

第八章 高级计数技术

Discrete Mathematics

Wei Li

liwei.cs@ncu.edu.cn

Information Engineering School
Nanchang University

May 23, 2022



Outline

- ① 递推关系的应用
- ② 求解线性递推关系
 - 齐次递推关系
 - 非齐次递归关系
- ③ 分治算法和递推关系
- ④ 生成函数



Outline

- 1 递推关系的应用
- 2 求解线性递推关系
 - 齐次递推关系
 - 非齐次递归关系
- 3 分治算法和递推关系
- 4 生成函数



递推关系的应用

讨论两种在算法研究中最重要递推关系。

- 动态规划：将问题分为重叠的子问题，通过递归关系找到子问题的解从而解出原始问题。
- 分而治之：将问题分解为不重叠的子问题，直到这些子问题被直接解决。

这些算法的复杂度可以采用特别的递推关系来分析。



递推关系

- 一个序列的递归定义指定了一个或多个初始的项以及一个由前项确定后项的规则。这个从某些前项求后项的规则就叫作**递推关系**。
- 如果一个序列的项满足递推关系，则这个序列就叫作递推关系的解。



8.1.2 用递推关系构造模型

EXAMPLE 1 兔子和斐波那契数 由意大利数学家莱昂纳多·斐波那契 (Leonardo Fibonacci) 在 13 世纪提出来的兔子数量问题。












新生的对数 (至少两个月大)	已有的对数 (比两个月小)	月	新生的 对数	已有的 对数	总对数
		1	0	1	1
		2	0	1	1
		3	1	1	2
		4	1	2	3
		5	2	3	5
		6	3	5	8
					

图 1 岛上的兔子

8.1.2 用递推关系构造模型

EXAMPLE 1 兔子和斐波那契数 由意大利数学家莱昂纳多·斐波那契 (Leonardo Fibonacci) 在 13 世纪提出来的兔子数量问题。

解：用 f_n 表示 n 个月后的兔子对数。

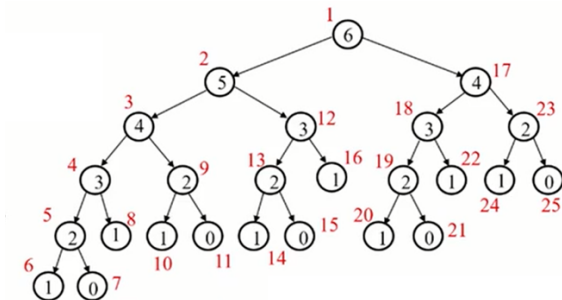
- 在第 1 个月末，岛上兔子的对数是 $f_1 = 1$ 。
- 由于这对兔子在第 2 个月没有繁殖，所以 $f_2 = 1$ 。
- 要计算出 n 个月后岛上的兔子对数，可以将前一个月的数量 f_{n-1} 和新生的兔子对数相加等于 f_{n-2} ，因为每对两个月大的兔子都生出一对新兔子。
- 因此，序列 $\{f_n\}$ 满足递推关系

$$f_n = f_{n-1} + f_{n-2}, n \geq 3 \quad (1)$$

- 所以岛上 n 个月后兔子的对数由第 n 个斐波那契数给出。

斐波那契数列

- 递归实现代价¹: $T(n)=T(n-1)+T(n-2)$



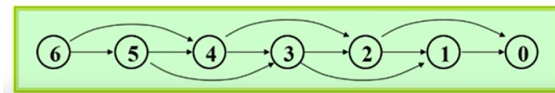
斐波那契数列

- 自顶向下 (DFS)
- 借助额外数组 (查表)

```
2 memo = {}
3 def fib2(n):
4     if n in memo:                # 查表
5         return memo[n]
6     else:
7         if n <= 2:                # 边界条件
8             f = 1
9         else:
10            f = fib2(n-1) + fib2(n-2) # 递归调用
11            memo[n] = f              # 将结果存储于表中
12            return f
```

斐波那契数列

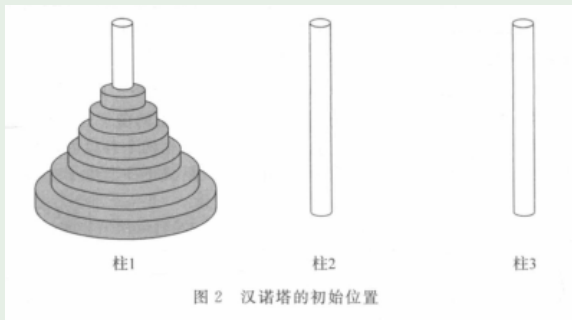
- 自底向上
- 利用拓扑排序



```
14 def fib_bottom_up(n):
15     fib = {}                                # 存储结果的字典
16     for k in range(n+1):
17         if k<=2:                             # 边界条件
18             f = 1
19         else:
20             f = fib[k-1]+fib[k-2]            # 自底向上填表
21         fib[k] = f
22     return fib[n]
```

8.1.2 用递推关系构造模型

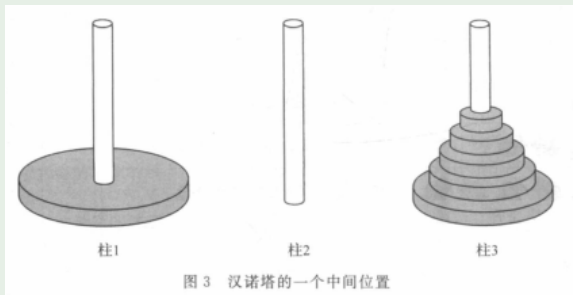
EXAMPLE 2 汉诺塔 每一次把一个盘子从一根柱子移动到另一根柱子，但不允许这个盘子放在比它小的盘子上。



8.1.2 用递推关系构造模型

EXAMPLE 2 汉诺塔

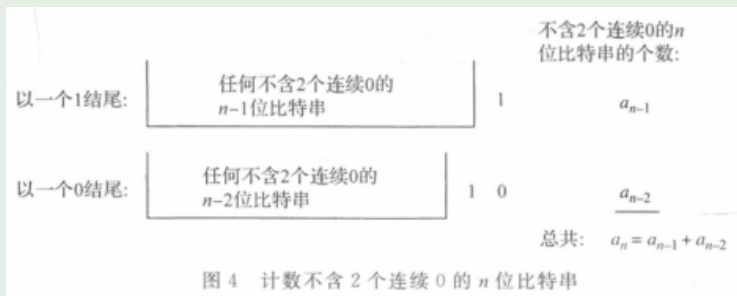
解：用 H_{n-1} 次将上边的 $n-1$ 个盘子移到柱 3，用一次移动把最大的盘子移动到柱 2，再使用 H_{n-1} 次将柱 3 的 $n-1$ 个盘子移到柱 2。



8.1.2 用递推关系构造模型

EXAMPLE 3 计数比特串 对不含 2 个连续 0 的 n 位比特串的个数，找出递推关系和初始条件。有多少个这样的 5 位比特串？

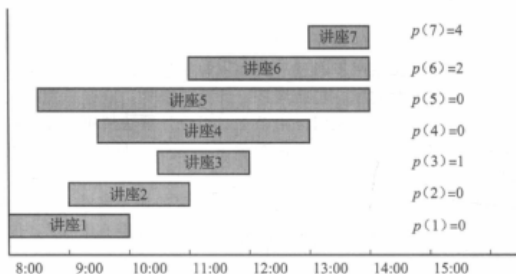
解：设 a_n 表示不含 2 个连续 0 的 n 位比特串的个数。我们假定 $n \geq 3$ ，比特串至少有 3 位， n 位比特串可以分为以 1 结尾的和以 0 结尾的。如图，有 $a_n = a_{n-1} + a_{n-2}$ 。



8.1.3 算法与递推关系

EXAMPLE 6 动态规划实例

设有 n 个讲座，讲座 j 开始于时间 t_j ，结束于时间 e_j ，有 w_j 个学生参与。我们需要规划最大的参与学生人数。

图5 一个展示 $p(h)$ 值的讲座规划

8.1.3 算法与递推关系

算法 1 调度讲座的动态规划算法

Procedure Maximum Attendees(s_1, s_2, \dots, s_n : 讲座的开始时间; e_1, e_2, \dots, e_n : 讲座的结束时间;

w_1, w_2, \dots, w_n : 讲座的参与人数)

将讲座按结束时间排序, 并重新标记讲座, 保证 $e_1 \leq e_2 \leq \dots \leq e_n$.

for $j := 1$ **to** n

if 没有任务 $i (i < j)$ 与任务 j 相兼容;

$p(j) = 0$

else $p(j) := \max\{i \mid i < j \text{ 并且任务 } i \text{ 与任务 } j \text{ 相兼容}\}$

$T(0) := 0$

for $j := 1$ **to** n

$T(j) := \max(w_j + T(p(j)), T(j-1))$

return $T(n)$ ($T(n)$ 是最大的参与人数)

Outline

- ① 递推关系的应用
- ② 求解线性递推关系
 - 齐次递推关系
 - 非齐次递归关系
- ③ 分治算法和递推关系
- ④ 生成函数



8.2.1 线性递推关系

定义

一个常系数的 k 阶线性齐次递推关系是形如

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} \quad (2)$$

的递推关系, 其中 c_1, c_2, \cdots, c_k 是实数, $c_k \neq 0$.

- 线性: 等式右边是序列前若干项的线性组合
- 齐次: 等式右边都是 a_j 的一次项, 每个系数都是常数
- k 阶: a_n 是用之前的 k 项来表示, 即最多与前 k 项有关

8.2.1 线性递推关系

- $P_n = (1.11)P_{n-1}$ 1 阶线性齐次递推关系
- $f_n = f_{n-1} + f_{n-2}$ 2 阶线性齐次递推关系
- $a_n = a_{n-1} + a_{n-2}^2$ 非线性、非齐次
- $H_n = 2H_{n-1} + 1$ 非齐次
- $B_n = nB_{n-1}$ 系数不是常数



8.2.2 求解常系数线性齐次递推关系

- 基本方法是寻找 $a_n = r^n$ 的解, 其中 r 是常数。
- 注意 $a_n = r^n$ 是递推关系 $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$ 的解当且仅当 $r^n = c_1 r^{n-1} + c_2 r^{n-2} + \cdots + c_k r^{n-k}$
- 通过代数运算得到特征方程:
$$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_{k-1} r - c_k = 0$$
- 当且仅当 r 是特征方程的解时, 序列 $\{a_n\}$ 以 $a_n = r^n$ 作为解。
- 特征方程的解称为递推关系的特征根, 可用特征根给出递推关系的所有解的显示表达式。

8.2.2 求解常系数线性齐次递推关系

定理

设 c_1 和 c_2 是实数, 假设 $r^2 - c_1 r - c_2 = 0$ 有两个不相等的根 r_1 和 r_2 , 那么序列 $\{a_n\}$ 是递推关系 $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ 的解, 当且仅当 $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n (n = 0, 1, 2, \dots)$, 其中 α_1 和 α_2 是常数。

8.2.2 求解常系数线性齐次递推关系

EXAMPLE 4 斐波那契数的显示公式

斐波那契数的序列满足递推关系 $f_n = f_{n-1} + f_{n-2}$ 和初始条件 $f_0 = 0, f_1 = 1$, 特征方程 $r^2 - r - 1 = 0$ 的根是 $r_1 = (1 + \sqrt{5})/2$ 和 $r_2 = (1 - \sqrt{5})/2$, 由定理 1 得到

$$f_n = \frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left(\frac{1 - \sqrt{5}}{2} \right)^n \quad (3)$$

Outline

- 1 递推关系的应用
- 2 求解线性递推关系
 - 齐次递推关系
 - 非齐次递归关系
- 3 分治算法和递推关系
- 4 生成函数



8.3.1 分治算法和递推关系

- 我们认识了形如 $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$ 的常系数 k 阶线性齐次递推关系及其求解方法。
- 在计算机科学中，还有另外一类递推关系，它通常产生于对算法进行时间复杂度分析的需求。特别地，在设计分治算法的时候就会产生递推关系。而这种递推关系的表现形式有别于常系数线性齐次递推关系，因而有不同的求解方法。
- 分治算法：将问题划分成规模较小的子问题，然后求得小问题的解从而得到原问题的解。

8.3.2 分治递推关系

- 假设一个递归算法将一个大小为 n 的问题分解成 n 个子问题。
- 假设每个子问题的大小为 n/b .
- 假设 $g(n)$ 是算法处理步中需要额外的操作的总量。
- 那么 $f(n)$ 表示求解规模为 n 的问题所需的运算数, 则得出 f 满足递推关系:

$$f(n) = af(n/b) + g(n) \quad (4)$$

这就叫做分治递推关系。

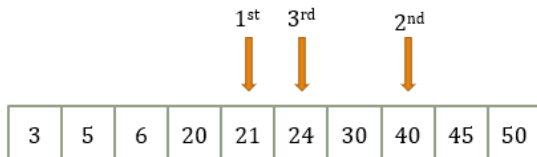
8.3.2 分治递推关系

EXAMPLE 1 二分搜索

如果 $f(n)$ 表示在大小为 n 的序列中搜索一个元素所需要的比较次数，且当 n 是偶数时，则： $f(n) = f(n/2) + 2$

□ 例子：查找“24”

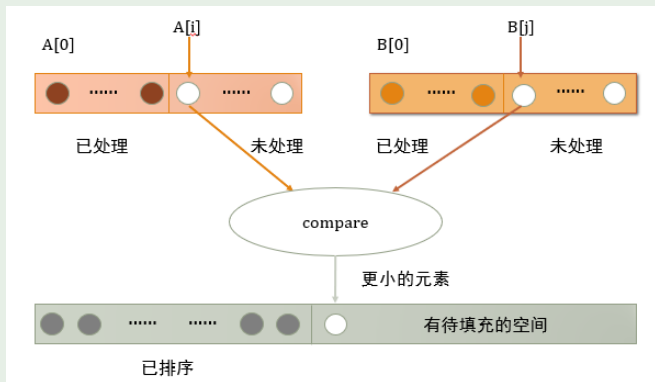
- 有序数组，常数时间寻址
- 折半搜索空间



8.3.2 分治递推关系

EXAMPLE 3 归并排序

对大小为 n 的序列进行排序所需的比较次数不超过
 $M(n) = 2M(n/2) + n$



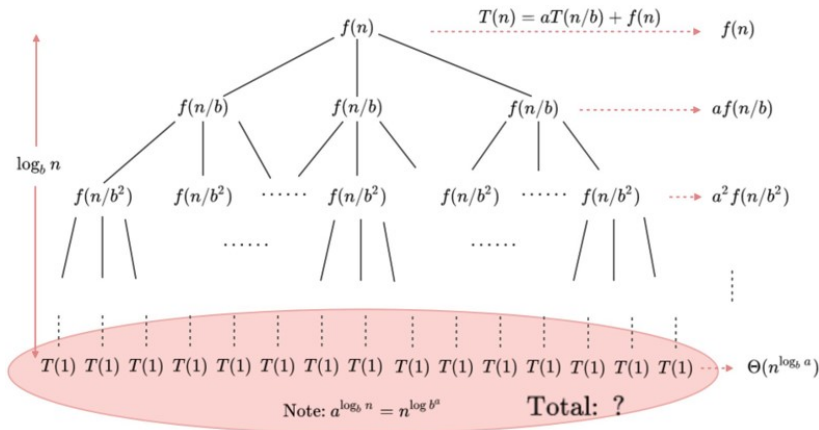
8.3.2 分治递推关系

- 形如 $f(n) = af(n/b) + g(n)$ 的递推关系，可以对满足满足这种递推关系的函数的阶做出估计。

$$\begin{aligned}
 f(n) &= af(n/b) + g(n) \\
 &= a^2 f(n/b^2) + ag(n/b) + g(n) \\
 &= a^3 f(n/b^3) + a^2 g(n/b^2) + ag(n/b) + g(n) \\
 &\vdots \\
 &= a^k f(n/b^k) + \sum_{j=0}^{k-1} a^j g(n/b^j)
 \end{aligned} \tag{5}$$

- $f(n) = a^k f(1) + \sum_{j=1}^{k-1} a^j g(n/b^j)$

8.3.2 分治递推关系



8.3.2 分治递推关系

定理

设 f 是满足递推关系

$$f(n) = af(n/b) + c \quad (6)$$

的增函数, 其中 n 被 b 整除, $a \geq 1$, b 是大于 1 的整数, c 是一正实数, 那么

$$f(n) \text{ 是 } \begin{cases} O(n^{\log_b a}) & a > 1 \\ O(\log n) & a = 1 \end{cases} \quad (7)$$

而且, 当 $n = b^k$ (其中 k 是正整数), $a \neq 1$ 时

$$f(n) = C_1 n^{\log_b a} + C_2 \quad (8)$$

其中 $C_1 = f(1) + c/(a-1)$ 且 $C_2 = -c/(a-1)$.

8.3.2 分治递推关系

定理

设 f 是满足递推关系

$$f(n) = af(n/b) + cn^d \quad (9)$$

的增函数, 其中 $n = b^k$, $a \geq 1$, b 是大于 1 的整数, c 和 d 是实数, 那么

$$f(n) \text{ 是 } \begin{cases} O(n^d) & a < b^d \\ O(n^d \log n) & a = b^d \\ O(n^{\log_b a}) & a > b^d \end{cases} \quad (10)$$

8.3.2 分治递推关系

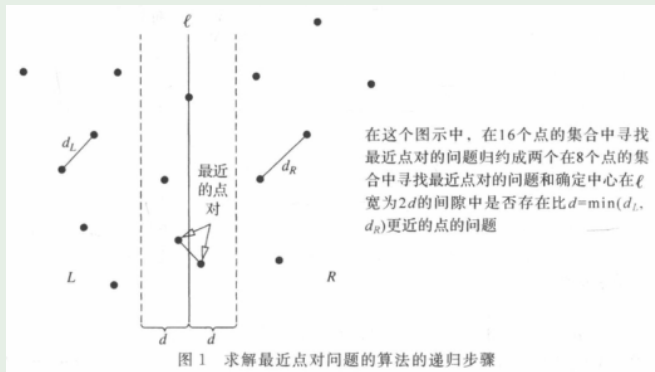
- 二分搜索的复杂度: $f(n) = f(n/2) + 2$
 - $f(n) = O(\log n)$
- 归并排序的复杂度: $M(n) = 2M(n/2) + n$
 - $M(n) = O(n \log n)$



8.3.2 分治递推关系

EXAMPLE 12 最近点对问题

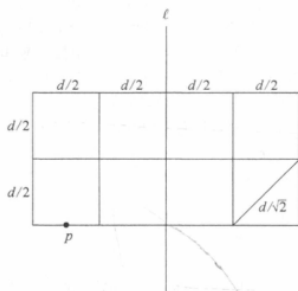
最近的点对的位置有三种可能：两点都在左边区域 L ；两点都在右部区域 R ；一点在左部区域且另一点在右部区域。



8.3.2 分治递推关系

EXAMPLE 12 最近点对问题

至多需要与 d 比较 $7n$ 个距离以找到点之间的最小距离, 满足递推关系: $f(n) = 2f(n/2) + 7n$, 得到 $f(n) = O(n \log n)$



包含 p 在内, 至多8个点可能处在中心在 ℓ 的 $2d \times d$ 的矩形内或者边上, 因为在8个 $(d/2) \times (d/2)$ 的正方形中, 每个内部或边上至多可能存在一个点

图2 说明对带状区域中的每个点至多需要考虑另外7个点

Outline

- ① 递推关系的应用
- ② 求解线性递推关系
 - 齐次递推关系
 - 非齐次递归关系
- ③ 分治算法和递推关系
- ④ 生成函数



生成函数

- 生成函数是表示序列的一种有效方法，它把序列的项作为一个形式幂级数中变量 x 的幂的系数。
- 在一些问题中，我们借助生成函数的概念将一个 \sum 形式的生成函数转化成其他的形式，然后进行函数层次上的操作，能使我们处理问题站在一个新的视角。

8.4.1 生成函数

定义

实数序列 $a_0, a_1, \dots, a_k, \dots$ 的生成函数是无穷级数

$$G(x) = a_0 + a_1x + \dots + a_kx^k + \dots = \sum_{k=0}^{\infty} a_kx^k \quad (11)$$

EXAMPLE 1

- 基于 $a_k = 3$ 的序列 $\{a_k\}$ 的生成函数为 $\sum_{k=0}^{\infty} 3x^k$.
- 基于 $a_k = k + 1$ 的序列 $\{a_k\}$ 的生成函数为 $\sum_{k=0}^{\infty} (k + 1)x^k$.
- 基于 $a_k = 2^k$ 的序列 $\{a_k\}$ 的生成函数为 $\sum_{k=0}^{\infty} 2^kx^k$.

8.4.3 计数问题与生成函数

EXAMPLE 10 求 $e_1 + e_2 + e_3 = 17$ 的解的个数, 其中 e_1, e_2, e_3 是非负整数, 满足 $2 \leq e_1 \leq 5, 3 \leq e_2 \leq 6, 4 \leq e_3 \leq 7$.

解: 解的个数是 x^{17} 在展开式中的系数

$$G(x) = (x^2 + x^3 + x^4 + x^5)(x^3 + x^4 + x^5 + x^6)(x^4 + x^5 + x^6 + x^7) \quad (12)$$

. 这是因为在乘积中得到等于 x^{17} 的项是通过在第一个和中取项 x^{e_1} , 在第二个和中取项 x^{e_2} , 在第三个和中取项 x^{e_3} , 其中需要满足 $e_1 + e_2 + e_3 = 17$ 的限制。

有三个解, 因为 x^{17} 的系数是 3.

8.4.3 计数问题与生成函数

EXAMPLE 13 用生成函数求 n 个元素的集合的 k 个组合的个数, 即 $C(n, k)$.

解: 集合中 n 个元素的每一个元素都对生成函数

$$f(x) = \sum_{k=0}^n a_k x^k \quad (13)$$

贡献了项 $(1+x)$. 因此 $f(x) = (1+x)^n$. 其中 $f(x)$ 是 $\{a_k\}$ 的生成函数, 其中 a_k 表示 n 个元素的集合的 k 个组合的个数. 根据二项式定理, 有

$$f(x) = \sum_{k=0}^n \binom{n}{k} x^k \quad (14)$$