内部资料，请勿外传。

含弘光大 继往开来

课后作业，下次课提问

■1. 假设二分查找法循环到了k次，剩余元素个数为a_k，问原数组最多有多少个元素？

■2. 假设二分查找法比较了k次即停止，且第k次查找的区间长度为1，问原数组最少有多少个元素？

■3. **归并排序的思想、递推方程**

内部资料，请勿外传。

■1. 二分查找法

每次比较中位数与x大小；如不等，则舍弃中间元素和一半元素；

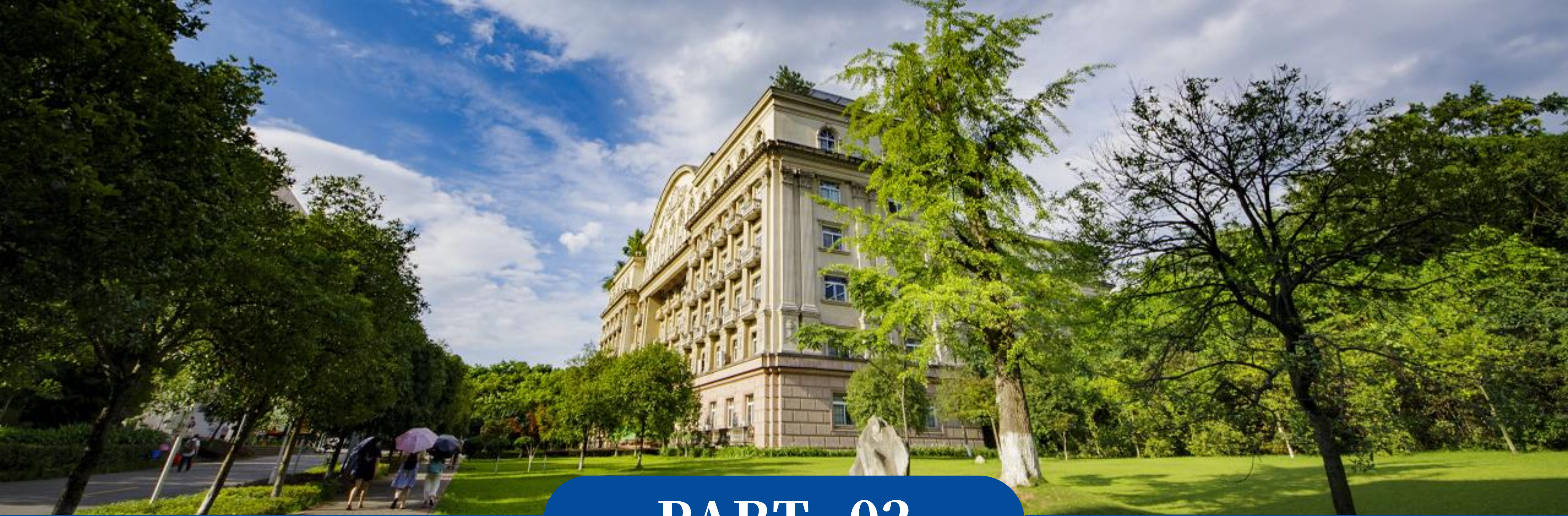
■2. Hanoi塔问题

将挪动n个圆盘的问题转化为2个n-1个圆盘挪动的问题；

■3. 分治算法的一般描述

分解，递归求解，归并。

内部资料，请勿外传。



PART 02

内部

分治算法的时间复杂度

■算法时间复杂度的递推方程：

$$\begin{cases} W(n) = W(|P_1|) + W(|P_2|) + \ldots + W(|P_k|) + f(n) \\ W(c) = C \end{cases}$$

内部资料，请勿外传。

■ P_1，P_2，…，P_k为划分后产生的子问题；

■f(n)为划分子问题（划分）及将子问题的解综合到原问题解（归并）的总工作量；

■规模为c的最小子问题的工作量为C；

■分治策略的算法分析工具：求解递推方程（求出解析式，即通项公式）

■两类递推方程 **方程1：** $T(n)=\sum_{i=1}^{k}a_iT(n-i)+f(n)$

方程2： $T(n)=T\left(\frac{n}{b}\right)+d(n)$

■求解方法：

■**方程1：迭代法、递归树**

■**方程2：迭代法、换元迭代法、递归树、主定理**

内部资料，请勿外传。

Hanoi塔问题时间复杂度满足：

$$\begin{cases} T(n) = 2T(n-1)+1 \\ T(1)=0 \end{cases}$$

二分查找最坏情况下时间复杂度$W(n)$满足：

$$\begin{cases} W(n) = W(\frac{n}{2})+1 \\ W(1)=1 \end{cases}$$

二分归并排序最坏情况下时间复杂度$W(n)$满足：

$$\begin{cases} W(n) = 2W(\frac{n}{2})+n-1 \\ W(1)=0 \end{cases}$$

内部资料，请勿外传。

内部资料，请勿外传。

PART 2.2

迭代法

- 几个有用的结果 $\sum_{k=1}^{n} a_k = \frac{n(a_1 + a_n)}{2}$

$$\sum_{k=0}^{n} aq^k = \frac{a(1-q^{n+1})}{1-q}, \quad \sum_{k=0}^{n} x^k = \frac{1-x^{n+1}}{1-x}$$

$$\sum_{k=1}^{n} \frac{1}{k} = \ln n + O(1)$$

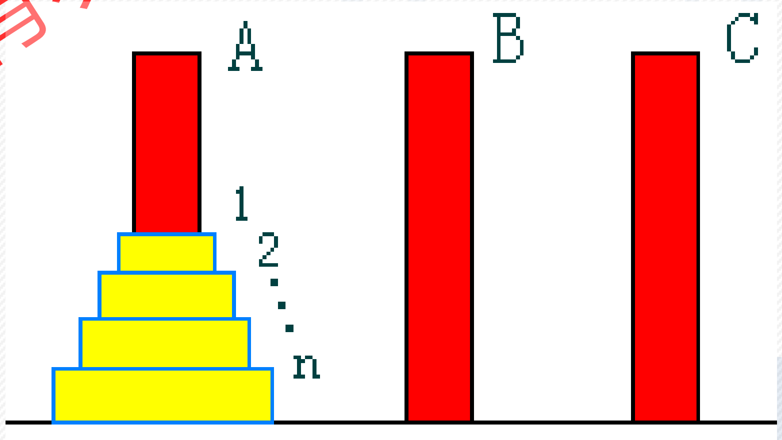
- 和的上界 $\sum_{k=1}^{n} a_k \le n a_{\max}$
- 假设存在常数 $r<1$，使得对一切 $k \ge 0$，$\frac{a_{k+1}}{a_k} \le r$ 成立，则

$$\sum_{k=0}^{n} a_k \le \sum_{k=0}^{\infty} a_0 r^k = a_0 \sum_{k=0}^{\infty} r^k = \frac{a_0}{1-r}$$

内部资料，请勿外传。

算法1.3 Hanoi(*A*,*B*,*n*) // 将*A*的*n*个盘子移到*B*

1. *if n*==1 then
2. move (*A*,*B*); // 将A的1个盘子移到*B*
3. *else*
4. Hanoi(*A*,*C*,*n*–1);
5. move(*A*,*B*);
6. Hanoi(*C*,*B*,*n*–1) ;
7. *end*



$T(n) = 2\,T(n-1) + 1$， $T(1) = 1$; 迭代解得 $T(n)$的解析式

内部资料，请勿外传。

■Hanoi塔

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

内部资料，请勿外传。

$$T(1) = 1$$

$$T(2) = 2T(1) + 1$$

$$T(3) = 2T(2) + 1 = 2(2T(1) + 1) + 1 = 2^2T(1) + \underline{2 + 1}$$

$$T(4) = 2T(3) + 1 = 2(2^2T(1) + 2^1 + 2^0) + 1 = 2^3T(1) + \underline{2^2 + 2^1 + 1}$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$T(n) = 2^{n-1}T(1) + 2^{n-2} + \cdots + 1 = 2^{n-1} + 2^{n-2} + \cdots + 1 = \frac{1 - 2^{n-1+1}}{1-2} = 2^n - 1$$

PART 2.3

换元迭代法

内部资料，请勿外传。

换元迭代法

■将对n的递推式换成对其他变元k的递推式

■对k直接迭代

■将求解关于k的函数，转换成关于n的函数

内部资料，请勿外传。

二分查找法

$$\begin{cases} W(n) = W(\frac{n}{2}) + 1 \\ W(1) = 1 \end{cases}$$

直接迭代？

换元，假设 $n = 2^k$

$$\begin{cases} W(2^k) = W(2^{k-1}) + 1 \\ W(1) = W(2^0) = 1 \end{cases}$$

$$\begin{aligned} W(n) &= W(2^{k-1}) + 1 \\ &= [W(2^{k-2}) + 1] + 1 = W(2^{k-2}) + 2 \\ &= [W(2^{k-3}) + 1] + 2 = W(2^{k-3}) + 3 \\ &= \ldots \\ &= W(2^0) + k \\ &= k + 1 \\ &= \log_2 n + 1 \end{aligned}$$

内部资料，请勿外传。

二分归并排序

算法1.5 MergeSort(A, p, r)

输入：A[p..r], $1 \le p \le r \le n$；

输出：升序排列的数组A；

1. if $p<r$ then

2. $q \leftarrow \lfloor (p+r)/2 \rfloor$; **// 对半划分**

3. MergeSort(A,p,q); **// 子问题1**

4. MergeSort($A,q+1,r$); **// 子问题2**

5. Merge(A,p,q,r); **// 归并**

6. end

最坏情况下的时间复杂度

$$\begin{cases} W(n) = 2W(\frac{n}{2}) + n - 1 \\ W(1) = 0 \end{cases}$$

内部资料，请勿外传。

二分归并排序

$$n = 2^k, \begin{cases} W(2^k) = 2W(2^{k-1}) + 2^k - 1 \\ W(1) = W(2^0) = 0 \end{cases}$$

$$\begin{aligned}
W(n) &= 2W(2^{k-1}) + 2^k - 1 \\
&= 2[2W(2^{k-2}) + 2^{k-1} - 1] + 2^k - 1 = 2^2W(2^{k-2}) + 2*2^k \underline{-2-1} \\
&= 2^2[2W(2^{k-3}) + 2^{k-2} - 1] + 2*2^k - 2 - 1 = 2^3W(2^{k-3}) + 3*2^k \underline{-2^2-2-1} \\
&= \ldots \\
&= 2^kW(2^0) + k2^k - (2^{k-1} + 2^{k-2} + \ldots + 2 + 1) \\
&= k2^k - 2^k + 1 \\
&= n\log_2 n - n + 1
\end{aligned}$$

内部资料，请勿外传。

二分归并排序

$$n = 2^k, \begin{cases} W(2^k) = 2W(2^{k-1}) + 2^k - 1 \\ W(2) = W(2^1) = 1 \end{cases}$$

$$W(n) = 2W(2^{k-1}) + 2^k - 1$$

$$= 2[2W(2^{k-2}) + 2^{k-1} - 1] + 2^k - 1 = 2^2 W(2^{k-2}) + 2*2^k \underline{-2-1}$$

$$= 2^2[2W(2^{k-3}) + 2^{k-2} - 1] + 2*2^k - 2 - 1 = 2^3 W(2^{k-3}) + 3*2^k \underline{-2^2-2-1}$$

$$= \ldots$$

$$= 2^{k\text{-}1} W(2^1) + (k\text{-}1)2^k - (2^{k-2} + \ldots + 2 + 1) = k2^k - (2^{k-1} + 2^{k-2} + \ldots + 2 + 1)$$

$$= k2^k - 2^k + 1$$

$$= n\log_2 n - n + 1$$

内部资料，请勿外传。

PART 2.4

递归树

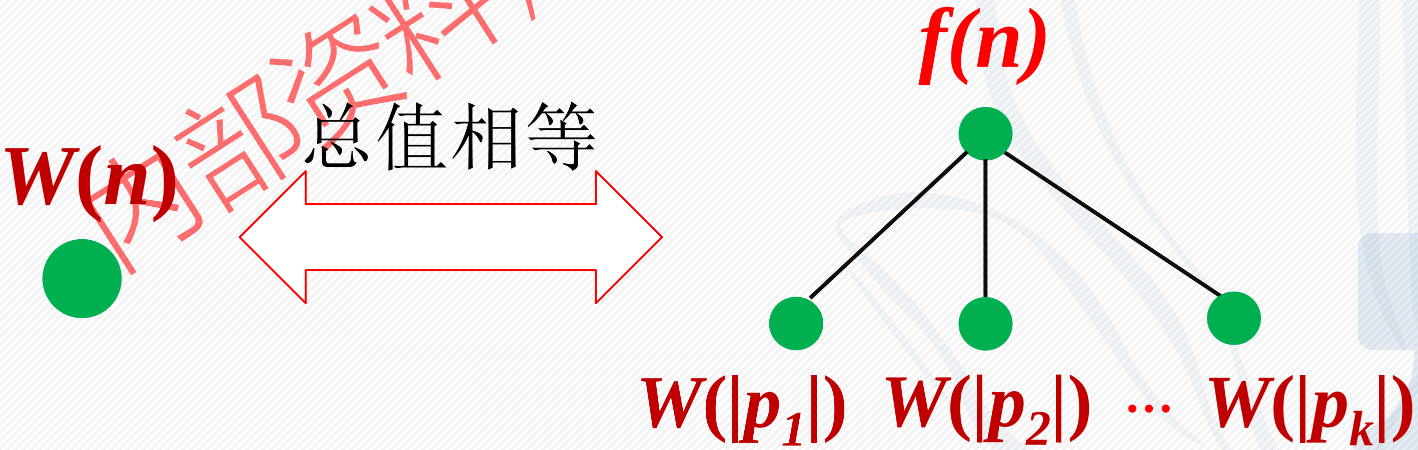
内部资料，请勿外传。

递归树

- 如果有递推方程：

$$W(n)=W(|P_1|)+W(|P_2|)+\ldots+W(|P_k|)+f(n)$$

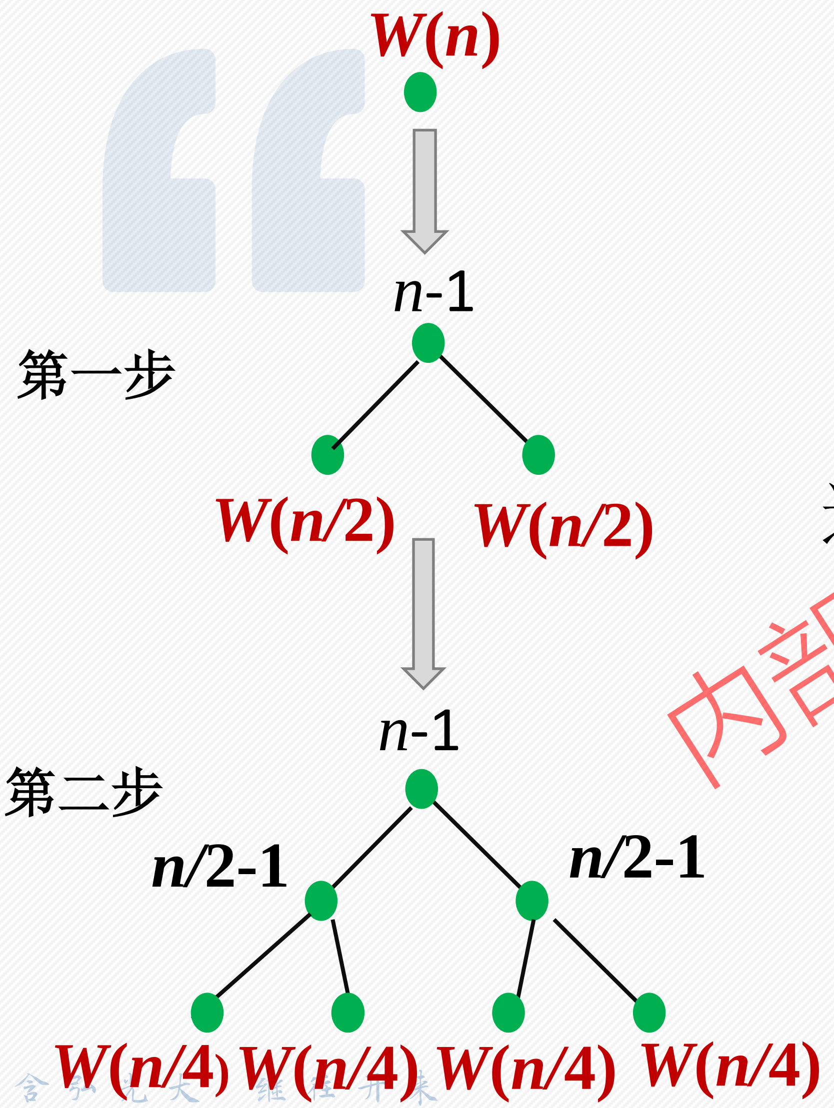
- 其中，$W(|P_1|),\ldots,W(|P_k|)$为函数项，$f(n)$是划分和归结的工作量。



每个叶节点为一个函数项

内部资料，请勿外传。

递归树



二分归并排序

$W(n) = 2W(n/2) + n-1, \quad n=2^k$

$W(2)=1.$

递推至树中无函数项，即递推到最小问题规模

对递归树上的项求和就是$W(n)$的解析式

总共迭代多少次？

$n=2^k$

$n=4$

k−1次迭代

内部资料 请勿外传

$$n-1$$

$$\frac{n}{2}-1 \qquad \frac{n}{2}-1$$

$$\frac{n}{4}-1 \qquad \frac{n}{4}-1 \qquad \frac{n}{4}-1 \qquad \frac{n}{4}-1$$

……

1 1 1 1 1……………………………………1 1 1

$n-1$

$n-2$

$n-4$

$2^{k-1}=n-2^{k-1}$

k−1次迭代

对递归树上的项求和

$W(n) = 2W(n/2) + n-1$，$n=2^k$，$W(2)=1.$

$W(n) = n-1 + n-2 + \ldots + n-2^{k-1}$

$= kn - (2^k-1) \qquad = n\log_2 n - n + 1$

内部资料，请勿外传。

递归树

■1. 能够画出树的结构；

■2. 确定迭代多少次；

■3. 每一层节点之和；

■4. 整棵树节点之和；

内部资料，请勿外传。

PART 2.5

主定理

内部资料，请勿外传。

主定理

定理1.6 设$a\geq 1$, $b>1$为常数，$d(n)$为函数，$T(n)$为非负整数，且

$$T(n)=aT(n/b)+d(n)$$

则有以下结果：

存在

1. 若$d(n)=O(n^{\log_b a-\varepsilon}), \varepsilon>0$，那么$T(n)=\Theta(n^{\log_b a})$
2. 若$d(n)=\Theta(n^{\log_b a})$，那么$T(n)=\Theta(n^{\log_b a}\log n)$
3. 若$d(n)=\Omega(n^{\log_b a+\varepsilon}), \varepsilon>0$，且对某个常数$c<1$和所有充分大的$n$有 $a f(n/b)\leq c f(n)$，那么$T(n)=\Theta(f(n))$

内部资料，请勿外传。

主定理

$$T(n) = aT(n/b) + d(n)$$

d(n)为常数 $T(n) = \begin{cases} \Theta(n^{\log_b a}) & a \neq 1 \\ \Theta(\log n) & a = 1 \end{cases}$ 第一种情况 / 第二种情况

d(n) = cn $T(n) = \begin{cases} \Theta(n) & a < b \\ \Theta(n\log n) & a = b \\ \Theta(n^{\log_b a}) & a > b \end{cases}$ 第一种情况 / 第二种情况 / 第三种情况

$\begin{cases} W(n) = W(\frac{n}{2}) + 1 \\ W(1) = 1 \end{cases}$ 二分查找法 $W(n) = \Theta(\log_2 n)$

$\begin{cases} W(n) = 2W(\frac{n}{2}) + n - 1 \\ W(2) = 1 \end{cases}$ 二分归并排序法 $W(n) = \Theta(n\log_2 n)$

内部资料，请勿外传。

含弘光大 继往开来

解的正确性证明

Hanoi塔问题

证明：下面递推方程的解是$T(n)=2^n-1$

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

- 数学归纳法:
- 证明: n=1, $T(1)=2^n-1=1$;
- 假设n=k时正确, $T(k)=2^k-1$;
- 则n=k+1时正确, $T(k+1)=2\,T(k)+1=2^{k+1}-1$;

内部资料，请勿外传。

小结

■1. 分治算法的递推方程（正确性证明）

■子问题规模常数级减少，倍数级减少；

■2. 迭代法

■不断用右边式子替换左边的项；

■3. 换元迭代法

■不方便直接迭代，将n替换，再进行迭代；

■4. 递归树

■根节点为非函数项，叶节点为函数项，不断用子树替换叶节点，直到递归截至条件，逐层加和，得到总和即为原函数的解析解；

■5. 主定理

■子问题规模倍数级减少。

内部资料，请勿外传。