歧路修远，上下求索——例谈信息学竞赛分析中的“深”与“广”

肖汉骏

指导教师

陈颖

福建省福州第一中学

高三（8）班

2008年1月

摘　　要

本文是作者对信息学竞赛新变化的一些思考和总结。着重阐述了信息学竞赛分析中“深”与“广”的特殊意义、做到这两点的方法，以及平常的学习过程中应该注意的一些地方。

关键词： 思维方法，深入分析，多角度，归纳

目　　录

[1]原创问题

[2]原创问题

[3]Ural1553. Novosibirsk SU Contest. Petrozavodsk training camp, September 2007

[4]2004 年第二届广东省大学生程序设计竞赛

第一章　汪洋恣肆:“深”与“广”在信息学中的意涵

人们往往用“深”与”广“来形容海洋，但另一方面，人类的脑海却比任何一个大洋都来得深邃、来得广阔。作为思维分析能力充分展示的舞台,“深”与“广”在信息学竞赛中又有什么特殊的含义呢？又如何在分析中做到”深“与”广“呢？本文将和大家一同探索这些问题。

1.1　何为“深”？

问题分析中的“深”，有这么几层意思：

一是**层次性**。一个问题在分析之初往往不那么容易，有的是高维数的，有的是条件复杂的。层次方面的观察就是要抓住问题的来龙去脉，从低维的情况、条件简化的情况来考察问题，得出一些性质。再由浅入深，分析这些性质在高维情况、条件复杂的情况下产生的变化，以求突破。

二是**连贯性**。分析问题的过程往往是漫长而坎坷的。有时候灵光一闪，似有所得，有时候又陷入困境，卒无所获。倘若走一步看一步，想到哪里就分析到哪里，思维就容易混乱，抓不到目标。反之，若能锲而不舍，沿着确定的方向步步深入，充分利用每一步分析的结果，就容易挖掘出一些有用的东西来。

三是注意**要素之间的关系**。哲学告诉我们,“事物是普遍联系的”。在信息学问题中，要素之间的关系也是值得深究的对象。问题并不是要素的简单相加，要素之间的相互关系也是客观存在的，而且更为本质和深刻。要素间诸如单调性、周期性等等关系，如果能有意识地深入研究，往往可以成为解题的突破口。

1.2　何为“广”

而问题分析中的“广”，内涵就更为丰富：

一是开阔的眼界。在考场的仓促时间内，很多问题要获得完美的解决是困难的，甚至是不可能的。这就需要我们发挥想象力，运用**各种策略**解决问题。可以是随机化算法，也可以是贪心算法，甚至针对数据可能的形态，分别设计算法。随着策略的不同，程序的实际效果也大相庭径。

二是广阔的思路。一个相同的问题，从**不同角度**看往往有不同结果。如果能把不同角度的分析结果综合起来，就能较为全面地理解问题，一些隐蔽的信息在各个方向的探测下，也就没有了藏身之处了。

三是丰富的分析手段。分析手段是思维的利器，能抓出问题的本质所在。但是，就像每种武器都有自己的长处和短处，分析手段也有各自的特点。比如，数学推演精确但抽象，图形性质直观但不易量化。若能**掌握各种分析手段**，就似精通了十八般兵器的武林高手,“谈笑间，问题灰飞烟灭”。

第二章　谋定而后动：深入分析问题本质

下面，以两道题为例，谈谈如何在问题分析中做到“深”。

2.1　例一：n 位数枚举[1]

问题描述：

统计所有 n 位数中，不包含数字 0 且被 9 整除的数的个数。

数据范围：

$1 \leqslant n \leqslant 500$

初步分析：

题目叫我们统计，很容易想到枚举算法。

算法一、在 n 位数中找 枚举所有 n 位数，再判断是否包含数字 0 和被 9 整除；

时间复杂度： $O(10^n)$

利用特殊条件深化分析：

注意问题中要求统计的是“不包含数字 0”且“被 9 整除”的数，这便是问题的**特殊条件**。从这两个角度，可以得出两种不同的算法：

算法二、在 9 的 n 位数倍数中找 利用被 9 整除这一条件，我们可以直接枚举这个 n 位数是 9 的多少倍，对范围内的倍数一一检查，效率至少是原来的 9 倍；

时间复杂度： $O(10^n)$

[1] 原创问题

算法三、在不含 0 的 n 位数中找　如果我们把 n 位数看成是 $0 \sim 9$ 的一个可重复的排列，那么不包括数字 0 的 n 位数，就是 $1 \sim 9$ 的一个可重复的排列，再对这些 n 位数进行是否被 9 整除的判断；

时间复杂度： $O(9^n)$

然而这些指数级别的算法，对于 n 的规模来说，显然是不够的。

由浅入深，层层递进：

回顾刚才的几种想法，算法一最简单直接不过；算法二利用了倍数的性质，只是稍有改进；算法三提出了把 n 位数看作是可重复排列的想法，自然地去除了 0 的干扰。但我们发现，算法二和三都是利用某个特殊性质，还未将两个性质结合起来使用。于是在算法三的基础上，结合 9 的倍数的特殊性质，我们提出算法四。

算法四、利用 9 的倍数的特殊性质　被 9 整除的性质决定了 n 位数的数字和也要被 9 整除，而当前 $n-1$ 位定下之后，由于 $1 \sim 9$ 除以 9 的余数各不相同，最后一位实际上可以被唯一确定。至此，问题化为 $1 \sim 9$ 的长度为 $n-1$ 的可重复排列数是多少，答案也是显然的：9^{n-1}。

时间复杂度： $O(\log n)$ [2]

至此，问题获完整解决。

枚举对象和效率的关系：

正如刚才我们所看到的，枚举也有低效和高效之分。然而造成这种区别的原因是什么呢？

如图 2.1 所示，A 区域表示所有的 n 位数，B 区域表示 9 的 n 位数倍数，C 区域表示不含 0 的 n 位数，D 区域则是我们要统计的数的集合。从集合的观点来看，枚举的一般算法就是找到一个包含最终目标的集合，作为我们的枚举对象，例如图示中的集合 A、B、C，然后对枚举出来的元素逐一检查，排除掉多余

[2] 采用快速幂算法，并假设高精度运算为 $O(1)$

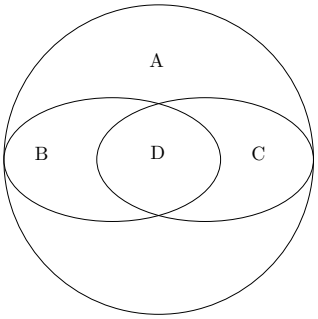

图 2.1: 集合的角度

的部分，从而得到最终目标。而枚举效率的高低，就和枚举对象的选择息息相关了。如果对应集合的元素个数越多，则效率越低，反之，则效率越高。

以上文提到的四种算法为例：算法一几乎就是按照题目所叙述的直接操作，对应的集合 A 也是最大的；算法二和算法三分别把握住了问题某个方面的性质，但并不全面，所以时间复杂度只是稍有优化，对应的集合 B、C 稍有缩小；算法四则把算法二和算法三结合起来，创造性地建立起原问题和另外一个易于计数的问题的一一对应关系，直接计算出集合 D 的大小。

如何选择合适的枚举对象：

枚举对象的重要性已经被充分证明了，但是如何选择合适的枚举对象呢？其实从图 2.1 中可以得到一些启示：一开始，我们明确题意后，接触的仅仅是问题最原始的形态，如图中的 A 部分，枚举对象是粗糙的；随着我们对问题特殊性质的注意，并加以挖掘和利用，就有可能由表及里，得到较好的枚举对象，如图中的 B、C 部分；而当我们注意到问题多种性质间的区别和联系，就可以综合地把握住问题本质，真正选择出合适的枚举对象。

这个过程，实际上也就是对问题的分析步步深入的过程。我们分析地越深入，越接近问题的本质，就越可能找到简单的算法解决问题。在分析中对**特殊性质**的注意，对**多种性质间相互关系**的注意，能帮助我们更准确地把握住问题的内涵和外延，甚至实现算法的突破。

2.2　例二：推箱子[3]

问题描述：

小 Z 是一个聪明且贪玩的孩子。一天，正当他在网上到处寻找乐趣的时候，一个经典但困难的游戏进入了他的视野——推箱子。

推箱子游戏是在一个由墙和空地组成的地图上进行的，其目的是要将所有箱子推到指定的位置。玩家控制一个小人，每次操作使小人向上下左右四个方向之一尝试移动。如果是空地，则只改变小人的位置；如果是箱子，且箱子的相同移动方向是空地，则小人和箱子同时移动一格；其余情况移动无效。

不看不知道，一看吓一跳，这个推箱子游戏的箱子数实在太多了！小 Z 看得头晕脑涨，于是他下决心只玩一个箱子的游戏。追求完美的他总是想用最少的操作完成游戏，你能帮助他完成这个游戏吗？

输入数据：

第一行是 n, m, k，分别表示地图的行数、列数以及在这个地图上进行的游戏次数。

接下来 n 行 m 列描述这张地图。墙用 '#' 表示，空地用 '.' 表示。

接下来 k 行每行 6 个整数，分别表示小人、箱子和目标位置的行号和列号。

输入数据保证小人不能走出地图，且小人和箱子不会重合，也不在墙上。

输出数据：

对每一次游戏都输出一行，为完成这个游戏所需要的最小操作次数。如果不能完成游戏则输出 'No Answer!'。

数据规模：

$3 \leqslant n, m \leqslant 30$

$1 \leqslant k \leqslant 1000$

[3] 原创问题

初步分析：

由于要求的是操作步数最少，就涉及到地形，箱子位置等诸多因素。在找不到合适的模型的情况下，搜索无疑是一种有益的尝试。

暴力搜索：

广度优先搜索。每次向四个方向进行移动尝试。基本的剪枝是避重性剪枝，搜索的一个状态可以用小人和箱子的坐标表示。如果碰到之前搜索过的就不必搜索了。

这个搜索的时间复杂度可以用状态的数量衡量，最坏情况下，小人和箱子的坐标可以取遍所有空地，故得：

时间复杂度： $O(kn^2m^2)$

两种移动的分离：

相对于问题的规模来说，算法一的复杂度还不尽人意。例一的分析过程提示我们，如果能发现问题独有的性质的话，可能会有新的突破。

再次回顾问题[4]，我们发现小人的运动实际上是有一定规律的：或者是为了推动箱子，或者是进行一系列移动，为下一次推动箱子做准备。然而我们的搜索程序并不能让我们的小人时时刻刻都遵守这两条规则。更多地，当小人不推动箱子时，它只是不住地上窜下跳。若强制让小人沿着最短路向它的下个目标走去，似乎可以避免不必要的搜索。

双重搜索：

根据上述思路，我们将搜索对象从操作本身转移到箱子的移动路线上。由于箱子移动之时小人必然在箱子四面的一面，故不同的状态数目也只有 $O(nm)$ 了。而小人不推动箱子的路线，则可以通过一个额外的广搜解决。一般情况下，这个额外的广搜应该很快结束，程序效率提高得也很明显。然而在极端情况[5]下，这个广搜还是有可能遍历整个棋盘的。

[4] 又或者多玩几盘推箱子游戏后... :-)

[5] 比如死胡同什么的。

另外需要注意的是，由于我们并不是严格按照操作数从少到多扩展状态，故有时候对同一个状态需要重复搜索。但由于重复搜索的条件比较苛刻，对时间复杂度的影响微乎其微。

时间复杂度： $O(kn^2m^2)$

相同的地图：

分析到这里，问题似乎难以有所突破了。但是在之前提出的算法中，我们都没有利用一个性质：回答的是 k 个**不同的问题**，但使用的是一个**相同的地图**。不变的地图难道就没有不变的信息么？如果能通过某种预处理，将地图的信息挖掘出来，或许就能“柳暗花明又一村”。

从搜索到最短路：

在这个想法的启发下，再仔细琢磨之前的双重搜索，不由得让我们眼前一亮：那个额外的广搜所解决的部分，不就恰恰脱离了具体问题，而只跟地图有关么？只要事前枚举箱子的位置，进行必要的搜索和储存，在后面的 k 个问题中，就不需要新的搜索了。

然而分析并不能就此止步。小人不推动箱子的移动可以预处理，推动箱子的移动不也可以预处理么？如果我们将算法二中的每个状态作为点，而将若干操作达到的状态转移作为边，边权赋值为操作的次数，那么原问题就可以用简单的最短路模型表示了！需要注意的仅仅是起点、终点最多可能均有四个，只要增设新的起点、终点，问题并不难解决。

时间复杂度： $O(n^2m^2 + knm)$ [6]

回顾：

应该说，以上的三种算法，都是建立在分析的不断深化上，相互之间是有深刻联系的。如果没有暴力搜索的尝试，或许我们就不会发觉小人无谓的上窜下跳，从而将小人的两种操作分离开来；如果没有双重搜索中对状态表示的改进，可能我们就难以将搜索问题图论化，以全新的算法解决原题。

[6] 使用 SPFA 算法，由于是稀疏图，且有很强的层次性，实际效果极好。

所以，“深入分析问题本质”的想法，还要求我们抓住那些**尚未利用**的性质和条件，寻找和**原有分析结果**结合的可能性，抽丝剥茧，一步步把问题本质揭露出来。

2.3　将问题纵向延伸

值得一提的是，解题的成功不意味着问题的结束。适当地对问题增减条件，思考新情况下问题产生的变化，则更能深刻地理解问题。

以第二道例题为例，推箱子是大家都很熟悉的问题，而这个问题的一般解法就是迭代加深搜索，而表现比较好的优化就是文中所提及的双重搜索。作者正是在思考双重搜索的时间效率时，注意到第二重搜索只跟地图有关的性质，从而对问题加以变化，使得搜索问题得以图论化处理。

而第一道题也可以纵向延伸：问题中不包含的仅仅是数字 0，整除性要求也仅仅是不被 9 整除。如果把不包含的数字推广到 $0 \sim 9$ 的任意子集，整除性也推广到除以 p 的余数不能是 $r_1, r_2, \cdots, r_k$。问题又将有新的变化。具体问题和解题报告可以参考我的原创比赛。

第三章　横看成岭侧成峰：多角度思考

接下来，再以两道题为例，谈谈如何在问题分析中做到“广”。

3.1　例三：Caves and tunnels [1]

问题描述：

给出一棵 n 个结点的树以及 Q 个操作。每个操作或者增加某个结点的权值，或者询问两个结点路径上的最大权值。权值一开始均为 0。

数据规模：

$1 \leqslant n, Q \leqslant 100000$

分析：

如果是在线性表上进行操作，那么就是经典的动态 RMQ 问题，用线段树可以轻松解决。然而本题的操作对象却是一棵树。线性表的操作如何推广到树呢？是仿照线性表的解决方法进行分治？还是通过化归，用线性表解决？方法似乎有很多，我们可以一个个尝试。

树的分治：

注意到我们解决线性表问题时，采用的基本思想就是分治。当问题推广到树的时候，对树进行分治似乎也是种不错的想法。

然而稍加尝试就会发现，由于树不具备线性性质，分治点和询问点的关系难以表达。故不能和线性表一样，在区间被包含时直接将区间的统计数据调出。多方努力无果后，分治的尝试宣告失败。

[1] Ural1553. Novosibirsk SU Contest. Petrozavodsk training camp, September 2007

化归到线性表：

另外一种想法，就是将问题化归到线性表上解决。“树内任意两点只有一条路径”，这个性质也的确和线性表有关系。

最简单的想法，就是为每个询问到的路径都建立一张线性表。这当然太慢了：路径数实在太多。但是，如果把路径数缩小一些呢？

在这个想法的启发下，我们任意指定树中一点为根，就可以把连接两点的路径以它们的最近公共祖先为分界点，分为两段处理。这样，所有的路径都指向根。如果为每个结点到根的路径都建立一张线性表，诚然可以解决问题，但其时空代价却难以承受。有没有更好的方法呢？

两种想法的结合：

刚才的两种想法，虽然都没有解决问题，但其中的思路却是可以借鉴的。多角度思考并不是打一枪换一炮，有意识地寻找不同想法结合的可能，或许可以打开局面，走出困境。

注意到线性表的想法之所以遇到阻碍，乃是因为建立的表的数量实在太多。而分治不恰恰就是减少问题规模的利器么？在这个思路的启发下，我们每次为当前树中最深的叶子结点到根的路径制作线性表，其余的分叉递归处理。就是将一棵树看成是由多个链组成的。将一颗树上的问题转化到多个链上的问题。可以证明，每次询问涉及到的线性表数目最坏情况下是 $O(\sqrt{n})$ 的。

时间复杂度： $O(Q\sqrt{n})$

3.2　例四：Paper [2]

问题描述：

小 D 要读 n 篇文章，每篇文章有一个阅读时间 T_i，价值 V_i . 读一篇文章的代价为从 0 时刻到读完这篇文章的时间乘上它的价值，小 D 希望安排一个读文章的顺序使得总的读文章的代价最小.

[2] 2004 年第二届广东省大学生程序设计竞赛

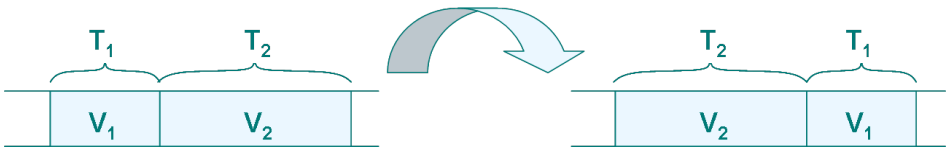

图 3.1: 交换顺序之后

另外，有一些文章的作者是相同的，小 D 希望按照这个作者写文章的先后顺序来读这些文章.

数据规模：

$1 \leqslant n \leqslant 50000$

初步分析：

这个问题要求的是一个最佳的阅读顺序，使得总代价最小。但又附加了一定限制，即同一作者所写的文章要按时间顺序阅读。

直接考虑问题本身似乎不容易找到很好的方向，所以我们不妨先尝试弱化条件，由浅入深地解决原问题。

去除限制条件：

我们尝试去除作者因素，考虑这个特殊情况下问题的解。

问题要我们求一个最优序列。显然，最优序列不是胡乱排列的，这个序列必然有某种特殊处，才使得它成为最优。而最优序列的特殊之处就在于：经过任意改变后，得到的结果都不比原序列优。

在序列中最简单的改变莫过于交换相邻元素了[3]。我们就照着这个思路进行深入分析：

考察相邻的两本书，设它们的阅读时间为 T_1, T_2，价值为 V_1, V_2。如图 3.1 所示，交换这两本书的阅读顺序后，第一本书的阅读完成时间推迟了 T_2，第二本书则提前了 T_1，所以，总代价的变化为 $T_2V_1 - T_1V_2$。

对于最优序列，一定有 $T_2V_1 > T_1V_2$，也即 $\frac{V_1}{T_1} > \frac{V_2}{T_2}$。相邻两项有这个关系，

[3]只改变两个元素，其余元素都不影响。

那么，整个序列不就是按照 $\frac{V_i}{T_i}$ 从大到小的顺序排列么？至此，特殊情况已被轻松解决。

最特殊的一般情况：

回到一般情况，我们该如何处理作者因素对阅读顺序的影响？仍然应用**从特殊到一般**的思想展开分析：

在所有一般情况中，只有一个作者的情况是最特殊的，但其顺序已被固定，没有研究的价值；接下来，就是只有两个作者的情况了。而在所有两个作者的情况中，某位作者只写了一篇文章的情况又最为特殊。我们不妨来考察一下这篇独立文章应何时阅读。

同样的，我们通过研究任意改变后的总代价的变化，考察最优阅读顺序所应具有的特殊性质。由于相同作者的文章顺序固定，不可改变，所以，改变必须包括那篇独立文章。而根据刚才的分析，最优序列中相邻的可交换元素满足关系：

$$\frac{V_i}{T_i} > \frac{V_{i+1}}{T_{i+1}}$$

也即，独立文章的比值要小于前面的，大于后面的。然而这又意味着什么呢？单纯从数学推演的角度出发，满足上式的位置可能有很多，没有好的性质能够加以利用。而二元组比值的形式，不禁促使我们联想到直线的斜率。这里，不妨把问题图形化，用广泛的分析手段尝试解决问题。

以形助数：

我们将每篇文章看作一个向量 (T_i, V_i)，则比值 $\frac{V_i}{T_i}$ 就是这个向量的斜率。我们将这些向量首尾相接，在平面直角坐标系中表示出来。那么，独立文章的插入位置在图形中可以形象地表示为“凸点”，如图 3.2：

当然，只注意问题一个方面的性质是有悖于“**广**”的要求的。注意那些不可插入位置，它们在图形中表示为“凹点”。容易用反证法证明，凹点是不能被新的文章插入的[4]。也就是说，一旦出现凹点，就可以把形成凹点的两个向量合并，去除这个凹点。

[4]否则与前面或者后面的向量交换，得到的解更优。

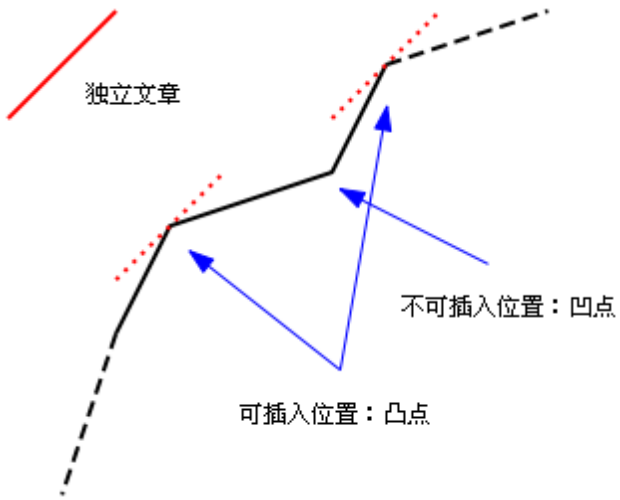

图 3.2: 以形助数

而什么图形不包含凹点呢？是的，就是凸包！而凸包上边的斜率，恰恰是单调递减的！可以预感，我们离最终的答案只有一步之遥。

最终算法：

分别处理每一个作者，求出文章形成的凸包。则该作者所写的文章被凸包上的点划分为若干段，每段的斜率单调递减。而后把所有文章段按斜率降序排序，即为所求的最优序列。容易证明这个序列是合法的，因为相同作者所写的文章段本身就满足斜率降序，故排序不会破坏写作的先后顺序。

时间复杂度： $O(n \log n)$

回顾：

回顾这道题的解决过程：

1. 我们先去除了限制条件，从特殊情况着眼，得出了最优序列相邻元素之间的关系
2. 在一般情况中，我们又选取了最为特殊的两作者情况讨论。得出了插入位置的数学性质。
3. 随后又运用了数形结合的分析方法，从正反两个方面考察了凸点和凹点。

4. 最后联想到凸包，一气呵成地解决问题。

可以看到，问题解决过程中的每一步，都或多或少地和“深”和“广”有所联系。

多角度思考实际上就是力求分析的广度，但这并不是说从不同角度**孤立地**看问题。而是针对问题本身不同侧面的性质，综合地把握住它的内涵和外延、特殊处与一般处。利用多个性质、多个模型解题的时候，也不是简单地相加、盲目地套用，而是将它们有机地、创造性地组合起来，才能起到最好的效果。

3.3　多作横向对比

而在平常的训练过程中，也不要只满足于一种解法。多吸收他人分析中的独特之处，在适当的时候尝试一下，就能起到开阔思路的作用。以前想不到的地方，在接受了新的想法之后，就可能比较轻松的考虑到。

另一方面，不同解法的相互比较也十分重要。不同解法的效果往往不尽相同，若能分析出产生差异的原因，对不同解法的认识也会更加深刻。遇到一题多解的时候，也能更加全面地考虑不同解法的特点，使得决策更为容易。

第四章　磨刀不误砍柴工：对程序设计学习的几点启示

4.1　培养良好的分析习惯

在平常的解题实践中，我们往往依靠经验、不自觉地使用各种分析方法。然而效果时好时坏，难以把握。这正说明了零碎的经验，是不能代替系统的理论的。若能在分析过程中遵循一定的规则，做到有理、有序地分析，就能避免凭借经验带来的不稳定性，这也正是良好的分析习惯带来的最大好处。

4.2　多思考、勤交流

正如前文所述，思考分析的深度和广度，决定了问题解决的好坏。另一方面，信息学竞赛的最终目的，也是培养选手的思维能力。所以，在平常的训练和学习过程中，对问题自觉地从多个角度作深入分析，是很有好处的。同时需要注意的是，如果只是自己单干，或者是小集团内部封闭式的交流，则不容易打开思路，吸收新鲜的想法。反之，若能广泛地了解他人对同一道问题的思路，则有助于更加灵活地分析问题。

至于交流的途径，则不必有所拘束，大可以各显神通。在信息资源丰富、联系日趋快捷的今天，我们既可以从书本和论文中汲取养分，也可以从网络上去粗取精；既可以面对面地讨论，也可以在网络上跨空间的交流。方式可以多样，但目的都是为了提高自己的思维能力。相信只要持之以恒，必然会有一定提高。

4.3　正确对待经典问题

经典问题之所以“经典”，不单是因为其常见，更是因为其中蕴含着基本但重要的信息学思维方法。然而在日常的训练中，经典问题往往只被我们当

作“死”的知识识记，在遇到类似问题时稍加改动就套用上去。这就忽视了经典问题对思维的训练作用。

比如大家所熟知的**快速排序**算法，堪称经典。然而很多同学只限于知道可以利用它进行排序，而没有关注其选择特定数划分数列、进行分治的基本思想。所以，遇到求数列中的第 k 大数的问题，不少同学大多都先进行排序后输出，时间复杂度是 $O(n \log n)$；但是，若能理解快速排序中分治的想法，则完全有可能设计出更快的算法——**快速选择**。

快速选择的基本思想，也正如快速排序一样，先在考察范围内选择一个基准数，则可以将整个数列分为两段或三段。这样以后，就可以依据 k 的大小，确定每一段是不是需要继续排序。这样做的时间复杂度是 $O(n)$ [1]的，达到了理论的下界。

这个例子生动说明了经典问题的重要意义。也告诉我们，再学习经典问题的过程中，应注意从思想本质上理解它们。真正把经典问题看作是特定模型中，各个量之间具备的客观联系的揭示。

4.4　适当的总结归纳

题目做多了，思路也有一定积累了，就有必要好好梳理一下，将那些重要的东西归纳出来。这有很多好处：一是可以巩固记忆，把得来的思路消化掉，成为自己的一部分；二是可以在总结的过程中，横向纵向地比较思路的差异，在碰撞中或许还有新的发现。

正如本篇论文，也是笔者在多年的竞赛过程中，一个比较系统的思考和总结。其中对学习过程中各个阶段常见问题的描写，笔者也是有亲身体验的。所以笔者对总结归纳的重要性深有感触。

[1]更确切地说，如果随机地确定基准数，期望的时间复杂度是 $n + \frac{n}{2} + \frac{n}{4} + \cdots + 1 = O(n)$。

参考文献

[1] 刘汝佳. 搬运工问题的启示. 2001.

[2] 周戈林. 《*Query on a tree*(*spoj375*)》解题报告. 2006.

[3] 杨哲. *QTREE* 解法的一些研究. 2007.

致　　谢

值此论文完成之际，谨在此向多年来给予我关心和帮助的老师、同学、朋友和家人表示衷心的感谢!

感谢陈颖老师对我的关心和指导。

感谢刘汝佳教练，胡伯涛学长对我论文提出的修改建议。

感谢集训队员余林韵，陈丹琦对我的帮助。

感谢福州一中信息组的全体成员对我的支持和鼓励。

谨把本文献给我最敬爱的父母!