

减少冗余与算法优化

A faint, light blue background illustration of a balance scale. The scale is tilted, with the right pan hanging lower than the left pan. The pans are empty. The scale's beam and vertical support are visible.

湖南省长沙市长郡中学
胡伟栋

减少冗余与算法优化

算法的目标：用最少的时间解决问题

要提高算法的效率，

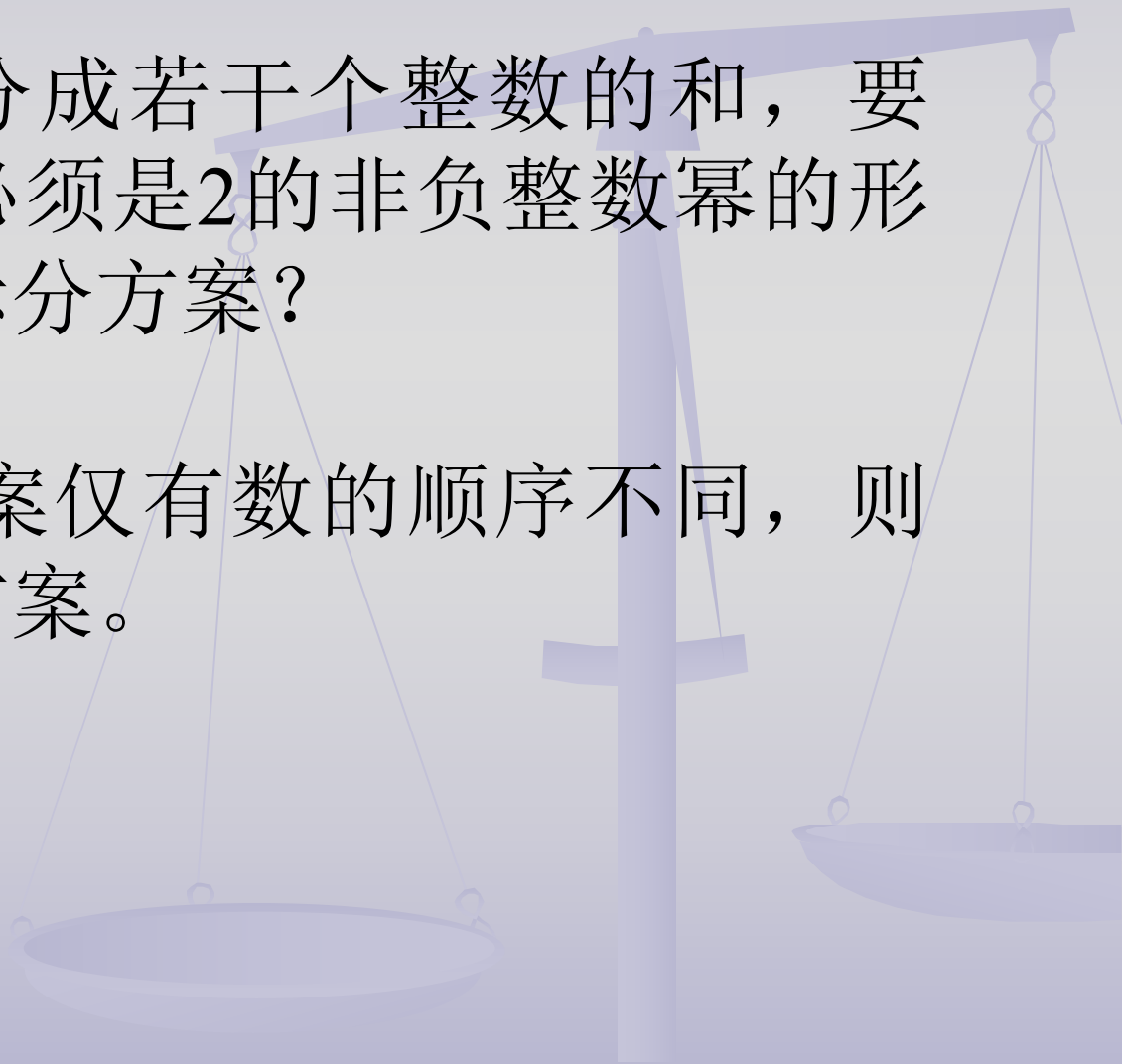
冗余：多余的或重复的操作
必须减少算法中的冗余

在搜索、递推、动态规划……中，都可能出现冗余

例1：整数拆分——问题描述

将整数 N 拆分成若干个整数的和，要求所拆分成的数必须是2的非负整数幂的形式。问有多少种拆分方案？

如果两个方案仅有数的顺序不同，则它们算作同一种方案。



例1：整数拆分——样例

当 $N=5$ 时，可以拆分成下面的形式：

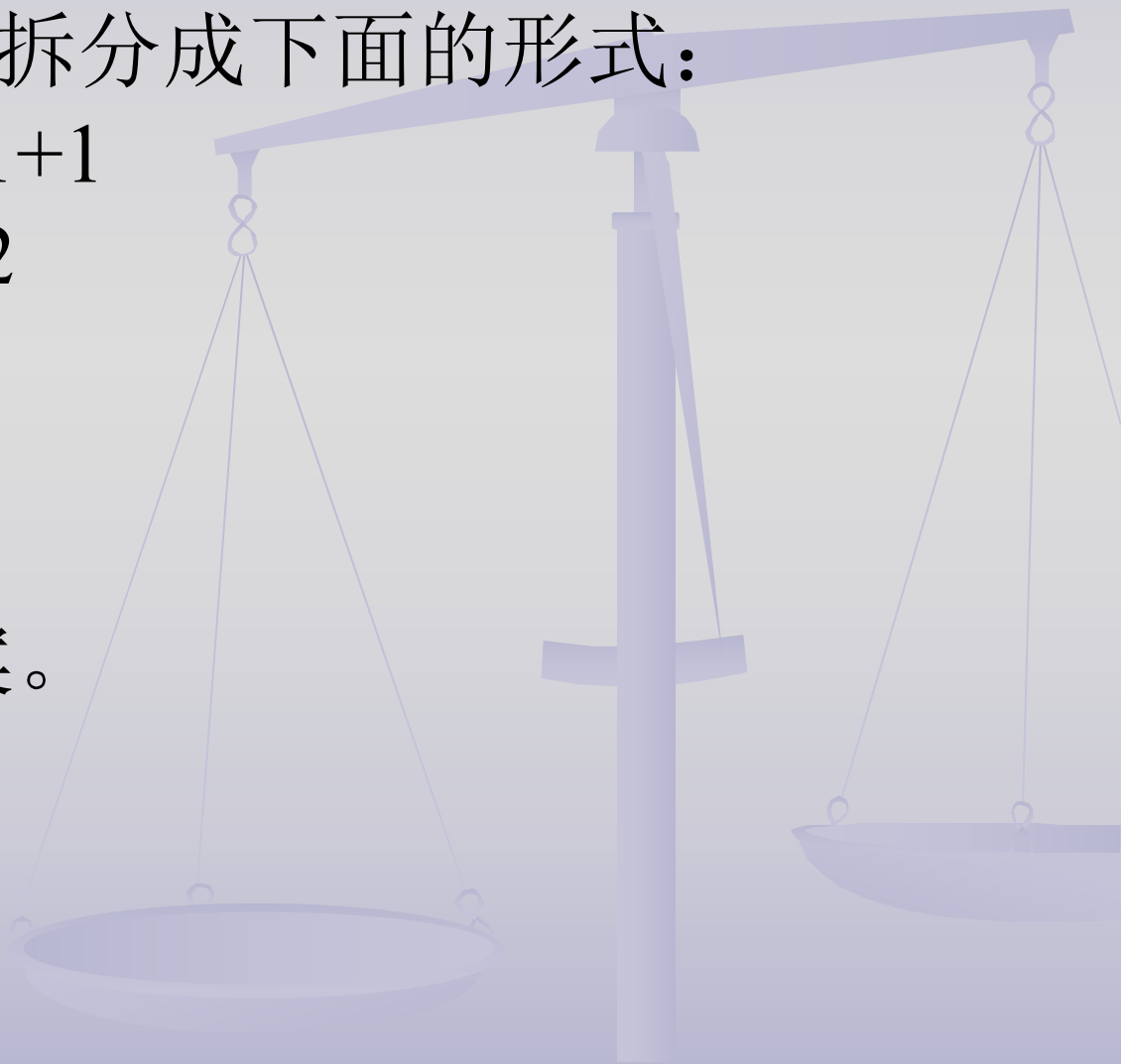
$$5=1+1+1+1+1$$

$$5=1+1+1+2$$

$$5=1+2+2$$

$$5=1+4$$

5有4种拆分方案。



例1：整数拆分——递推的建立

递推的表示：

用 $F[i, j]$ 表示 i 拆分成若干个数，其中最大的数不超过 2^j 的拆分方案数。

递推方程：

$$F[i, 0] = 1 \qquad F[0, j] = 1 \quad (\text{初始值})$$

$$F[i, j] = F[i - 2^j, j] + F[i, j - 1]$$

最大数是 2^j

最大数小于 2^j

目标： $F[N, \lfloor \log_2 N \rfloor]$

例1：整数拆分——递推复杂度

$$F[i, j] = F[i - 2^j, j] + F[i, j - 1]$$

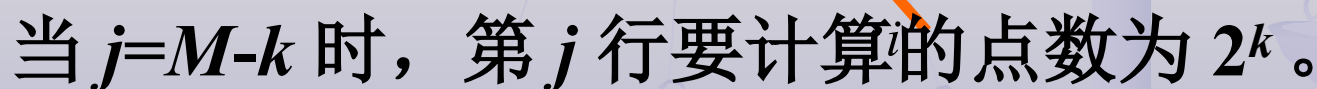
$$1 \leq i \leq N \quad 1 \leq j \leq \lfloor \log_2 N \rfloor$$

复杂度：

时间复杂度： $O(M \log_2 N)$

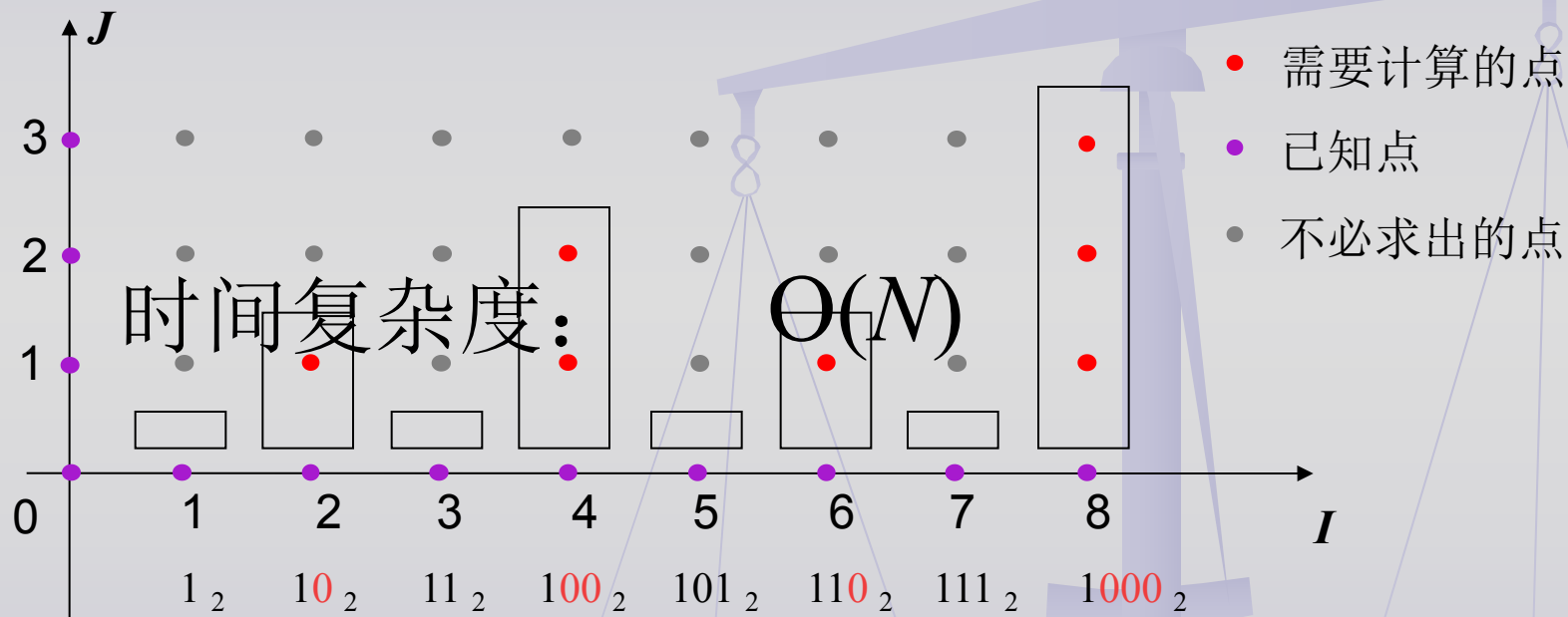
空间复杂度： $O(M \log_2 N)$

当 $N=2^M$ (M 是非负整数) 时



例1：整数拆分——减少冗余

当 $N=2^M$ (M 是非负整数) 时

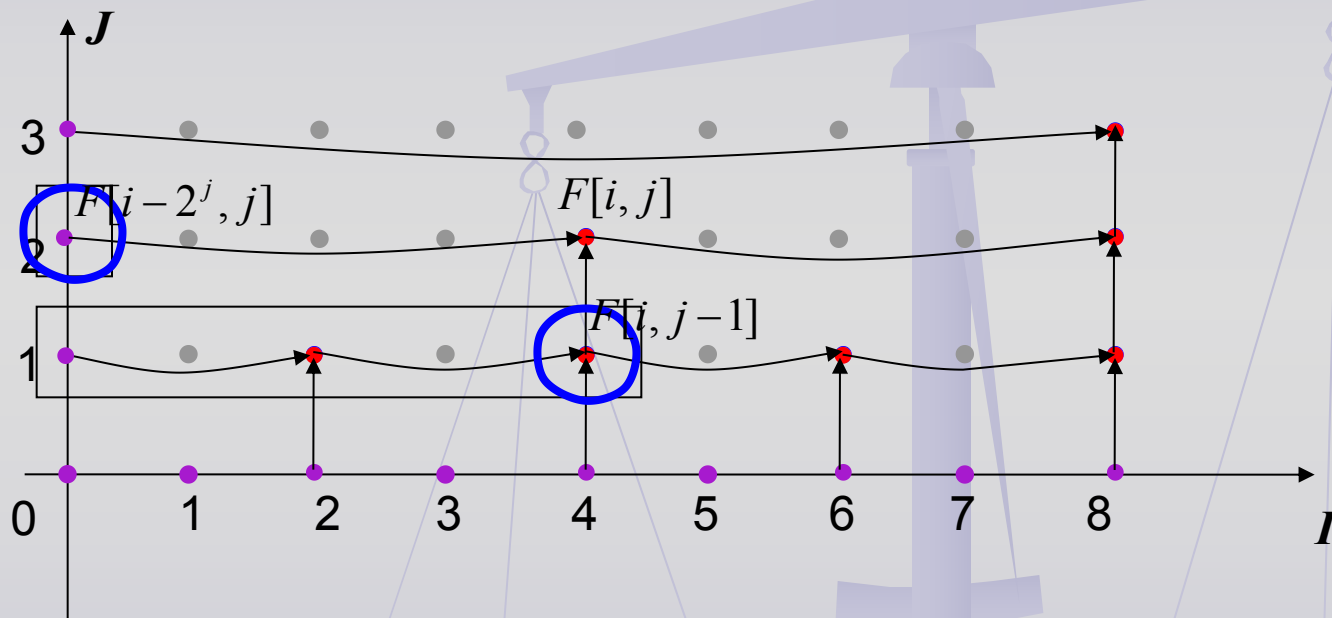


当 $i=x$ 时，第 i 列要计算的点数与 x 的二进制表示中最末的0的个数相等

例1：整数拆分——减少冗余

当 $N=2^M$ (M 是非负整数)时

- 未知点
- 处理中的点
- 已知点
- 不必求出的点

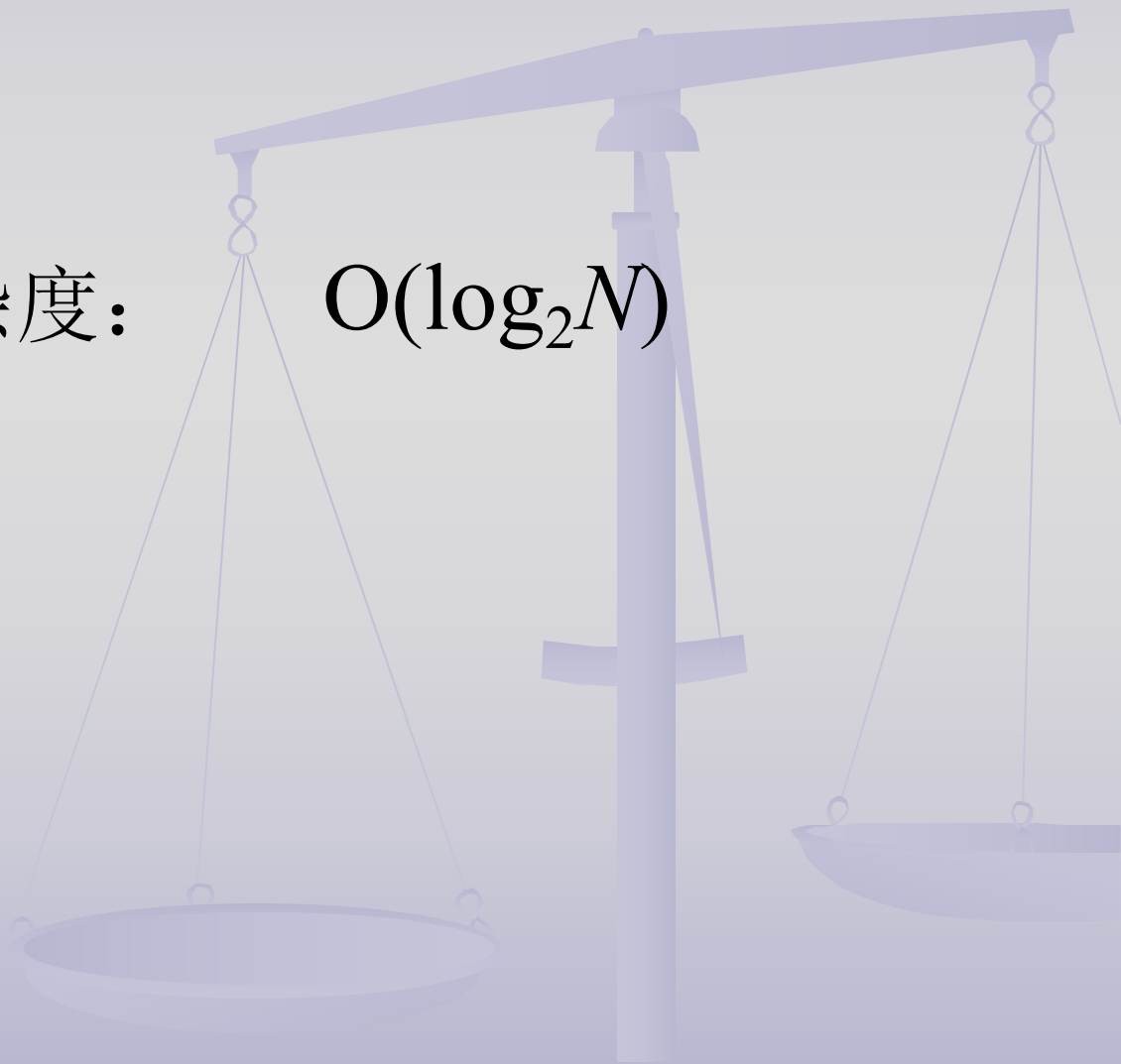


在所有 $F[x, j]$ (j 一定, x 为变量)中, 只要存储 x 最大的一个即可。

例1：整数拆分——减少冗余

空间复杂度：

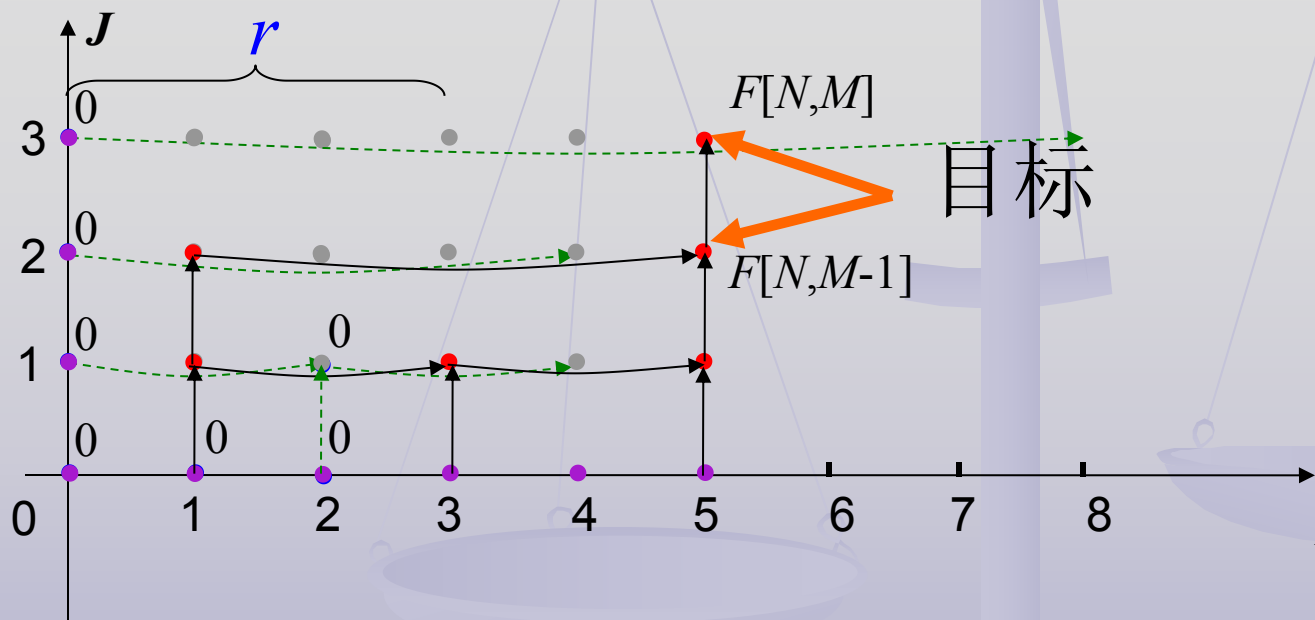
$O(\log_2 N)$



例1：整数拆分——减少冗余

当 $N \neq 2^M$ 时，可转化成 $N=2^M$ 的形式求解

设 $N=2^M-r$ ($2^{M-1} < N < 2^M$)



例1：整数拆分——小结

冗余 时空复杂度较高

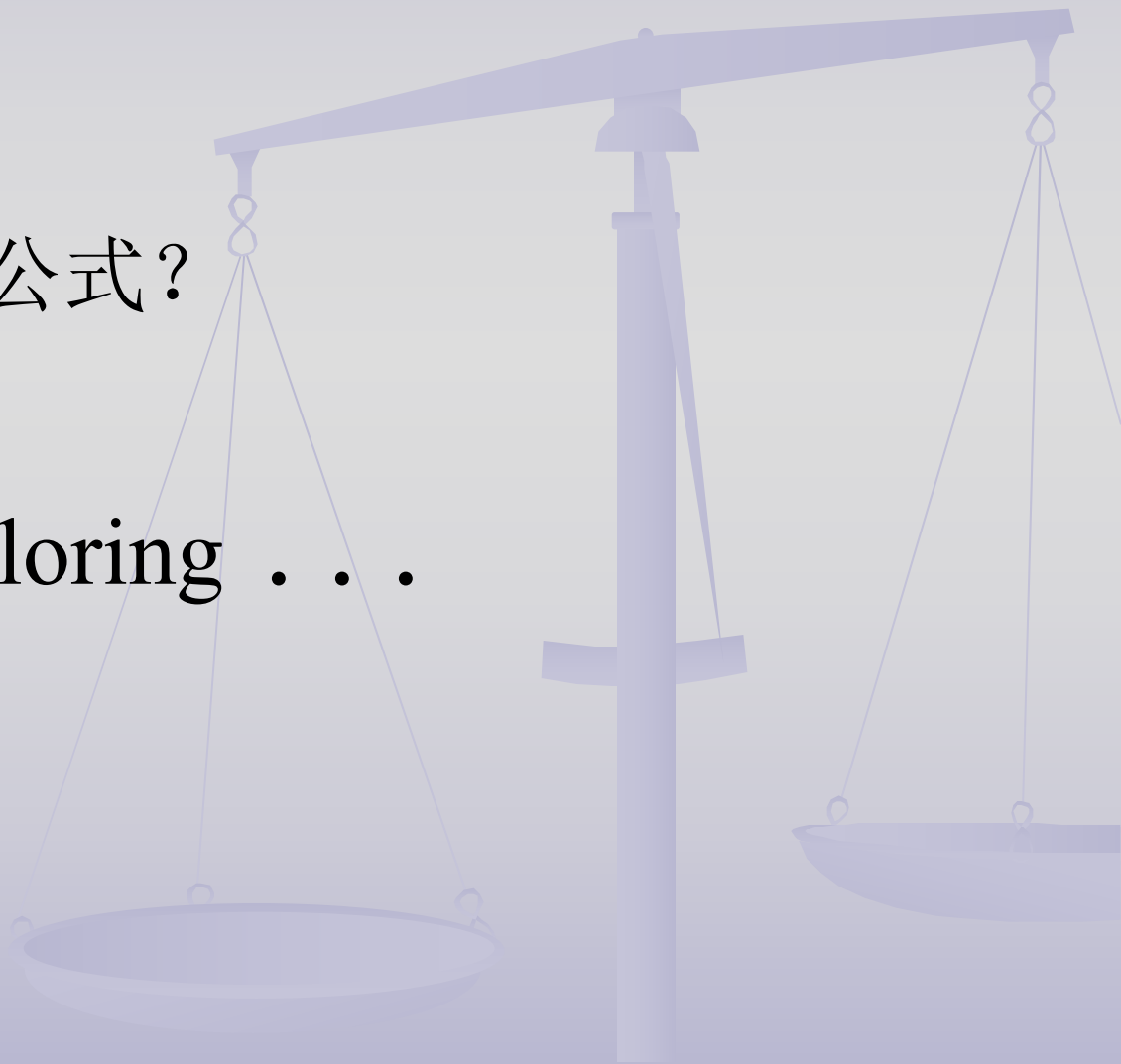
去除冗余后 时空复杂度相对很低

去除冗余 优化本题的关键

例1：整数拆分——最后的思考

更优秀的算法？ 公式？

Exploring . . .



例2：最大奖品价值——问题描述

有 $N+2$ 级楼梯，分别用0至 $N+1$ 编号，第1至 N 级楼梯上每级都放有一个奖品，每个奖品都有一个正的价值。如果某人从第0级开始，向上走 M 步正好到达第 $N+1$ 级楼梯，他将得到所走过的楼梯上的所有奖品，否则他将一无所获。问能得到的奖品价值的和最大是多少？

当然，一步不可能走太多级楼梯，假设每步最多上 K 级，即最多从第 i 级走到第 $i+K$ 级。

例2：最大奖品价值——数学模型

有一列数 $a_0, a_1, a_2, \dots, a_N, a_{N+1}$

其中 $a_0=0$ $a_1, a_2, a_3, \dots, a_N > 0$ $a_{N+1}=0$

从中选 $M+1$ 个数 $a_{i_0}, a_{i_1}, a_{i_2}, \dots, a_{i_M}$, 使

$$1) 0=i_0 < i_1 < i_2 < \dots < i_M=N+1;$$

$$2) i_1-i_0, i_2-i_1, i_3-i_2, \dots, i_M-i_{M-1} \leq K$$

$$3) a_{i_0} + a_{i_1} + a_{i_2} + \dots + a_{i_M} \text{ 最大}$$

例2：最大奖品价值——动态规划

状态表示：用 $F[i, j]$ 表示走 i 步到达第 j 级楼梯能得到的奖品的价值和的最大值

$$F[i, j] = \max_{j-k \leq x < j} \{F[i-1, x]\} + a_j$$

时间复杂度： $O(NMK)$

例2：最大奖品价值——规划中的冗余

从 $F[i-1]$ 到 $F[i]$ 的转移

$f_1[j]$ 表示 $F[i-1, j]$

$f_2[j]$ 表示 $F[i, j]$

$$\begin{array}{ccccccc} & & & \max & f_2[j] & & \\ & & & \underbrace{\hspace{10em}} & & & \\ f_1[j-k-1] & \underline{f_1[j-k]} & f_1[j-k+1] & \dots & f_1[j-3] & f_1[j-2] & f_1[j-1] \\ & \underbrace{\hspace{10em}} & & & & & \\ & \max & & & & & \\ & f_2[j-1] & & & & & \end{array}$$

例2：最大奖品价值——减少冗余

高

静态的考虑：

线段树 $O(NM \log_2 K)$

动态的考虑：每次都是找 f_1 堆连续的一段 $O(NM \log_2 K)$

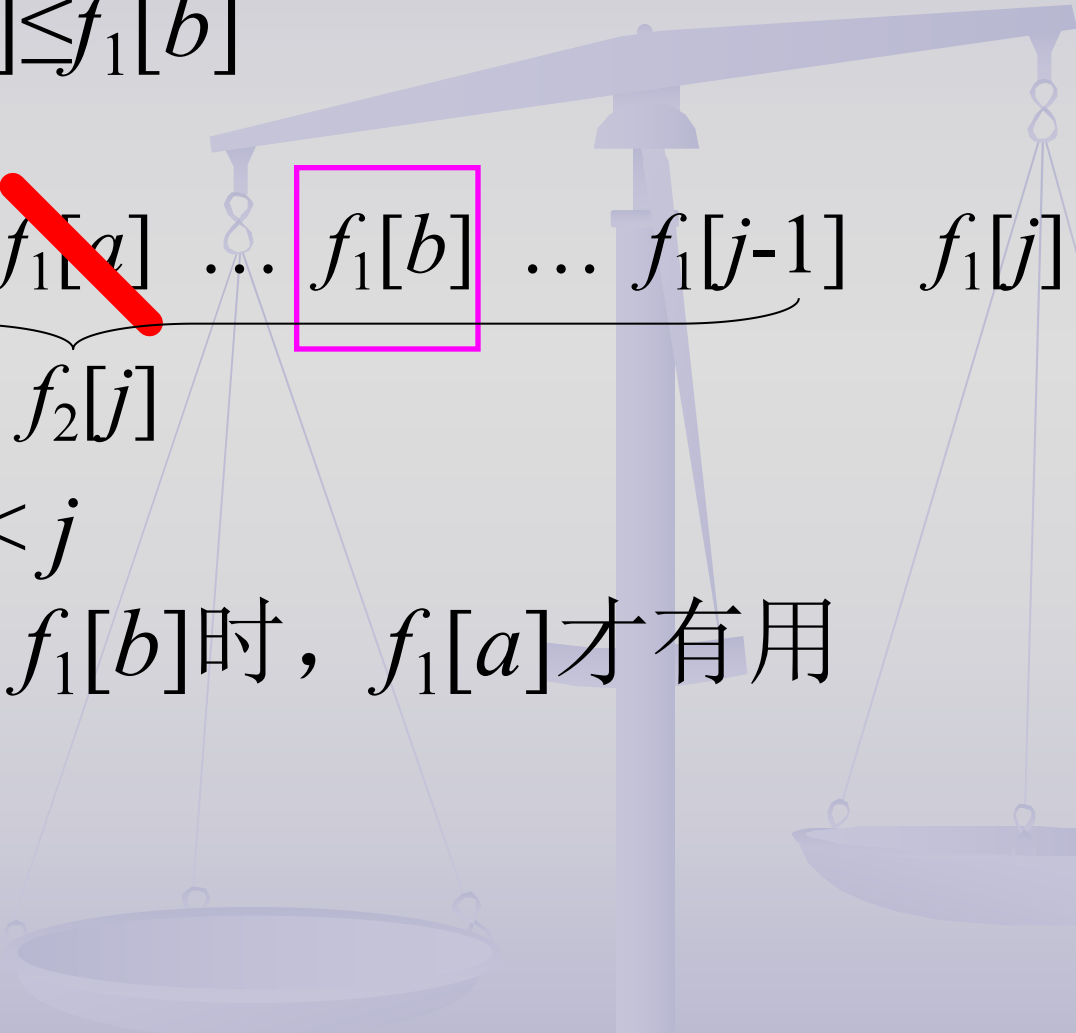
每次要求的 f_1 的一段都是变化的

编程复杂度 每次会加入一个新元素

每次会删除一个元素

例2：最大奖品价值——减少冗余

$$a < b < j \quad f_1[a] \leq f_1[b]$$

$$\underbrace{f_1[j-k] \ f_1[j-k+1] \ \dots \ f_1[a] \ \dots \ f_1[b] \ \dots \ f_1[j-1] \ f_1[j]}_{f_2[j]}$$


对于任意 $a < b < j$

只有 $f_1[a] > f_1[b]$ 时, $f_1[a]$ 才有用

例2：最大奖品价值——减少冗余

$$j-k \leq a_1 < a_2 < a_3 < \dots < a_r \leq j$$

$$\frac{f_1[a_1]}{\max} > f_1[a_2] > f_1[a_3] > \dots > f_1[a_r]$$

数据结构： 线性表*

删除第一堆栈素



新增元素到最后位置

维护这个数据结构使它保持递减的性质

x



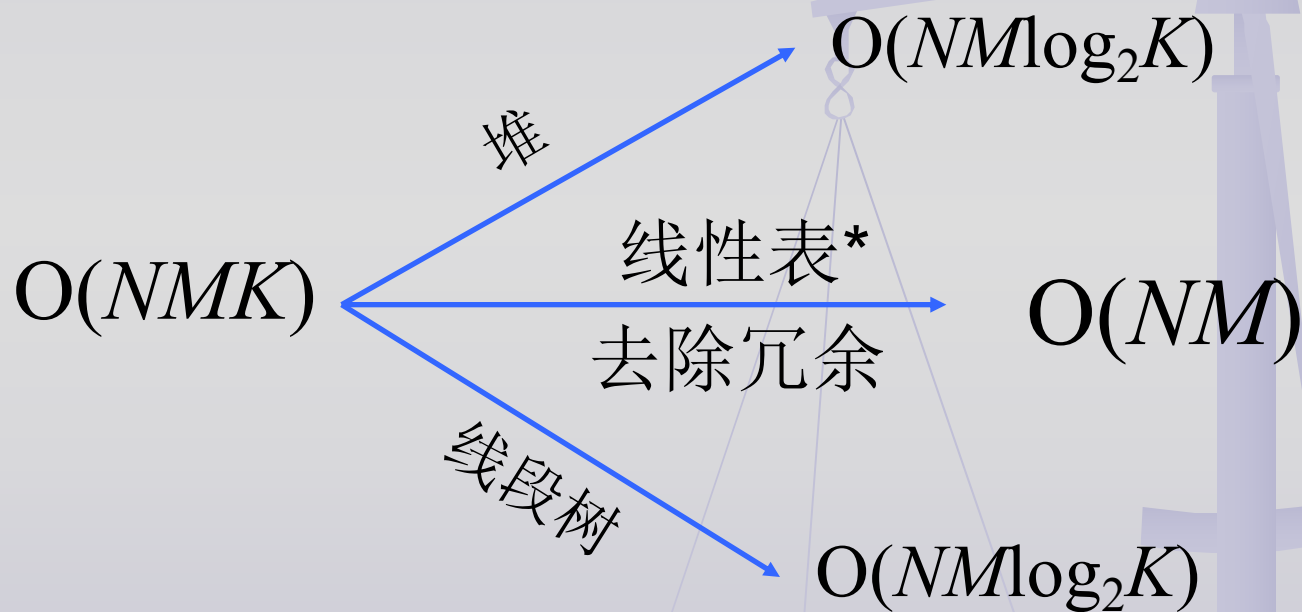
例2：最大奖品价值——时间复杂度

时间复杂度

$O(NM)$



例2：最大奖品价值——小结

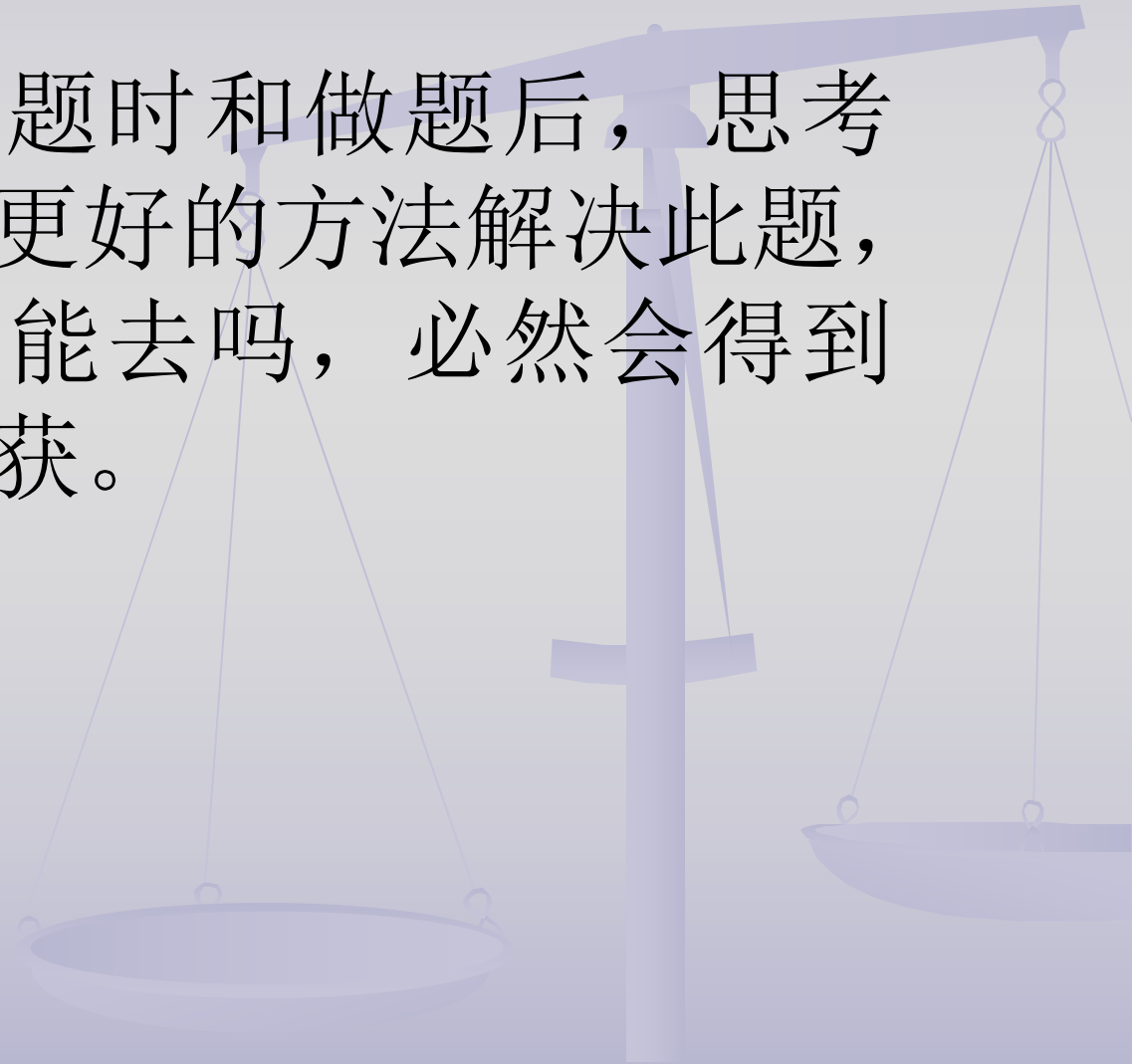


总结

- 在算法设计和编程过程中，冗余的出现是难以避免的
- 冗余是高效率的天敌，减少冗余，必然会使算法和程序效率提高很多
- 去除冗余没有可套用的定理公式可用，只有认真分析、善于探索，并在做题中积累经验，才能得到去除冗余的好方法

总结

如果在做题时和做题后，思考一下，能否有更好的方法解决此题，此题还有冗余能去吗，必然会得到意想不到的收获。



谢谢

