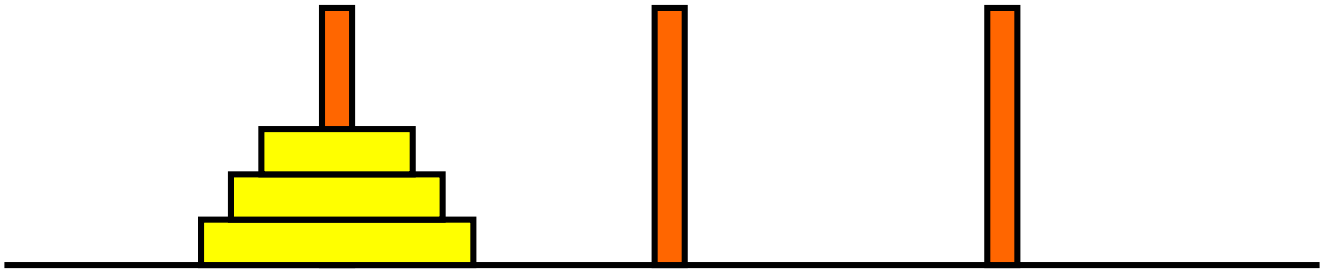
8.3 递推方程的求解与应用

- Hanoi 塔问题
- 递推方程的定义
- 二分归并排序算法的分析
- 快速排序算法的分析
- 递归树
- 分治算法分析的一般公式

Hanoi塔问题

Hanoi塔问题：

从*A*柱将这些圆盘移到*C*柱上去. 如果把一个圆盘从一个柱子移到另一个柱子称作 1 次移动，在移动和放置时允许使用*B*柱，但不允许大圆盘放到小圆盘的上面. 问把所有的圆盘的从*A*移到*C*总计需要多少次移动？



算法设计与分析

算法 Hanoi (*A*,*C*,*n*)　//*把*n*个盘子从*A*移到*C*

1. Hanoi (*A*,*B*,*n*-1)

2. move (*A*,*C*)　　//*把1个盘子从*A*移到*C*

3. Hanoi (*B*,*C*,*n*-1)

移动*n*个盘子的总次数为*T*(*n*)，得到递推方程

$T(n) = 2T(n-1) + 1.$

$T(1)=1.$

可以求得 $T(n)=2n-1$

1秒钟移动1次，64个盘子大约需要5000亿年

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

$$= 2[2T(n-2)+1]+1 \quad (T(n-1)\text{被含}T(n-2)\text{的项替换})$$

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

$$= 2^2[2T(n-3)+1]+2+1 \quad (T(n-2)\text{被含}T(n-3)\text{的项替换})$$

$$= 2^3 T(n-3)+2^2+2+1$$

$$= \ldots$$

$$= 2^{n-1}T(1)+2^{n-2}+2^{n-3}+\ldots+2+1$$

$$= 2^{n-1}+2^{n-2}+2^{n-3}+\ldots+2+1 \quad (\text{代入初值})$$

$$= 2^n - 1 \qquad (\text{等比级数求和})$$

递推方程的定义

定义10.5 设序列$a_0, a_1, \ldots, a_n, \ldots$, 简记为$\{a_n\}$, 一个把$a_n$与某些个$a_i$（$i<n$）联系起来的等式叫做关于序列$\{a_n\}$的递推方程.

实例：

Fibonacci数列： $f_n=f_{n-1}+f_{n-2}$, 初值 $f_0=1, f_1=1$

阶乘数列$\{a_n\}$， $a_n=n!$： $a_n=na_{n-1}, a_1=1$

$$\begin{cases} T(n)=\dfrac{2}{n}\sum_{i=1}^{n-1}T(i)+O(n), & n\geq 2 \\ T(1)=0 \end{cases}$$

求解方法：迭代法

二分归并排序算法

算法Mergesort(*A*,*s*,*t*)　//*排序数组*A*[*s*..*t*]

1. *m*←(*t*-*s*)/2

2. *A*←Mergesort(*A*,*s*,*m*)　//*排序前半数组

3. *B*←Mergesort(*A*,*s*+1,*t*) //*排序后半数组

4. Merge(*A*,*B*)　//*将排好序的*A*,*B*归并

假设$n=2^k$，比较次数至多为$W(n)$

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

归并两个$n/2$大小数组的比较次数为$n-1$

实例

输入：[5, 1, 7, 8, 2, 4, 6, 3]

划分：[5, 1, 7, 8], [2, 4, 6, 3]

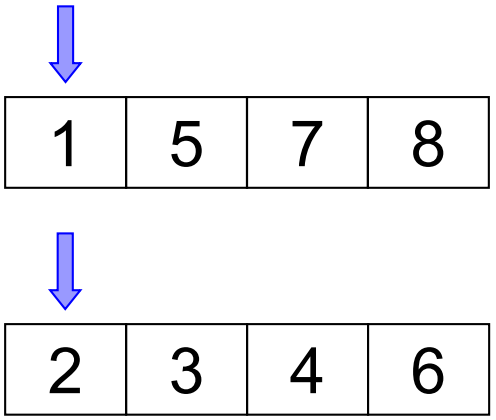
递归排序前半个数组：[5, 1, 7, 8] ⇒ [1, 5, 7, 8]

递归排序后半个数组：[2 ,4, 6, 3] ⇒ [2, 3, 4, 6]

归并：[1, 5, 7, 8] 和 [2, 3, 4, 6]

输出：[1, 2, 3, 4, 5, 6, 7, 8]

归并过程



求解递推方程

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

归纳法验证解

- n=1代入上述公式得

$$W(1)=1\ \log 1-1+1=0,$$

符合初始条件.

- 假设对于任何小于n的正整数t，$W(t)$都是正确的，将结果代入原递推方程的右边得

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

$$=2(2^{k-1}\ \log 2^{k-1}-2^{k-1}+1)+2^k-1$$

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

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

快速排序算法

算法 Quicksort(*A*,*p*,*r*) //*排序数组*A*[*p*..*r*]

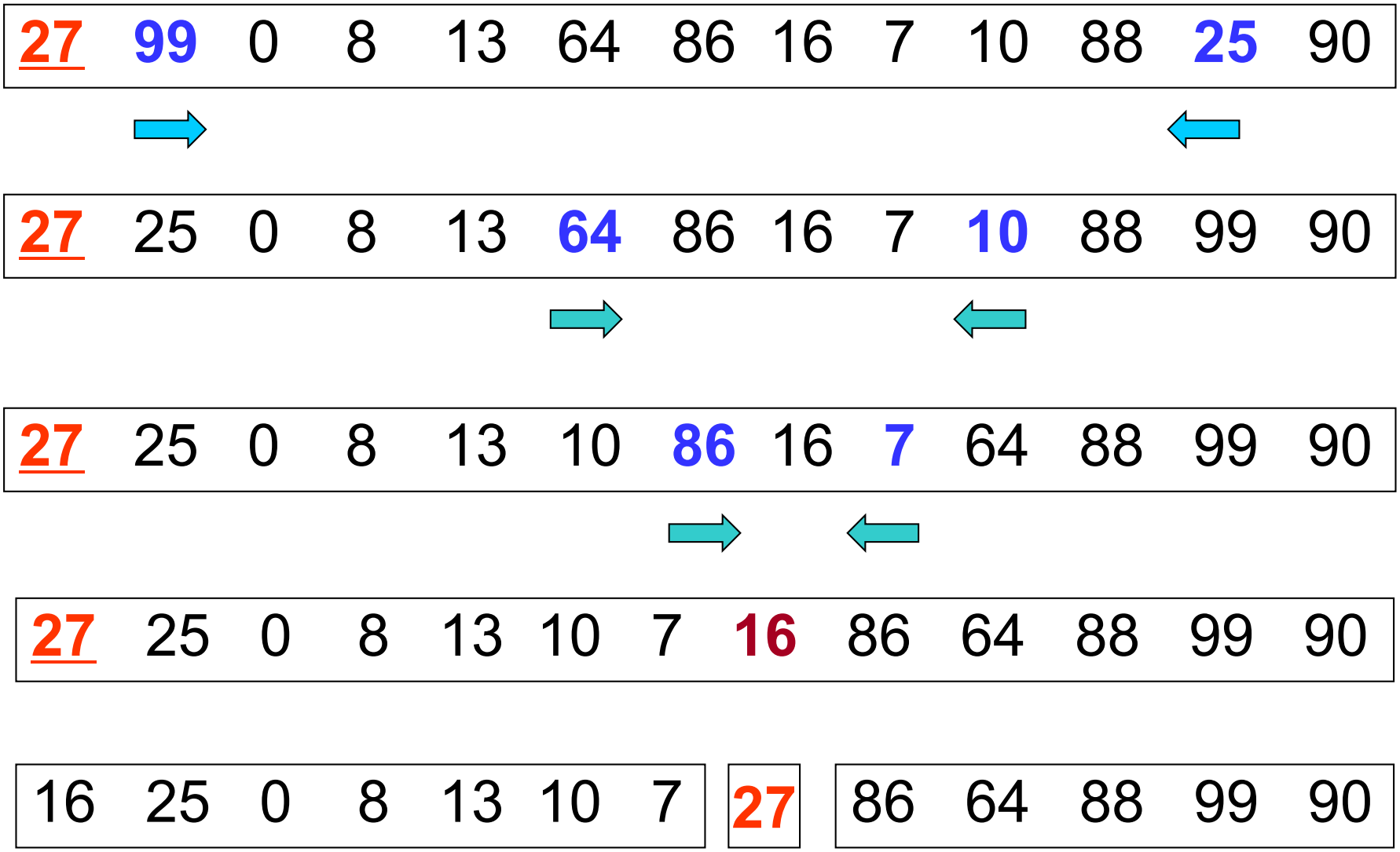
输入：数组*A*[*p*..*r*]

输出：排好序的数组*A*

```
1. if p < r
2.   then q←Partition(A, p, r)  //*以A[p]为准划分A
3.        A[p]↔A[q]             //*A[p]与A[q]交换
4.        Quicksort(A,p,q-1)    //*对子数组递归排序
5.        Quicksort(A,q+1,r)
```

划分过程

```
Partition(A,p,r)
    1.  x ← A[p]
    2.  i ← p
    3.  j ← r+1
    4.  while true do
    5.      repeat j ← j –1
    6.      until A[ j ]<x  //*右边第1个比A[p]小的A[j]
    7.      repeat i ←i +1
    8.      until A[ i ]>x  //*左边第1个比A[p]大的A[i]
    9.      if i < j
    10.         then A[ i ] ↔ A[ j ]  //*交换A[j]与A[i]
    11.         else return j
```


27 99 0 8 13 64 86 16 7 10 88 25 90
27 25 0 8 13 64 86 16 7 10 88 99 90
27 25 0 8 13 10 86 16 7 64 88 99 90
27 25 0 8 13 10 7 16 86 64 88 99 90
16 25 0 8 13 10 7
27
86 64 88 99 90

平均时间复杂度

- $T(n)$为对数组的各种输入平均做的比较次数
  将输入按照$A[p]$在排好序后的位置分别为1, 2, …, n进行分类. 假设每类输入出现的概率相等
- $A[p]$处位置1，划分后子问题规模分别为0和n-1
  …
  $A[p]$处位置n，划分后子问题规模分别为n-1和0
- n 种输入的平均复杂度为

$$T(n)=\frac{1}{n}[(T(0)+T(n-1))+(T(1)+T(n-2))+...$$

$$+(T(n-1)+T(0))]+O(n)=\frac{2}{n}\sum_{i=1}^{n-1}T(i)+O(n)$$

递推方程求解

$$\begin{cases} T(n) = \dfrac{2}{n}\sum_{i=1}^{n-1} T(i) + O(n), & n \geq 2 \\ T(1) = 0 \end{cases}$$

差消法化简

$$nT(n) = 2\sum_{i=1}^{n-1} T(i) + cn^2 \quad c\text{为某个常数}$$

$$(n-1)T(n-1) = 2\sum_{i=1}^{n-2} T(i) + c(n-1)^2$$

$$nT(n) - (n-1)T(n-1) = 2T(n-1) + O(n)$$

$$nT(n) = (n+1)T(n-1) + O(n)$$

迭代

$$\frac{T(n)}{n+1}=\frac{T(n-1)}{n}+\frac{c}{n+1}$$

c为某个常数

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

$$=\cdots$$

$$=c[\frac{1}{n+1}+\frac{1}{n}+\ldots+\frac{1}{3}+\frac{T(1)}{2}]$$

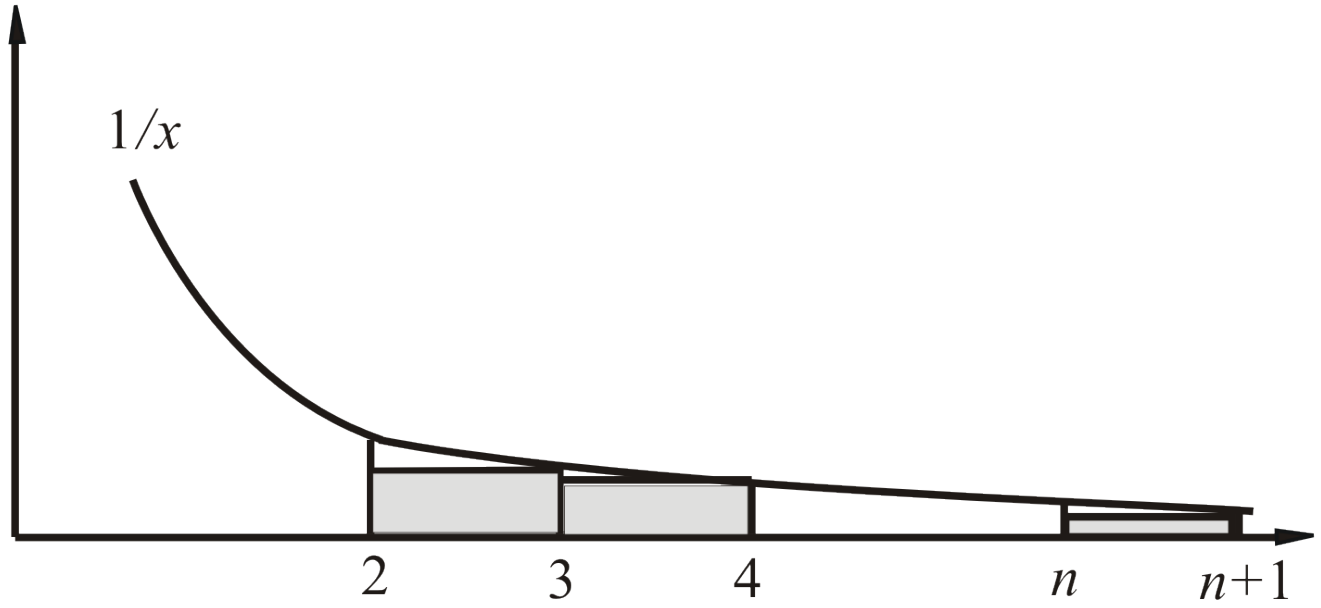
$$=c\left[\frac{1}{n+1}+\frac{1}{n}+\ldots+\frac{1}{3}\right]$$

用积分近似

$$\frac{1}{n+1}+\frac{1}{n}+\ldots+\frac{1}{3}\le\int_2^{n+1}\frac{1}{x}dx=\ln x\Big|_2^{n+1}$$

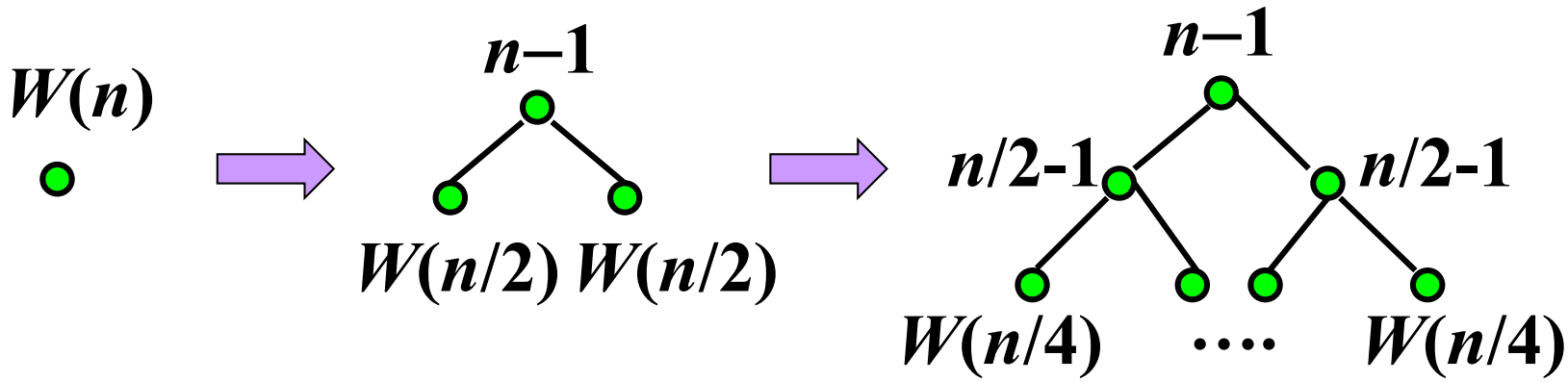
$$=\ln(n+1)-\ln 2=O(\log n)$$

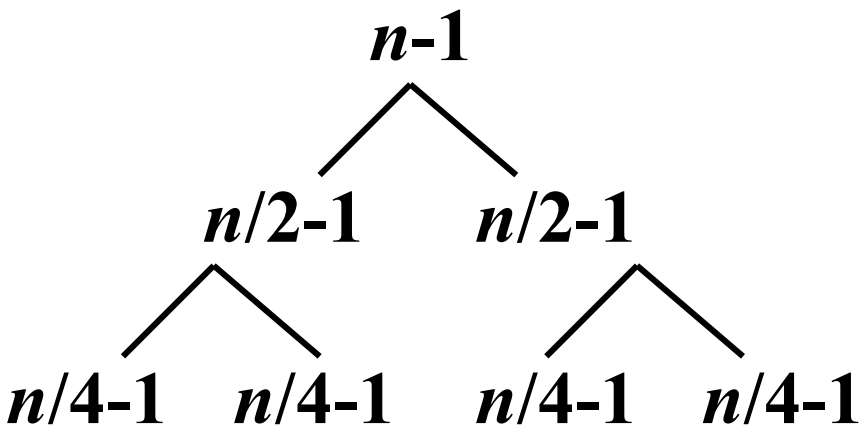
$$T(n)=O(n\log n)$$



递归树

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





n-1

n-2

n-4

n-2^{k-1}

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

$$= nk-(1+2+...+2^{k-1})$$

$$= nk-(2^k-1) = n\log n - n + 1$$

分治算法的常用递推公式

$$\begin{cases} T(n) = aT(n/b) + d(n) & n = b^k \\ T(1) = 1 \end{cases}$$

其中a为子问题个数，n/b为子问题规模，$d(n)$为分解成子问题或组合解的代价

方程的解为：

$$d(n) = c: \quad T(n) = \begin{cases} O(n^{\log_b a}) & a \neq 1 \\ O(\log n) & a = 1 \end{cases}$$

$$d(n) = cn: \ T(n) = \begin{cases} O(n) & a < b \\ O(n \log n) & a = b \\ O(n^{\log_b a}) & a > b \end{cases}$$

迭代

$$T(n) = a^2 T(n/b^2) + ad(n/b) + d(n)$$

$$= \ldots$$

$$= a^k T(n/b^k) + a^{k-1} d(n/b^{k-1}) + a^{k-2}(n/b^{k-2}) + \ldots + ad(n/b) + d(n)$$

$$= a^k + \sum_{i=0}^{k-1} a^i d(n/b^i) \qquad a^k = a^{\log_b n} = n^{\log_b a}$$

Case1 $d(n)=c$

$$T(n) = \begin{cases} a^k + c\dfrac{a^k - 1}{a-1} = O(a^k) = O(n^{\log_b a}) & a \neq 1 \\ a^k + kc = O(\log n) & a = 1 \end{cases}$$

Case2 $d(n)=cn$

$$T(n)=a^k+\sum_{i=0}^{k-1}a^i\frac{cn}{b^i}=a^k+cn\sum_{i=0}^{k-1}(\frac{a}{b})^i$$

$$=\begin{cases} n^{\log_b a}+cn\dfrac{(a/b)^k-1}{a/b-1}=O(n) & a<b \\ n+cnk=O(n\log n) & a=b \\ a^k+cn\dfrac{(a/b)^k-1}{a/b-1}=a^k+c\dfrac{a^k-b^k}{a/b-1}=O(n^{\log_b a}) & a>b \end{cases}$$

应用实例

- 二分归并

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

$a=2$，$b=2$，$d(n)=O(n) \Rightarrow T(n)=O(n\log n)$

- 二分查找

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

$a=1$，$b=2$，$d(n)=O(1) \Rightarrow T(n)=O(\log n)$

例题：关系计数

设A为n元集，A上可定义个不同的二元关系，其中有

(1) 多少个自反的二元关系?

(2) 多少个对称的二元关系?

(3) 多少个反对称的二元关系?

(1) 二元关系个数：2^{n^2}

(2) 自反关系个数：2^{n^2-n}

(3) 对称关系个数：$2^{\frac{n(n+1)}{2}}$

(4) 反对称关系个数：$2^n 3^{\frac{n(n-1)}{2}}$

例题：函数计数

设A、B分别为m元集和n元集，m和n为正整数，则从A到B有多少个函数？

(1) 当m与n满足什么条件时存在单射函数？有多少个?

(2) 当m与n满足什么条件时存在双射函数？有多少个?

有n^m个从A到B的函数，其中

(1) 当$m \le n$时存在单射函数. 单射函数有 P_n^m 个

(2) 当$m=n$时存在双射函数. 双射函数有$n!$个.

P_n^m