数学归纳法与解题之道

山西省实验中学　张昆玮

教练：唐文斌　胡伟栋

2008 年 12 月 16 日

引言

> 道生一，一生二，二生三，三生万物。
>
> ——老子：《道德经》

我们在学习算法的时候，总会有这样那样的疑惑：如此之多的浑然天成的算法，是怎样想到的？这些算法巧诚巧矣，可它们为什么是正确的呢？其实林林总总的算法背后，无不隐藏着真正的解题之道。解题之道博大精深，该从何谈起？上面的疑惑又该如何解决？两千五百多年前，著名的哲学家和思想家老子对道之本质参透得可谓淋漓尽致。引文中他给我们的答案是：参悟解题之道，从**数学归纳法**开始。

摘要

本文简述了数学归纳法的相关理论，并结合作者的解题实践，以一些经典问题与竞赛题目为例，从证明算法正确性、构造性算法、与算法优化的关系等几个方面介绍了数学归纳法在信息学竞赛中的应用，讨论了数学归纳法的实用性与适用范围，以及归纳式算法设计相关的其他一些延伸与拓展。

目录

关于数学归纳法

➢ 简短的回顾

数学归纳法原理的发现可以追溯到公元 1494 年意大利数学家 Francesco Maurolico 的著作 *Arithmeticorum libri duo*。作为皮亚诺公理系统的应用，它自诞生之日就受到广泛关注。同时，作为应对与正整数相关的问题的一种通用而简洁的处理方法，数学归纳法在许多算术、逻辑问题的解决中都占有核心地位。即使是一些难于解决的问题，应用数学归纳法大多可以化简问题描述，理清其本质，为接下来的解答铺平道路。

由于其固有的递归性，构造性和美学价值，数学归纳法是中学课本与学科竞赛中的热门话题。特别地，信息学竞赛中的算术、逻辑、拓扑、数论以及组合数学相关的问题层出不穷，我们有理由相信，数学归纳法，以及由数学归纳法衍生的结构归纳法等构造性证明方法在信息学的算法设计中，同样会有其用武之地。

➢ 基本的定理、概念与方法

✧ 第一数学归纳法

任意给定关于自然数的命题 $\mathrm{P}(n)$，若 $\begin{cases}\mathrm{P}(1)=\mathrm{True}\\ \mathrm{P}(n)\Rightarrow\mathrm{P}(n+1) \quad n\in N^*\end{cases}$，那么，命题对所有自然数均成立。

形象地说，如果把整个自然数集想象成一串多米诺骨牌，$\mathrm{P}(1)=\mathrm{True}$ 这个要求就推倒了其中的第一块，因此有时被形象地称之为“奠基”；我们通常称为归纳的是其中证明

$\mathrm{P}(n)\Rightarrow\mathrm{P}(n+1)$的部分，其作用是只要前一块骨牌被推倒就保证后一块一定会倒，这样整队骨牌就会依次倒下。

✧ 第二数学归纳法

任意给定关于自然数的命题$\mathrm{P}(n)$，若$\bigcap_{k=1}^{k<n}\mathrm{P}(k)\Rightarrow\mathrm{P}(n)\quad n\in N^*$，那么，命题对所有自然数均成立。

第二数学归纳法是第一数学归纳法的推广，一般而言，在线性结构上除了一些简单的问题可以直接使用第一数学归纳法之外，我们使用的大都是第二数学归纳法。它的优点在于在证明问题时可以使用任何之前证明过的结论而不仅仅是前一个。这样我们就可以在适当的时候把问题规模缩小一半或更多，而不仅仅是减一，从而得到一些灵活高效的算法。

✧ 结构归纳法

结构归纳法是应用在数理逻辑、计算机科学、图论和一些其他数学领域中的一种特殊化的数学归纳法，一般用来解决非线性问题。通俗地说，如果要证明的一组基于结构的命题之间有一个“拓扑序”（通常定义为问题的规模），那么我们往往可以从空集出发进行归纳证明或构造。例如，树形结构上我们每次去掉根将问题规约到几个较小的子问题，图论算法中每次处理一条边就能不断减小问题的规模……如果对于一个规模足够小的问题我们能够解决，我们就完美的解决了原问题。

不同于一般的数学归纳法，结构归纳法的正确性依赖于“最小数原理”的推广：假设存在反例，那一定会有最小的反例，但是运用上述论证减小其规模可得一个反例，它的规模比最小的反例更小，这样就产生了矛盾。因此假设不成立，结构归纳法是正确的。

从证明中就可以看出，结构归纳法中问题规模的定义非常重要。同时，问题规模在应用中也异常灵活。如果我们不想一次性“树敌太多”，那么在必要的时候需要加强归纳假设，把貌似相同的变量剥离开来。关于这一点，请看下面的例子。

【例 1】无根标号树的 Prüfer 编码

为了证明 Cayley 公式，Heinz Prüfer 于 1918 年提出了著名的 Prüfer 编码，其核心思想与这里谈到的结构归纳法是一致的。Cayley 公式的内容为 n 个顶点可能构成的不同无根标号树共有 n^{n-2} 个。Prüfer 编码断言：存在一个双射将 n 个顶点 m 种可行标号构成的无根树对应到长度为 $n-2$，取值范围为 $1\sim m$ 的数列。

如果我们按照问题规模中的 n 确定归纳假设，只需找出其中编号最小的叶子，以与它相邻的节点的编号为数列前缀，在删除此叶子所构成的新树中运用归纳假设就可以完成 Prüfer 编码的构造性证明。而 Cayley 公式只是 Prüfer 编码当 n 与 m 相等时的特例。注意这里如果我们不使用结构归纳法而是笼统的对 Cayley 公式的 n 归纳，问题难度就会随着 n 的过度出现而成倍增加。

➢ 是总结更是探索

许多同学从长期的解题实践中可以发现，不管是数学问题的解决还是数学模型的建立，都由两部分组成：一部分是猜想与探索，另一部分便是总结与证明。一般认为，数学归纳法虽然名为“归纳”，却在演绎推理方面应用更多，也就是说往往用来在已知“正确”的情况下加以证明，而不是对未知的世界进行探索。其实很多时候，我们在思考问题时也同样已经不自觉地运用了数学归纳法的思想。本文的写作目的也包括唤起大家的热情，在猜想与探索方面也重新认识数学归纳法的优势。

【例 2】社会名流问题

求一个邻接矩阵表示的有向图中只有入度没有出度的点，使得边的询问总次数最小化。直接确定答案搞不好会扫描全图，确定某个点不是答案却要容易得多。何不试试数学归纳法？一旦我们能确定某个点不是答案，我们就减小了问题的规模，之后的事情就是运用归纳假设。任意询问一条边，存在就删掉起点，不存在就删掉终点，如此反复即可。这道题是一个标准的交互式问题，解法也不那么需要证明，但是探索解题方法的过程中数学归纳法确实发挥了核心作用。

在证明算法正确性上的应用

虽然我们在日常解题中往往把 AC 作为一道题目的终结，但是学习中知其然，更要知其所以然，能够证明算法的正确性无疑等于让我们埋头于题目的时候吃下了一颗定心丸。毕竟，解题之道不是仅仅凭印象，凭感觉就可以办到的，尤其是对于贪心算法来说，证明往往是算法成败的关键。作为一种强有力的证明方法，数学归纳法在证明算法正确性时有广泛的应用，甚至可以说没有数学归纳法就没有算法证明。关于这一点，下面举隅说明：

➢ 贪心算法

【例 3】Robbers

有 N 个强盗抢劫银行成功获得了 M 个金币。抢劫前他们约定之后第 i 个强盗会得到 $\frac{X_i}{\sum X_i}$ 的钱。可惜世上没有那么多的公平，金币也不能拆成半个。设现在每个强盗得到 K_i 个金币，请设计方案使 $\sum\left|\frac{X_i}{\sum X_i}-\frac{K_i}{\sum K_i}\right|$ 最小。

一个显而易见的方案是先将 $M_0=\left[\frac{M}{\sum X_i}\right]$ 的金币分给大家，然后根据经济学边际效用递减的原理，每次拿出一块金币给 $\Delta_i=\frac{X_i}{\sum X_i}-\frac{K_i}{\sum K_i}$ 最大，也就是缺口最大的强盗。不过

“边际效用递减”只能用来说服自己，想要说服这些强盗这就是最优方案，还得着实动一番脑子。

归纳假设：　　对已分配的 m 个金币有 $\max_i \Delta_i - \min_i \Delta_i \le \frac{1}{M}$。

当 $m = M_0$ 时由于金币可以按比例均分故 $\frac{X_i}{\sum X_i} = \frac{K_i}{\sum K_i}, i \in [1, n]$ 归纳假设成立。假设 m 个金币时命题成立，现在讨论 $m+1$ 个金币的情形。如果给 $\min_i \Delta_i$ 的强盗的一枚金币没有让他成为 $\max_i \Delta_i$，那么 $\max_i \Delta_i - \min_i \Delta_i$ 的值变小了（为什么？）等于不用证了。如果这枚金币让他成为了 $\max_i \Delta_i$ 利用 $\max_i \Delta_i' = \min_i \Delta_i + \frac{1}{\sum K_i} = \min_i \Delta_i + \frac{1}{M} \le \min_i \Delta_i' + \frac{1}{M}$ 可推得最终归纳结论 $\max_i \Delta_i - \min_i \Delta_i \le \frac{1}{M}$。

接下来要证明的是归纳结论保证方案最优，由于和数学归纳法关系不大，证明方法也多种多样（作者想出的就有两种），这里只给出证明提示[1]，具体细节请同学们自己完善，这里因篇幅所限不再赘述。

➢ 其他算法

【例 4】Think Positive

[1]证明提示：首先使用调整法可以证明最优解必然满足归纳结论，其次证若允许 Δ_i 的不同排列，则最优解存在且唯一。

Eisiem 星球上一年有 n 天。会使人高兴的日子被赋值为 $+1$，别的苦日子就是 -1。每过一天，这一天的情绪就会加在居民们已有的情绪上 不过元旦那天辞旧迎新的时候 大家会忘记曾经的喜怒哀乐。国王希望大家天天高兴，也就是每一天人们的情绪都为正。可是元旦该怎么定呢？国王决定先让你算一算有几种定法。

这是一个实际问题，建立数学模型可以发现，其实题目是在求能使一个循环数列的前缀和均恒正的起始点数目。看到此题之后，作者首先想到的是一个很暴力的算法：让起始点逐个后移，用单调队列维护前缀和序列中的最小值，所有最小值为正的时刻都是答案。虽然使用特殊的数据结构确实做到了线性的时间复杂度，但 WA 了好多次，充分说明编程复杂度高，思维量也够大。下面我们用第二数学归纳法证明一个可以瞬间解决此题的结论。

归纳假设：　　欲求 $n<k$ 天的方案总数目，只需将所有日子的值求和，负值输出 0。

当 $k<2$ 时，手工验证 $n\in\{0,1\}$ 得归纳基础成立。下面假设 $k\geq 2$。如果该循环数列中存在相邻的“$+1\ \ -1$”，那么将其去掉。此时数列的和不变。显然这两个数都不能作为数列的起始点：那去掉它们之后，其他起始点的前缀和呢？容易发现，这些前缀和不以去掉的这对“$+1\ \ -1$”结束的不变，以这对数结束的不小于前面紧接着的，没有讨论价值。这样，问题规模就降到为 $n-2$，此时使用归纳假设即可。

如果该循环数列中不存在相邻的“$+1\ \ -1$”呢？那有两种可能。要么这个数列中根本不存在 $+1$，应了“负值输出 0”这句话；要么每个 $+1$ 的后面都不是 -1 而是 $+1$，那么数列中全都是 $+1$，每个点都可以做起始点。这样我们就得到了一个非常简单的线性时间算法。诸如此类的题目还有很多，虽然不一定需要我们大动干戈，但是往往需要更多猜想的智慧和证明的勇气。

在构造性算法中的应用

在各种程序设计方法中，构造性算法往往是最令我们头疼的：与其他算法不同，它往往缺乏固定套路，需要更多的创造性。更有甚者，明知要构造却又绞尽脑汁无从下手。另外，这类算法往往依赖于具体的数据结构，因此考察的综合性较大。

如今，越来越多的同学老师把构造性算法当作学习重点，命题热点来关注。既然数学归纳法有着鲜明的构造性色彩，把数学归纳法的思维方式应用在构造性算法中，就一定可以帮助我们打开突破口，找到解题之道。换句话说，数学归纳法可以成为想不出解法时的“退路”，也可以成为一种通用的构造性解题策略。

➢ 数据结构的恢复性构造

数据结构中线性结构与树形结构都有明显的递归性，如果要对这类数据结构进行恢复或重构，首选解题方法当然是从部分到整体，一次增加一个元素，步步为营，与其对应的思维方法就是不断补充归纳假设，完善状态设计。既然这类问题常常出现，又恰恰是数学归纳法与归纳式算法的强项，我们就从数据结构的恢复性构造说起。

✧ 线性结构

【例 5】Set Cover

子集覆盖问题定义为选出尽量少的子集，使已知集合中的每个元素至少属于其中的一个。线段覆盖问题定义为选出尽量少的整点，使给定的每条线段上都至少有其中的一个。以整点为子集，所有包含这个整点的线段为子集中的元素，可以把一个线段覆盖问题归约到子集覆盖问题。如果给定一个由线段覆盖问题归约成的子集覆盖问题，该怎么解决呢？

对于线段覆盖问题而言，互相包含的线段在最终问题里只需考虑被包含的一条，我们预处理后就可以假定线段之间没有包含关系。之后我们只需按数轴顺序扫描线段，若某条线段当前没有被覆盖，就添加能覆盖它的最靠右的点。这是一个我们熟知的贪心算法。

鉴于子集覆盖问题本身是一个 $\mathcal{NP}$ 完全问题，其解决的复杂度要远远超过线段覆盖问题。因此本题重点无疑是要恢复一个线性结构。根据贪心算法的提示，我们不需要完全恢复原问题，只需要找出线段的位置关系和每个线段的右端点。下面我们定义一条线段的位置为中点的坐标，这样可以将线段之间的两两距离看作它们中点之间的距离。

> 归纳假设：　已知当前连通分量中已计算完成的所有线段的位置，以及其中一条与要添加的线段 X 相交的线段 A。

关于这个归纳假设，有几点需要说明：

1.两组两两不相交的线段可以相隔任意远且在最终结论中无差别，因此在上面的归纳假设中，我们将问题按照相交关系划分为连通分量来处理。

2.每个分量中第一条线段的位置对我们来说无关紧要，我们把它简单的赋值为 0 就可以作为归纳基础。

3.归纳假设中后一个要求是容易满足的，如果无论如何选择 X 仍然不能满足此要求，等价于说明这个连通分量已经结束了。

有了这个归纳假设，之后只需要确定 X 的位置。不难看出既然 A 与 X 相交，他们的距离其实就是对应集合元素差异个数 $|A \operatorname{xor} X|$ 的一半，容易求出。可是 X 在 A 的哪一侧呢？只有一条线段时方向无关紧要——选择不同的方向会使最终的结果互为轴对称，不会影响结论。其他情形中为了判断方向，我们需要调整归纳假设。

> 归纳假设：　如果线段总数不止一条，那除了以上假设，还需要取一条已求出的线段 B，知道其关于 A 的方向。

枚举几种情况可以发现，X 与 B 在 A 的同侧的充分必要条件是他们在 A 之内的部分存在包含关系即 $X \cap A \subseteq B \cap A$ 或 $X \cap A \supseteq B \cap A$。这样我们成功确定了 X 的方向，完成了算法求位置的过程。

求线段的右端点怎么办呢？设需要求出右端点的线段为 A。与这个问题相关的线段除了 A 本身，还有所有与 A 相交、在 A 右侧的线段。 A 的右端点无非就是这些线段与 A 的公共点。我们只需要求出所有对应集合的交集，从中任取一个元素充当 A 的右端点即可。

以上解题过程用庖丁解牛的方法步步推进，将问题分为多个部分，逐一化解。其中数学归纳法灵活的归纳假设为主要问题的求解提供了不少便利，算法实现中我们不断克服困难的过程就是归纳假设不断完善的过程。

✧ 树状结构

【例 6】Roman Roads

两千多年前，罗马帝国控制着欧洲包括地中海在内的广袤的交通网络，做到了条条大路通罗马。为了节省资源，交通网络中没有环。给出所有路之间两两是否直接相连（多条路可以在一个城市交汇，这时他们两两相连）的史料记载，试判断史料的真伪并给出古罗马的交通方案。

题目中给定了边之间的邻接关系，要求的是一棵树。树中讨论边的邻接关系，自然也离不开点的层次关系。既然顶点与边的数目大致相同，我们可以把一个顶点 x 对应到它连往父亲的边 X。下面的算法中，我们就这样使用对应的大写字母来表示一个顶点连往其父亲的边。有了上一个问题积累的经验，我们从树的递归结构入手，也就是说可以考虑按照树本身的层次结构来展开归纳。从上题的经验可以看出，直接将归纳假设定为某棵子树是不恰当的，我们的归纳假设中应当包含更多有价值的信息。

归纳假设：　已知某子树根 x，指向 p 的边 X，和边 P 三个条件时，可求出树形方案。

与边 X 相邻的边可以分为两类：一类连在 p 上，另一类连在 x 上。连在 p 上的边在处理以 p 为根的子树时已经处理完成，连在 x 上的边需要我们建立一棵子树，然后对子树运用归纳假设。 P 的引入为我们分开两类边提供了可能：连在 p 上的边与 P 当然是相邻的，

连在 x 上的边与 P 却不相邻。这样，算法在实现时可以使用集合论中简洁高效的差集运算来完成，方案的真伪也只需使用充要条件“所有的枝条都有且只有一个根”来判断。

这个归纳假设中默认了我们的算法选择了自顶而下的构造。虽然自底向上恢复更符合逻辑，而且确实有对应的算法（大家可以尝试使用每次删去一个叶子的方式解决此题），但是题目中给出的条件没有给出叶子的信息，从叶子出发需要反复寻找叶子，判断真伪也必须等到树恢复之后单独进行，平添了许多不必要的麻烦。树形结构恢复的题目还有很多，其中不乏一些需要特殊数据结构的题目，都很有代表性，只是这里限于篇幅无法一一列举。作者相信大家只要掌握了数学归纳法的精神，遇到这些问题自然是如鱼得水，游刃有余。

➢ 策略与解决方案的构造

既然是构造，那么一定和数据结构息息相关。恢复性构造让我们体验了一回“建筑家”的感觉，不过更多的时候，我们面对的是“技术顾问”这样的职业，总是有一些郁闷的国王或者不合时宜的天灾人祸让我们出主意。这时最重要的就是寻求一个整体与部分同时满足，与题目要求息息相关的性质，并将其作为我们使用数学归纳法解题的契机。

【例 7】Royal Federation

Fooland 王国两两城市之间有且只有一条无环路。现在国王要改组整个王国为联邦制。每个州为防止浪费最少 B 个城市，为有效率最多 $3B$ 个城市，同时恰有一个行政中心。行政中心可以不在州里，也可以共用，但州中的每个城市都必须能到达行政中心，途中不出州。请设计满足要求的改造方案。

这道题在国家集训队 2004 年的 SGU 表格中有所涉及，大家对它一定比较熟悉了，基本思想是贪心构造，但是假如我们自己去想算法却未必那么顺利。我们尝试使用结构归纳法诠释其中的解题之道。

归纳假设：　已知如何分划一个规模较小的有根树，每部分的城市数满足 $B \le x \le 2B$，剩下的城市与根连通，数目满足 $x \le B$。

拿到问题后我们首先对根的每个子树使用归纳假设，再累加所有剩下的城市数，每凑够 B 个城市作为一个州，首府设为当前的根。由于每棵子树剩下的城市数目满足 $x \le B$，这样构造的州中城市数目不会大于 $2B$，最终剩下的城市数不会大于 B，归纳步骤完成。最后只需要把剩下的城市并入与其相邻的一个州就可以满足城市数目 $x \le 3B$。这道题的解题方法比较朴素，主要技巧就是找到了一个关键的性质作为归纳步骤之间衔接的桥梁——“剩下的城市与根连通，数目满足 $x \le B$。”这种思维应用起来目的更明确，方向感更强。

【例 8】Electricity

由于最近北美地区的停电事故，政府决定重组供电网络。网络是一个有向图，为了系统稳定，在这个有向图中没有环。政府现在没有钱彻底改造网络，只能改变一些边的方向。平稳起见，每天只能更改一条边，而且任何时刻都不能有环。请设计满足要求的改造方案。

重边的情况可以特殊处理，以下我们假设不存在重边。上一道题中我们自己寻找了一个关键性质来完成归纳，这一道题的题目中有现成的性质 DAG（有向无环图），于是我们使用时只需套用“拿来主义”的方法。之所以这里还要增加这道例题，更多的是想介绍一下两重归纳法的应用。很多时候，由归纳假设证明归纳结论的过程并不是一帆风顺的，其中需要很多因题而异的方法，当然也可能是数学归纳法：这道题目就是一个典型的例子。

归纳假设：　对一个顶点数少于当前图的子图可以设计满足要求的方案。

在应用归纳假设之前我们要从原图中删除至少一个点。为了不出现环，我们不妨找一个在末状态中没有出度的点 A，这可以通过拓扑排序来实现。图中所有指向 A 的边都不需要变化，而所有从 A 出发的边需要反向。怎样把这些边反向才能每一步都不出现环呢？

归纳假设：　只要减少至少一条边，就能找到使剩下的边依次反向的方法。

这么多归纳假设，听起来像童话故事里的魔咒对吗？没关系，很快它们就能变成现实。现在，选择终点在原图中拓扑序最靠前的那条边（设为 $A \to B$ ），把它先反向为 $B \to A$ 。假如此时出现环，环上必然有一个新的从 A 出发的边 $A \to C$ ，也就是说此环可以表示为 $A \to C \to \ldots \to B \to A$ ，但是别忘了， B 的拓扑序是最大的！ C 的拓扑序一定在 B 后面， $C \to \ldots \to B$ 这样的路径怎能存在！于是我们保证这一次更改后不出现环。现在我们减少了一条需要更改的边 $A \to B$ ，剩下的事情就可以统统交给归纳假设了。

进一步分析发现，我们在处理 A 的过程中，其他点之间的边没有变，因此图中除 A 外的拓扑序不变。当 A 点处理完的时候，我们从初末两状态的图中都删除 A 点，其余点的拓扑序仍然不变。因此拓扑排序只需要进行一次，算法也就可以一句话搞定：以起点在末状态中拓扑排序的逆序为第一关键字，终点在初状态中拓扑排序的顺序为第二关键字，将边排序输出。如果不是仰仗了数学归纳法的威力，怎么能想到这个简单的算法属于这样一个貌似复杂的题目呢？

数学归纳法与算法优化

这里所说的算法优化，不是汇编层面上的常数优化，而是同一个算法的不同实现，以及同一个模型的不同处理所造成的运行效率的差异。既然是优化，就带有几分技巧性与偶然性，缺乏一些通用的章法。数学归纳法正好可以为我们弥补这个缺憾。对于许多算法的改进，我们都可以从中找到数学归纳法的影子。而数学归纳法本身的简约明了，让我们能用一个统一的视角来看待这些优化方案。解题时只要潜心挖掘，就一定能从数学归纳法的不同环节中找到改进的地方。

➢ 巧妙选择归纳对象

【例 9】Explaining the Stock Market

股市瞬息万变，可是 Mark Stockle 想找出价格变化的规律。他只知道两种最简单的规律：单调上升和单调下降。于是他想把价格序列划分成几个子序列来研究，怎样才能使划分的个数最少呢？注意单独的一天不能算是单调子序列。

很容易从任何一个子序列中拆除一天加到单独的一天上，所以单独的一天不能算是单调子序列这条规则只有在天数太少周转不开是才会派上用场，完全可以作为特殊情况。这道题本身无疑可以使用搜索算法。大家知道，搜索方法的本质正是状态间的转移，下面的归纳假设直接以数列长度作为归纳对象，易于理解：

归纳假设：　已知一个比原数列短的数列如何得到最优解。

这对应了一个暴力的搜索算法，每次尝试从原数列中抽掉一个单调子序列，并计算余下的数列的单调划分，直到数列变为空列。在题目时限范围内这个算法是可以 AC 的，不过由于实现上的各种限制，以及每次状态转移都必须设法枚举所有的单调子序列，算法效率并不高。这个算法还有很多可改进的余地，比如限制抽取顺序等方法都有可能奏效。然而，对于算法本身焚琴煮鹤导致的效率问题，如果拘泥于框架小修小补，自然不一定能达到大幅度优化算法的目的。

不妨先限制问题中只允许取上升列，让我们设计一个 one-pass 算法。从前到后扫描，每当数列末端增加一个数，其实就是要求我们将其加入已经存在的某个终点比这个数小的上升列中，并提升此序列的终点。新开一列的情况相当于把空列终点的 $-\infty$ 进行了提升。如果有多个序列可以提升，我们就选择终点最大的那个“幸运列”，因为终点小的潜力大，用了可惜。正规一点说，如果截至某一时刻，两个部分解的子序列终点为 $\{a_n\},\{b_n\}$ 且满足 $a_i \le b_i, i \in [1,n]$，而 $\{b_n\}$ 最后产生了最优解，在 $\{a_n\}$ 上可以如法炮制产生划分个数一样少，甚至更少（如果第 i 个子序列之后没有拓展，且 $-\infty = a_i < b_i$）的最优解，所以考虑 $\{a_n\}$ 就足够了。我们把这样产生的部分解叫做“最有潜力解”，就可以使用数列不断增长的前缀作为归纳对象了。

归纳假设：　已知如何求解数列某个前缀的最有潜力解。

题目中的下降列与上升列同时出现，最有潜力解中也就会包含若干个上升列与下降列（当然只需要保存它们的终点），而我们在归纳过程所要做的就是分别尝试将增加的那个数加入上升幸运列和下降幸运列中使用归纳假设，取最终子序列数较小的一个。新的算法对前若干个数构成的部分解进行归纳，理论上需要 $O(2^n)$ 的时间完成搜索，不过作者加上最优性剪枝后所有的数据都可以瞬间出解。

➢ 力求完善归纳基础

【例 10】快速排序的优化

快速排序对我们而言已经非常熟悉了，如果我们用归纳式算法思想来理解，就是在归纳的推证阶段使用划分序列的方法将问题化简到归纳假设，算法在这里的表现也是优雅而迅速的。可是面对一些比较短的序列，不断的分治与递归却严重影响了算法效率。要优化这个归纳算法，看来只有从归纳基础部分着手。原来的归纳基础是若序列长度为 1 则不必排序，我们可以将其调整为若序列长度小于一个预先设定的数值就改为使用插入排序或堆排序。这样虽然增加了归纳基础的时间开销，但是递归次数减少了，性能有所提升。

事实上，目前得到广泛应用的 Introsort 等算法正是应用了这样的思想，从而大大提高了最坏情况和平均情况的排序速度。可见，在归纳基础上优化算法时要力求完善，理由是归纳基础本身在算法中往往多次被调用，花一点时间做一点优化也是值得的。我们时常提到的预处理加速的方案很多时候也都能与这里提到的方法不谋而合。

➢ 慎重选择归纳方向

【例 11】多项式求值的 Horner 规则

已知 x ，求多项式 $\sum_{i=0}^{N} a_i x^i$ 的值。这个问题挺简单的，只不过归纳假设为式子中的前 K 项之和已经求出是不够的，需要已知 x^K 的值完成接下来的计算。那我们天天用的 Horner 规则又是怎么回事呢？调整归纳方向，反过来算 $\sum_{i=1}^{N} a_i x^i = x\sum_{i=1}^{N-1} a_i x^{i-1} + a_N$ 试试，简单了不少吧。归纳式算法中先进行的部分承受的操作要多一些对吗？那就顺水推舟，让需要乘法次数较多的高次项先算，这也提示我们选择归纳方向要看问题本身的特点。

【例 12】建立二叉堆的两种方法

从一个无序表建立二叉堆同样有两种不同算法：自顶向下与自底向上。我们已经知道自底向上建堆的方案的时间复杂度为 $\Theta(n)$，要优于自顶向下的 $\Theta(n\log n)$。这应当属于不同归纳方向对算法造成的影响，可是原因何在呢？直接从这两个复杂度分析的过程入手难以看清其本质，如果从数学归纳法中先易后难的角度就看得很清楚：归纳式算法中先进行的部分由于问题规模尚小往往会有速度上的优势，堆的特点是上轻下重，不难看出如果能把这一优势送给底端占多数的节点就可以事半功倍，事实也确实如此。

➢ 适当加强归纳假设

【例 13】Yet Another Digit

一个数的十进制表示我们很熟悉，二进制也并不陌生，共同性质就是任何整数都只有唯一一种表示方法。不同的是，如果我们保持二进制中的每一位的权值，同时允许其中出现“2”，并称之为整数的 RBR 表示的话，并不满足此性质。现在给定一个大整数，请求出它的 RBR 表示方法数。

运用第二数学归纳法思想，想到一个可行的算法并不困难。设答案为 $\mathrm{f}(n)$，最直接的归纳假设如下：

归纳假设：　　所有 $\mathrm{f}(k), k<n$ 的值可以求出。

除去边界情况，问题都可以使用递推关系式 $\mathrm{f}(n)=\begin{cases}\mathrm{f}\left(\frac{n}{2}\right)+\mathrm{f}\left(\frac{n}{2}-1\right) & n\equiv 0(\bmod 2)\\ \mathrm{f}\left(\left[\frac{n}{2}\right]\right) & n\equiv 1(\bmod 2)\end{cases}$ 解决。粗略的估计得到复杂度是 $\mathrm{O}(n)$，考虑到两种情况不可能总是出现第一种，细致一点的分析可以得到 $\mathrm{T}(n)=\frac{\mathrm{T}_{\mathrm{odd}}(n)+\mathrm{T}_{\mathrm{even}}(n)}{2}=\frac{\mathrm{T}\left(\frac{n}{2}\right)+\left(\mathrm{T}_{\mathrm{odd}}\left(\frac{n}{2}\right)+\mathrm{T}_{\mathrm{even}}\left(\frac{n}{2}\right)\right)}{2}=\frac{3}{2}\mathrm{T}\left(\frac{n}{2}\right)$，从而有总复杂度 $\Theta\left(n^{log_2 3-1}\right)\approx\Theta\left(n^{0.58}\right)$。本题中大整数的范围可能达到 10^{100}，这样高的复杂度我们无法承受，退一步讲，即使我们用记忆化思想成功处理了重叠子问题，降低了时间复杂度，这样大的空间复杂度依然使我们望而却步。对付此类问题最有力的武器就是记忆局部化，分析出重叠子问题的出现方式，并将其添加到我们的归纳假设中。注意到计算时 $\mathrm{f}(n)$ 与 $\mathrm{f}(n-1)$ 成对出现，所以我们设 $\mathrm{g}(n)=\mathrm{f}(n-1)$，并因此有了新的归纳假设：

归纳假设：　已求出了 $\mathrm{f}\left(\left[\frac{n}{2}\right]\right),\mathrm{g}\left(\left[\frac{n}{2}\right]\right)$ 的值。

这次我们发现问题不再以第二数学归纳法的形式出现，变得类似第一数学归纳法，尤其是如果从二进制的角度看，归纳假设等价于求出了某个前缀的表示方法数。这样做的好处是归纳过程中只使用一个子问题，可以记忆化求解，更可以化递归为迭代。下面的递推公式佐证了我们的猜想：

$$\mathrm{f}(n)=\begin{cases}\mathrm{f}\left(\left[\frac{n}{2}\right]\right)+\mathrm{g}\left(\left[\frac{n}{2}\right]\right) & n\equiv 0(\bmod 2)\\ \mathrm{f}\left(\left[\frac{n}{2}\right]\right) & n\equiv 1(\bmod 2)\end{cases},\mathrm{g}(n)=\begin{cases}\mathrm{g}\left(\left[\frac{n}{2}\right]\right) & n\equiv 0(\bmod 2)\\ \mathrm{g}\left(\left[\frac{n}{2}\right]\right)+\mathrm{f}\left(\left[\frac{n}{2}\right]\right) & n\equiv 1(\bmod 2)\end{cases}$$

实现时使用 f ， g 两个变量从高位到低位扫描二进制大整数，某位为 0 就 $f \coloneqq f+g$ ，否则就 $g \coloneqq f+g$ ，总时间复杂度控制在 $\Theta(\log n)$ 。

启发作用与美学价值

大家一定注意到，以上对数学归纳法的讨论，很多尝试都在企图揭开解题之道神秘的面纱，尽量把信息学题解中的“神来之笔”予以化解，还其本来面目。这是因为作者认为，算法还是应以简单自然为上，总是故弄玄虚，让人意想不到的算法谈不上有什么美感，与其高谈阔论这些只会让人感到困惑的“解题之道”，不如实实在在地去尝试。数学归纳法正是这样一种尝试的手段，因此我们称之为一种解题之道，一种从简单的做起，从不同角度观察的解题之道。

当年吴文俊院士把机器证明引入到数学中的几何学研究，可以让计算机来完成通常需要许多探索性思维来完成的几何证明。既然存在“证明的方法”，那么我们中的许多人一定会想：是不是可以找到一种“算法的算法”，能够帮助我们寻找可行的解题方法？理论上这样一种绝对通用的算法当然是不可实现的，但这并不妨碍我们寻找一些通用的解题策略，并把它们应用到我们的解题实践中去。如果说解题之路中需要的是数学中的那些定理、方法，那么解题之道中需要的就是数学中的思维策略。我们相信在解题之道中数学归纳法的应用只不过是冰山一角，更多的思维策略还有待我们去发掘……

问题与缺陷

没有什么方法是万能的或是一成不变的 数学归纳法虽然是较为通用的一种分析方法，也免不了有一些问题与缺陷，以及不适用的情况。让我们本着辩证分析的原则一同来分析一下数学归纳法的一些不足与我们的应对之道。

➢ 理论上是否欠完备

看到这里一定有同学会问：数学归纳法可以用来证明其他算法的正确性不假，可是它本身的正确性又如何呢？数学归纳法的正确性依赖于皮亚诺公理系统中的归纳公设。根据哥德尔不完备性定理，任何公理系统正确性的证明都不能在该系统内部完成。虽然如此，由于我们在应用中不会接触到关于公理系统的正确性的问题[2] ,所以说作为演绎逻辑的一种方法，数学归纳法在我们所涉及的范围内基本还是靠得住的。不过数学归纳法要从从有限涉及无穷，却没有正确性的直接证明的特点，不能不说是一种缺憾。

➢ 应用上是否较繁琐

信息学中涉及的知识千变万化，算法模型层出不穷。在我们的学习应用中，尤其是在比赛中，如果每逢贪心算法必定证明正确性，每逢设计算法必定先写归纳假设，无疑是为问题本身增加了不必要的开销。这时，从能用、会用上升到活用、善用，就是我们轻装上阵的必要武器。

➢ 不适用的问题

作者从实践中发现，数学归纳法可能不适用的问题包括：线性规划、整数规划相关的问题和一些可以规约到线性规划的问题（如图论中的网络流），部分最优化问题，大部分的非完美算法，数值算法、应用了矩阵或其他数学模型的问题等。虽然如此，由于作者才疏学浅，解题经验有限，在归纳中难免有以偏概全之处，还是希望大家提出意见或者给出补充。

---

[2]正如我们不会因为做不出题就猜想说“这个命题是不是传说中不知在哪里好像见过或者没见过的什么什么不可判定命题”一样。

后记

我学数学竞赛时曾读过单墫老师写的《解题研究》一书，其中对解题之道要简单自然，直剖核心的要求一直不能忘怀。信息学解题也是这样，这篇论文中提到的数学归纳法只是参悟解题之道的一个手段与方法。正因为如此，这篇论文开头的时候本来想写成一个数学归纳法的专题，后来尝试揭开解题之道神秘的面纱的同时，却不知不觉写得洋洋洒洒万言有余：虽然这一点让我也有点始料未及，但还是希望能对大家有所帮助。另外我想借此机会感谢我们中学的王晋生、马宏春老师对我平时的鼓励与关心，集训队唐文斌、胡伟栋教练对本文的关注，以及其他默默支持着我的人。希望大家将自己的心得不吝赐教，或者将找到的错误与疏漏之处与我交流（张昆玮，aceeca.135531@gmail.com）。

题目来源[3]

文中例题除经典问题外部分应用了 Andrew Stankevich's Contest 的题目。由于涉及的题目确实较多而篇幅有限，请大家原谅此处没有附上题目全文，只是在题目名称上做了指向题目出处的超级链接。对此有兴趣的同学可以同时参阅作者的解题报告与源程序或与作者联系。

[3]作者在引用很多题目时做了翻译，为了节省大家的阅读时间，只保留了题目中与本文相关的核心内容。希望有兴趣了解题目详细背景（很多别致而有趣的题目背景经过了其原作者的精心设计）的同学阅读题目原文及有关内容。