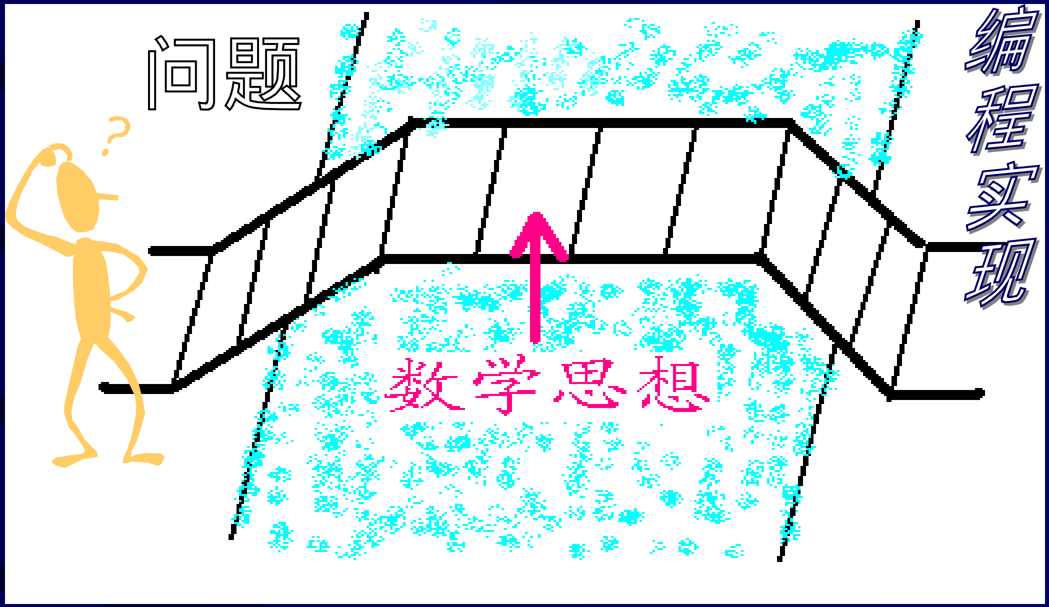
数学思想助你一臂之力

←— — — —

2003 1

数学和计算机原本就是密不可分的学科。有许多计算机编程问题如果不利用数学思想则很难甚至无法达到预期的效果。如果把问题和编程实现看成是河的两岸，那么数学思想就是连接河两岸的一座桥梁，有了这座桥，从河的一岸到另一岸便不再是件难事了。



有些问题，利用这座桥可以更方便地往返于河两岸，而还有一些问题，如果不利用这座桥，可能根本无法到达河对岸。

也就是说，有些问题利用数学思想可以走捷径（例如 NOI2002 的“荒岛野人”），而还有一些问题，如果不利用数学思想，就根本无法解决（例如 NOI2002 的“机器人 M 号”）。今天，我们将从四个方面探讨利用数学思想提高算法效率，简化问题的例子：

利用数学思想直接找出解的一般规律

利用数学模型化繁为简

通过数学分析化未知为已知

利用数学结论优化算法

一．利用数学思想直接找出解的一般规律

有些问题，如果直接用动态规划或是图论的方法来解决效率可能会并不理想；这时，我们首先应该想到的是优化，而如果优化无法达到预期的效果，那我们只有重新寻找算法了。于是，我们就试图找出问题的一般规律、或是该问题所用到的一个小问题的一般规律，这样，时间效率将会大大提高。

我们首先来看一个直接找出原问题的一般规律的例子——

例题一 最优分解方案

把正整数 N 分解成若干个互不相等的自然数的和，且使这些自然数的乘积最大。

[输入]

只有一行，包括数 N （3<=N<=1000）。

[输出]

第一行输出最优分解方案，相邻两数之间用单个空格隔开；第二行输出最大的乘积。

算法分析

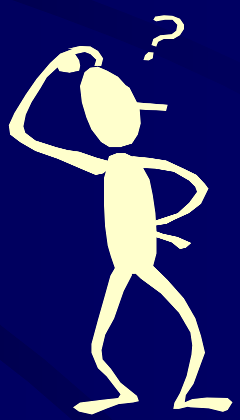
初看本题，发觉很显然可以用动态规划解决，但我们并不满足——直觉告诉我们，应该可以找到最优分解方案的一般规律，而一旦找到，时间效率将大大提高。

我们的直觉告诉我们，将 N 分解成的 m 个数应尽量接近，而且 m 应该尽量得大。

更一般地说，就是把 N 写成连续自然数 2 ， 3 ， …， k 之和（当然，由于自然数 1 不影响乘积，自然不将其加入），然后，将剩下的数依次平均分配到 k ， k-1 ， …， s 上，让这些数都加 1 。

例如，当 N=55 时由于 55=2+3+…+10 ＋ 1 ，将多余的 1 分配到 10 上，就得到

$$55 ＝ 2 ＋ 3 ＋…＋ 8 ＋ 9 ＋ 11$$



对于一些较小的数，我们发觉这个猜想是完全正确的。这促使我们跃跃欲试：证明这个猜想的正确性。

我们先来明确一下拆分方案的几种情况：

N=2＋3＋…＋s＋（s＋2）＋…＋k （如 N=55)

N=3＋4＋…＋k （如 12 ＝ 2＋3＋4＋3 ＝ 3＋4＋5）

N=3＋4＋…＋k＋（k＋2） （如 13 ＝ 2＋3＋4＋4 ＝ 3＋4＋6）

由此，我们可以把证明过程分为四步：

引理一：不存在$i(1 \le i < m)$，使得$a_{i+1} - a_i > 2$。

引理二：不存在$(i, j)(1 \le i < j < m)$，使得$a_{i+1} - a_i = 2$且$a_{j+1} - a_j = 2$。

引理三：$a_1 < 4$。

引理四：若$a_1 = 3$，则不存在$i(1 \le i < m-1)$，使得$a_{i+1} - a_i = 2$。

实际上，对每一步的证明过程并不难。总的来说，是利用反证法和调整的思想——先假设命题不正确，然后构造出另一列和为 N 的自然数，但乘积更大，从而导出矛盾。下面简单说一下每一步的证明：

引理一的证明：假设存在$i(1\le i<m)$，满足$a_{i+1}-a_i>2$，则将a_i调整为a_i+1，将a_{i+1}调整为$a_{i+1}-1$后，乘积更大，矛盾。

引理二的证明：假设存在$i, j(1\le i<j<m)$，满足$a_{i+1}-a_i=2$且$a_{j+1}-a_j=2$，则将a_i调整为a_i+1，将a_{j+1}调整为$a_{j+1}-1$后，乘积更大，矛盾。

引理三的证明：

若$a_1>4$，则将a_1拆分为2和a_1-2后，乘积更大，矛盾；

若$a_1=4$，则由引理一知$a_2\le 6$，而若$a_2=5$，则可将5拆成2和3，乘积更大，矛盾！故$a_2=6$。同理可知，$a_3=8$，而这已与引理二矛盾！

引理四的证明：

若存在i $(1\le i<m-1)$，满足$a_{i+1}-a_i=2$，则由引理二，知$a_{i+2}=a_{i+1}+1$，则若令：$a_{m+1}'=2$，$a_{i+2}'=a_{i+2}-2$，$a_j'=a_j$ $(1\le j\le m\ ,\ j\ne i+2)$，则$\prod_{k=1}^{m+1}a_k'>\prod_{k=1}^{m}a_k$矛盾。故命题得证。

至此，我们的问题应该已经得到了圆满的解答。

让我们再回顾一下解题过程，对于较原始的动态规划算法，我们觉得里面显然有许多不必要的计算，从而提出：能否直接导出一般规律？通过大胆的猜想和严密的证明，这一点得到了肯定，而算法的时间复杂度也从动态规划的 $O(N^2)$ ，下降到了现在的 $O(N)$。而这一切都应该归功于数学思想的大胆和合理的应用。

二．利用数学模型化繁为简。

能够对原题导出数学结论无疑是最直接地运用了数学思想，而在很多情况下，往往没有那么简单。在有些实际应用中，我们需要将原来的实际问题抽象成数学模型，然后加以解决。而在某些程序设计竞赛的试题中，建立数学模型也需要一定的技巧，需要运用一定的数学思想。

下面，我们来考虑一个利用数学思想“建模”的例子——

例题二 三角形灯塔

有一个N行（0<N<=50）的三角形灯塔，它的第1行有1个灯，第2行有2个灯，…，第N行有N个灯。我们用（I，j）表示从上至下第i行，从左至右第j个灯。

每个灯有明、暗两种状态，第i行（1<=I<N）的任一个灯（I，j）的状态由下一行的两个灯（I+1，j）和（I＋1，j＋1）的状态决定（1<=j<=I）。具体的规则为：当且仅当两个灯（I＋1，j）和（I＋1，j＋1）的状态不同时，灯（I，j）为亮。

请你编一个程序，从已知的P（P>=0）个灯的状态出发，推出最底一行N个灯的所有可能的状态总数。

[输入]

第一行行首为三角形灯塔的行数 N，从第 2 行开始每行为一个已知状态的编号为（I，j）的灯的信息（1<=I<=N，1<=j<=I），即三个由空格隔开的整数：I，j，k，其中 k 为该灯的状态，由 0，1 表示（0 表示暗，1 表示亮）。输入数据以满足 I=j=k=0 的一行结束。

[输出]

若问题无解，则输出“ NO ANSWER！”；否则输出可能的状态总数。

乍一看，可能会觉得此题只有用枚举搜索的办法解决，但其效率却并不理想。因此，我们不得不另觅他路。

如果我们仔细研究一下灯亮暗所遵循的规律，会发现：如果用 light[I， j] 表示灯（I， j）的状态（为 1 表示灯亮，为 0 表示灯暗），则有：

light[I， j]=light[I+1， j] xor light[I+1， j+1]

由于有了这一规律，我们便可根据最底行灯的状态，利用数学运算，推出所有灯的状态。

算法分析

但 xor 的运算仍然使我们觉得不好处理，这便使我们想到了运用无进位的二进制加法——

运用无进位的二进制加法，简单地说就是：$1+1=0$，$1+0=1$，$0+1=1$，$0+0=0$，其结果与“xor”运算是等价的，但由于其本质是加法运算，因此在本题中容易处理。

这样，我们便可利用最底部的灯的状态用加法表达出所有灯的状态（当然，这里的加法指的都是无进位的二进制加法），于是，根据已知的一些灯的状态，可以列出若干方程，组成方程组。这个方程组有 n 个未知数，即 light[n，I]（$I=1, 2, \ldots, n$），于是问题的解就转化为求方程组的解数。而这是不难求得的：

步骤一：用高斯消元法把方程个数减为一个，若在此过程中出现了矛盾，则输出“0”；

步骤二：若最后剩下的那个方程为恒等式，则输出 2^n；否则，若含有 p（$p>0$）个未知数，则由于随意确定其中的 p-1 个未知数的值，都将唯一确定另一个的值，故解数为 2^{p-1}。

这样，我们就找到了一个较为高效的算法。

在本题中，我们先是由灯的亮暗所遵循的规律发觉了一个较为一般的递推关系，而下面一步才是非常关键的——把递推关系中的逻辑运算转换成数学运算，这一步，为我们构造方程组这一数学模型提供了极大的方便，使问题迎刃而解。

数学思想在本题的应用，使我们方便地构造出了数论模型，使问题圆满地解决。

三．通过数学分析化未知为已知。

有些构造性地问题本身就是由数学问题衍生而来，但正因为问题的“构造性”，使这类问题的解法让人捉摸不透，很难想到。在这种情况下，我们可以试着先从数学的角度分析问题，若能得出对问题直接有益的结论则是最好，如果不能，我们也可以从分析问题的过程中启发思维，从而巧妙地构造算法。

下面来看一个具体的例子体会一下数学分析对解题的帮助——

例题三 配锁问题

某机要部门安装了电子锁。M 个工作人员每人发一张磁卡，卡上有开锁的密码特征。为了确保安全，规定至少要有 N 个人同时使用各自的磁卡才能将锁打开，并且任意 N 个人在一起都能将锁打开。现在需要你计算一下，电子锁上至少要有多少种特征，每个人的磁卡上至少有几个特征。如果特征的编号用从 1 开始的自然数表示，将每个人的磁卡的特征编号打印出来。要求输出的电子锁的总特征数最少。

为了使问题简单，规定：

$3<=M<=7$， $1<=N<=4$， $N<=M$

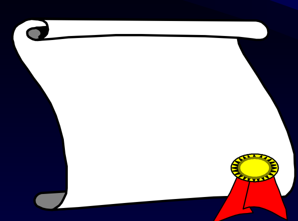
[输入]

只有一行，包括两个由空格隔开的正整数 M， N 。

[输出]

输出包括 M 行，第 i 行有若干个递增的正整数，表示第 i 个工作人员所持磁卡上的全部特征的编号。

算法分析



初看本题，也许会觉得毫无思路，但我们肯定能意识到：“至少要有 N 个人同时使用各自的磁卡才能将锁打开，并且任意 N 个人在一起都能将锁打开”，这个条件是本题的突破口。

让我们先利用这个条件进行一番分析：

由于“至少要有 N 个人同时使用各自磁卡才能将锁打开”，意即任意 N-1 个人在一起，都无法将锁打开，从而必然缺少一种开锁的密码特征 A ；并且在其余的 M- （N-1）个人中，任意一人加入到 N-1 个人中，他们就能将锁打开，故**这其余的 M-（ N-1 ）个人必同时拥有密码特征 A** 。而容易证明：每 N-1 个人在一起，他们缺少的一种密码特征 A 不能和其他一组 N-1 个人一起缺少的密码特征相同（否则，由于这两组至少有 N 个不同的人，且他们都缺少密码特征 A ，故这些至少 N 个人在一起无法将锁打开，矛盾）。从而电子锁上特征数 tot 应满足

$$\boxed{tot \geq C_m^{m-n+1}}$$

另外，对于每一个工作人员 T 来说，在其余 M-1 个人中，任选 N-1 个人在一起，都会因为缺少某种特征而无法开锁，而这缺少的特征必须是 T 所具备的。故每个工作人员的磁卡上的特征数 per 应满足

$$\boxed{per \geq C_{m-1}^{n-1}}$$

在上面的证明过程中，我们最感兴趣的并不是得到的结论（因为本题要求打印方案，故 tot 和 per 的值完全可以在打印过程中累加），而是在证明的“过程”中用到的：“每 N-1 个人都缺少一种密码特征，而**每 M-（N-1）=M-N+1 个人都同时具备该密码特征**，并且这个密码特征对从 M 个人中选出不同的 M-N+1 个人的组合来说是不同的”。这句话为我们打开了思路，由此，一个有效算法便应运而生了——

初始时，特征数置为 1，在 M 个人中每选取 M-N+1 个人的组合，就为组合中每个工作人员配备当前特征，并将特征数 +1 。这样，枚举出了所有的组合后，便得出了所有工作人员磁卡上特征的方案了。

在本题中，我们通过数学分析，尽管也得出了结论，但由于本题要求输出所有方案，它的用处并不大，但分析过程中所用到的一个思路却直接导致了算法的形成。如果没有进行数学分析，或者没有考虑到这一思路，本题似乎就很难完成了。

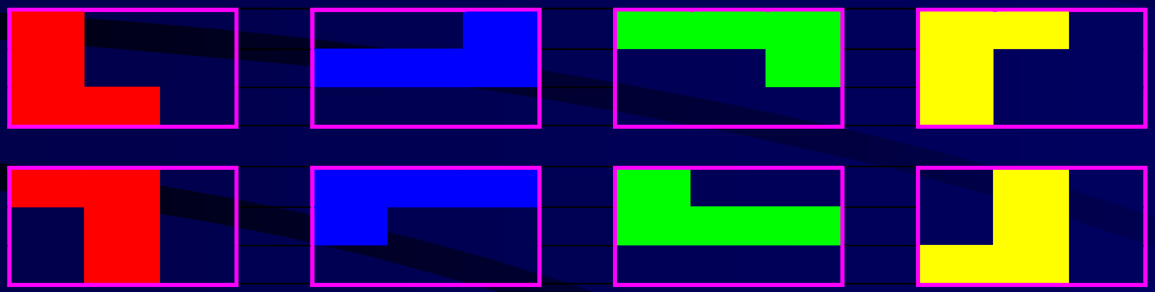
四．利用数学结论优化算法。

在上面的例题中，我们清晰地看到：数学思想的 火花孕育了解题的算法。综合上面的三个例题，数学方法、思想的巧妙使用，破解了一个个难题。最后，我们将通过一个并不难的搜索题，通过优化前和优化后算法时间效率的比较，来体会一下数学在优化算法方面的应用。

例题四 骨牌覆盖问题

对于任意一个 m*n 的矩阵，求 L 形骨牌覆盖后所剩方格数最少的一个方案。

下图为 L 形骨牌的八种形态:



[输入]

输入包括一行，有由空格隔开的正整数 m ， n ，表示矩阵的大小。

[输出]

输出是一个 m*n 的矩阵，矩阵中元素的值表示该格所在的骨牌编号（若该格不被骨牌覆盖，则值为 0），骨牌编号为从 1 开始的连续正整数。

算法分析

对于搜索题，我们应该把注意力集中在优化算法上。对于本题，显然在回溯搜索时，不能让已经“浪费”了的方格数太多，这样再继续搜索下去也是做无用功，于是我们就需要一个“浪费”方格数的上界，也就是最优方案中，最多能覆盖的骨牌数目的下界；而事实上，我们完全可以求出它的确切值。

实际上，一般来说，对于$m\times n$的矩阵，能覆盖它的L形骨牌数目最多为：$number=\left[\dfrac{m\times n}{4}\right]$，而事实上，这一点仅对$m\times n\equiv 4(\mathrm{mod}\,8)$时，不正确，此时，应有$number=\left[\dfrac{m\times n}{4}\right]-1$。我们只需证明如下的引理：

引理：如果一个$m\times n$的矩阵能用L形骨牌完全覆盖，则$\dfrac{m\times n}{4}$必为偶数。

下面将简单证明这个引理，而对于最大值能够取到，也是可以用数学归纳法证明的，由于时间关系，这里从略。

引理的证明：

我们引入L形骨牌的“头部”和“尾部”的概念：即字母“L”底部的两个方格称为“尾部”，另外两个方格称为“头部”。“头部”和“尾部”都类似多米诺骨牌。不妨设n为偶数，则考虑$m\times n$矩阵的每一条水平线，这m条水平线共穿过“L”形骨牌的“头部”或“尾部”的数目总和记为c，一方面，对于每一条水平线，若它穿过奇数个“头部”或“尾部”，则在该水平线上部的其余奇数个方格内要实现多米诺骨牌的完全覆盖是不可能的。因此，每一条水平线必然穿过偶数个“头部”或“尾部”，故c为偶数。另一方面，每个“L”形骨牌恰好被一条水平线切割到其“头部”或“尾部”，从而“L”形骨牌的数目为偶数，即$\dfrac{m\times n}{4}$为偶数，得证。

由此，我们得到了优化后算法的大致流程（回溯法）：
自左而右、自上而下搜索每一个方格的覆盖情况，如果可以以该方格为左上角放置骨牌，则试着放置该骨牌，设置相应的方格被覆盖标志；然后设置该方格不被覆盖标志，搜索下一方格。搜索过程中，一旦已经产生的空格数超出空格数的上界，则回溯。

对比优化前和优化后算法的时效，我们发现，由于加入了槛值，使得优化后的算法减少了很多不必要的搜索，算法的效率自然提高了不少。

本题给了我们这样的启示：数学思想不仅仅能直接提供解题思路和方法，在更多的场合，它只是以一种辅助工具的形式出现的。这就要求我们具备将多种算法，多种思想结合使用的能力。

结束语

Conclusion

Thank you

从上面的几个例子中，我们可窥一斑——数学思想作为沟通问题与编程实现的一座桥梁，有着极其广泛的应用。它不仅可以直接为解题提供思路，如果与其他算法或思想相结合，也能够起到很好的辅助作用。最后，送给大家一句话，希望有所帮助：

在你觉得“山穷水复疑无路”时，不妨用数学的角度重新审视问题，也许就会“柳暗花明又一村”。