压去冗余 缩得精华

——浅谈信息学竞赛中的“压缩法”

安徽 周源

摘要

在信息学竞赛中，我们经常遇到这样一类问题：数据规模大，或是数据间的关系复杂，总而言之即输入数据的信息量过大。

作者在综合分析了很多信息学竞赛试题后，提出了“压缩法”这个概念：即压去输入数据中的冗余信息，保留下对解决问题有帮助的精华部分。压缩法在信息学竞赛中有着广泛的应用，但在各类问题中却有看起来截然不同的表现形式，本文即将选择多道有代表性的例题，提炼它们的共同点，提出压缩法适用问题的两个要点。最后，作者将在总结部分着重分析压缩法的工作特点，认为压缩法是在化归思想的基础上，加上了“信息化”的因素，通过合理的利用信息达到化简问题的目的。

关键字

信息学竞赛 压缩法

冗余/精华信息

可压缩性 替代法则 化归思想 信息化

目录

引子

可能不少同学都对题目中“压缩法”这个名词感到很陌生，不错，因为这是作者自己发明的一个名词☺。这篇论文就将带领大家走近我所说的“压缩法”，熟悉“压缩法”。

看到“压缩”二字，可能有的同学会想到我们常用的压缩软件，如 WinZip, WinRAR 等等，他们都是想尽办法的减小文件占用的空间来为我们服务。这正是压缩一词的本义，即“加以压力,以减小体积、大小、持续时间、密度和浓度等”[1]。

然而本文中要讨论的“压缩法”的含义则为：“除去冗余信息，留下精华，以减小问题的规模”，看得出，这正是为信息学竞赛量身定制一种好方法。

下面，就让我们看看压缩法在竞赛中的精彩表现吧！

压缩法的定义

一般来说，当我们处理问题时常常遇到一个集合 S，S 内部各个元素之间的关系对问题的结果不造成影响，而且也不会受到外部因素的干扰，那么我们可以将 S 看作一个内外隔绝的**包裹（package）**，忽略包裹内部的冗余信息，并将这个包裹与外部事物的联系保留，打包、**压缩后**作为一个新的元素存储下来以供以后分析处理。

这就是压缩法工作的基本流程，其好处在于略去了包裹内部种种纷扰却无用的信息，从而化简了问题，进而用更好的方法去解决问题。

压缩法的简单实例

其实我们对压缩法并不陌生，相信大家对下面两个例子都有一定的了解。

[例一]多源最短路问题（经典问题）

如下图。在一个加正权的有向图 $G = \{V, E\}$中，给出源的位置，求源到其余所有点的最短路长度。与一般最短路问题不同的是，本题中源是一个集合 S 中的所有点。而 S 到某一个点 p 的最短距离等于 S 中所有点与 p 最短距离的最小值。[2]

[1] 摘录自《高级汉语大词典》。

[2] 图中边上的数值为边的权值，顶点后括号内的数值为到该点最短路的长度。

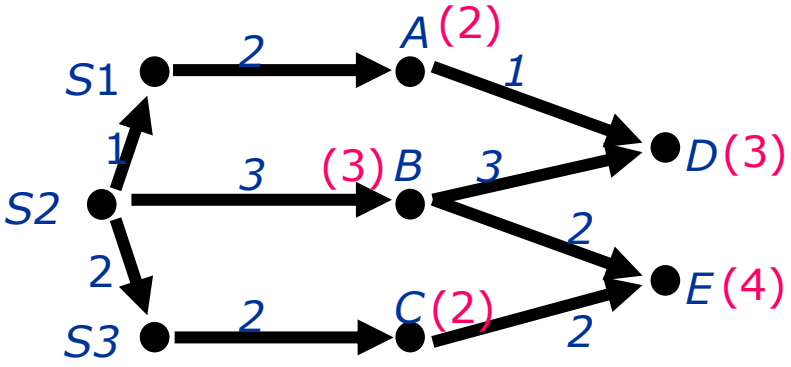

图 1

[分析]

这道题目的关键在于有多个源，而为了提高效率，只能执行一次 Dijkstra 算法。

其实这是很简单的，很多不同的方法都可以做到这一点。下面让我们看一看压缩法是怎么实现的：

可以看出，由于不可能有一条最短路径从一个源点出发却又经过另外一个源点，因此源点集合 S 中任何两点间的连边关系对答案都没有影响。如上文所述，可以将 S 视为一个内外隔绝的“包裹”，舍去包裹内的冗余信息，并将其“压缩”成为一个新的点 P_S。

另一方面，我们还需要保留 S 中节点对外的连边情况作为压缩后节点 P_S 的对外连边。

这样，我们就成功的完成了压缩流程，得到的是一张新图和一个单源最短路问题：这正是 Dijkstra 所做的。

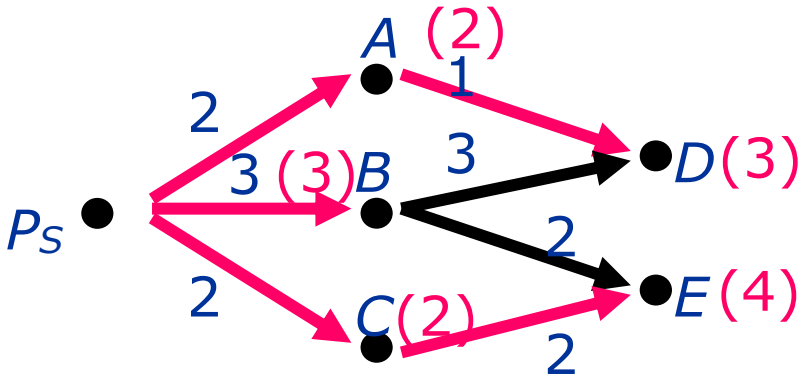

图 2

[例二]球队问题（经典问题）

给出某个篮球队的球员通讯图 $G = \{V, E\}$如下，若存在有向边(u, v)表示球员 u 可将消息及时告诉 v，若教练想将一条紧急消息告知给全体队员，利用这个通讯图，他至少要亲自通知几个队员？

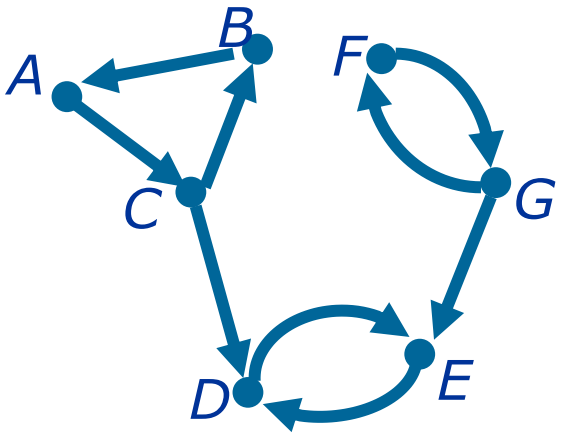

图 3

[分析]

看着这个杂乱无章的通讯图，我们决定还是使用压缩法“压去”一些不必要的关系，简化题设条件。

分析图中的每一个强连通分量 S，若 S 中的某一个球员得知了消息，那么显然 S 中其他所有的队员都可以及时获得消息。即 S 中每一个球员是否得知消息的状态都是相等的，如果有必要的话，教练最多只会通知 S 中的一个球员。此时保留 S 中的通讯边已经毫无意义，我们可以舍去它们，并将 S 压缩成为一个点 P_S，并保留 S 中每个球员的对外通讯情况作为 P_S 的连边情况。

经过对每一个强连通分量的压缩处理后，我们得到了一个全新通讯图如下。由于所有的强连通分量都被压缩成为一个点，新图中已经没有环的存在。

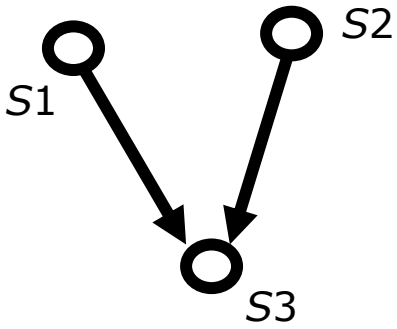

图 4

此时问题就变得很明朗了：没有入度的点（可能是一个强连通分量）是不可能被别的球员通知到的，因此教练必须亲自通知；另一方面，由于没有环的存在，每一个有入度的点，都至少可以被一个没有入度的点直接或者间接的通知到，因此教练没有必要亲自通知他们。

综上所述，教练需要通知的最少人数即为新图中没有入度的点的个数。问题得到解决。

可以看出，在上述两个例子中压缩法都有着出色的表现，而这两个例子又以图论中的“缩点法”为我们熟知。然而压缩法的应用并不仅仅局限于图论算法中，只要问题满足一定的条件，我们都可以向压缩法求助。

压缩法的要点

1. 可压缩性

可压缩性是压缩法可行性的理论保障。

如果可以将压缩法应用于一个问题，这个问题首先应当存在某一种或者多种可压缩性。可压缩性即是问题本身的某些元素集合在满足特定条件的情况下可以被压缩的性质，如上述两例中的源点集合和强连通分量。

一般来说，一个集合 S，若其内部各个元素之间的联系不会和外部元素相互作用而影响到问题的结果；则我们可以忽略 S 内的相互联系，从而将它们**压缩**为一个新的个体，即 ***S* 可压缩**。

在[例一]中，任意两个源点间的可到达关系对答案没有影响。鉴于此我们可以**舍去**源点集合 S 中两两节点间的**相互联系**，将其压缩，变成一个单一的源点考虑，**即 *S* 满足可压缩性**。

而在[例二]中，我们说过，每一个强连通分量中的球员是否得知消息的状态始终是相等的：无论这些球员相互的通讯情况如何，都不会改变这个事实。因此它们都是**可压缩的**，我们**舍去**分量内球员的**通讯情况**，将它们**压缩**成为一个个新的节点，考虑简化后的问题。

2. 替代法则

需要注意的是，压缩不是完全的忽略，可压缩性也不是可删除性。压缩法，顾名思义，“压”即为可以压去被压缩集合 S 内部各个元素的个体性质和它们的相互关系，这是冗余信息，对问题的结果并没有影响；而“缩”则是缩得精华，保留下集合 S 作为一个点与外部信息的联系。

如果说可压缩性体现了“压”的过程，那么“缩”则是替代法则的任务。

所谓替代法则，即用什么样的形式，表现出精华部分的内容。

在[例一]和[例二]中，都使用了最简单的替代法则：用一个新的点代替多个点的集合，并用相同形式的有向边代替原先每个点和集合外元素的连边。压缩前后的元素形式完全一致。而在一些复杂化的问题中，我们则需要认真考虑这个问题：压缩后保留下来了些什么，用什么形式保留？

不妨来看这样一个小例子。

[例三]模方程组的替代法则（经典问题）

我们在解一元模方程组的时候，也可以使用压缩法：通过将两两方程压缩减小问题的规模。方程压缩的可行性是毋庸置疑的，下面就让我们研究一下这里压缩的替代法则。

假设有两个方程需要压缩：

$$\begin{cases} a_1 x \equiv c_1 \pmod{b_1} \\ a_2 x \equiv c_2 \pmod{b_2} \end{cases}$$

而要求的替代法则就是用具有相同解集的单一方程来代替。

可将上面两个方程写作二元一次不定方程：

$$\begin{cases} a_1x + k_1b_1 = c_1 \\ a_2x + k_2b_2 = c_2 \end{cases}$$

将它们视为单独的方程，分别解出其中的一个解 x_1 和 x_2，则可将以上两个方程写作：

$$(*)\begin{cases} x = x_1 + p_1\Delta_1 \\ x = x_2 + p_2\Delta_2 \end{cases}$$

其中

$$\begin{cases} \Delta_1 = (a_1, b_1)b_1 \\ \Delta_2 = (a_2, b_2)b_2 \end{cases}$$

联立(*)式，仍是一个二元一次不定方程：

$$x_1 + p_1\Delta_1 = x_2 + p_2\Delta_2$$

可以得到 x 的一个解 x_0，那么就可以将 x 的通解写作：

$$x = x_0 + k[\Delta_1, \Delta_2]$$

即：

$$x \equiv x_0 (\mathrm{mod}\,[\Delta_1, \Delta_2])$$

至此，我们就用上面的一个模方程代替了压缩前的两个模方程：这就是本例中的替代法则。

至此我们已经初步熟悉了压缩法的两个要点，在很多竞赛问题中，只要我们能够找出这两个要点的具体体现，就等于找到了压缩法的着力点，进而可以用压缩法解题。

压缩法的应用

[例四]欧元兑换（BOI 2003）

你每天会收到一些欧元，可能为正数也可能是负数，银行允许你在某天将手头所有的欧元兑换成 lei。第 i 天的兑换比率是 1 euro ： i lei，同时你必须在多付出 T lei 被银行收取。在 N 天你必须兑换所有持有的欧元。要求找一个方案使第 N 天结束时能得到最多的 lei。

数据范围：

1 ≤ N ≤ 34 567

0 ≤ T ≤ 34 567

[分析]

首先让我们把这题的任务转化为较为简捷的数学语言。即给定一个费用 T 以及 N 个整数（**有可能是负数**）：

$$a_1, a_2, a_3, \ldots, a_N$$

要求一种划分方案：

$$1 \leqslant c_1 < c_2 < \ldots < c_k = N$$

即分为连续的 k 份，第 i 份的所有欧元都在第 c_i 天兑换，不妨将 c_i 称为**兑换点**。

从而使目标函数

$$Z = (\sum_{i=1}^{c_1} a_i) * c_1 - T + (\sum_{i=c_1+1}^{c_2} a_i) * c_2 - T + \ldots\ldots + (\sum_{i=c_{k-1}+1}^{c_k} a_i) * c_k - T$$

最大化。

[动态规划的矛盾]

看上去这是一道完全的动态规划问题，与压缩法没有什么关系，因为很显然我们都会构造出这样一个动态规划算法：

令 $f(n)$表示在 1~n 天的一个子问题：即以第 n 天($n \leqslant N$)为最后一个兑换点时的最大收益。则有边界条件

$$f(0) = 0$$

通过枚举上一个兑换点 $i<n$，得到动态规划方程

$$f(n) = \max_{0 \le i < n} \{ f(i) + (\sum_{j=i+1}^{n} a_j) * n - T \}$$

不妨设 S 为 a_i 数列的部分和，则可以简化上面的式子：

$$f(n) = \max_{0 \le i < n} \{ f(i) + (S_n - S_i) * n - T \}$$

至此，我们已经发现这个动态规划算法虽然时间复杂度是 $O(N^2)$的，并不能胜任题目的数据规模。通过一些分析，参考作者去年的论文[3]，我们知道了类似的动态规划方程式可以被优化的：如下图，在计算 $f(n)$时可将从前的每一个决策状态对应为平面上的一个个点

$$[\ S_i \quad , \quad f(i)\ ]$$

[3] 具体方法参见附录以及作者去年的集训队论文。

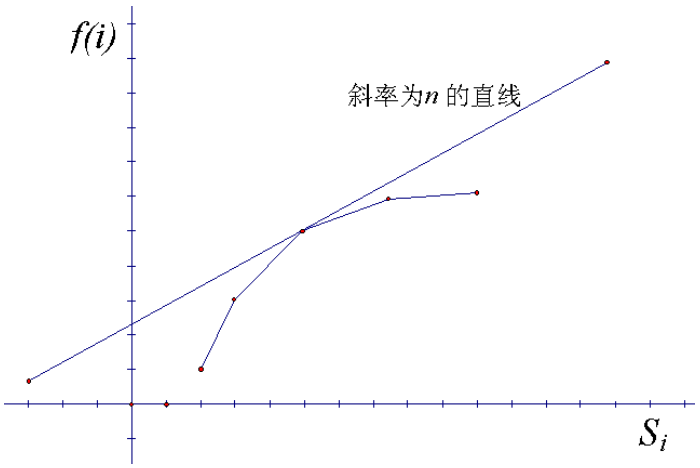

图 5

随着 n 的递增，依次将这些点加入一个下凸函数，并维护该函数，利用下凸折线斜率的单调性加速决策过程，从而可以得到一个 O(N)的方法。

然而新的问题又出现了：由于本题中的 a_i 可以是负数，也就是说部分和函数 S_i 并不是递增的，甚至不是非降的！那么在计算动态规划函数的过程中，随着 n 的递增，S_n 有可能变小，那么对应的点[S_n ， $f(n)$]将可能被加入到折线的“中间”，如下图所示，这样维护这个折线将变得异常困难！

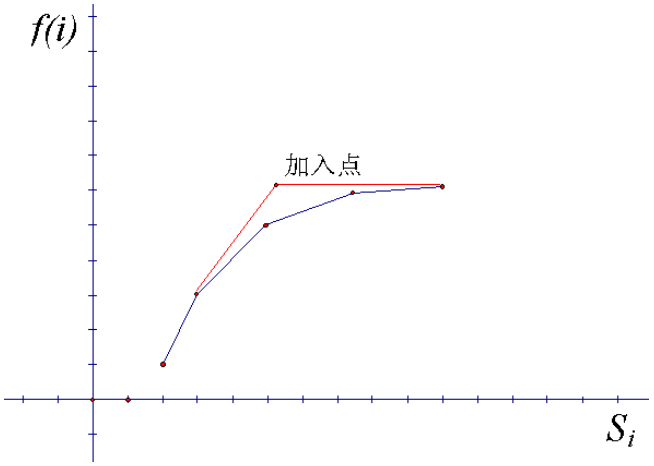

图 6

此时我们的分析就走到的本题矛盾的激化点：传统的斜率优化法要求操作函数 S_n 是单调的（至少非严格单调）；而题目中给出的条件却允许该函数任意的变化！

山重水复疑无路，两者直接的、完全的冲突似乎彻底否定了这个做法。而压缩法正是在此时大显身手，导演了一出柳暗花明又一春的好戏。

[压缩法化解矛盾]

让我们仔细的回顾一下题目，其中有这样一句话：“**第 i 天的兑换比率是 1 euro ： i lei**”，其含义则是，随着天数的递增，欧元将变得越来越值钱。那么不难想象出，如果当前手中的钱是正数，当然不急着将其兑换成 lei：过些日子将会兑换到更多；而如果当前手中的钱是负数，那么应该尽量早的将其卖出：

否则过些天将要赔出更多的 lei，但尚要注意，还应保证总兑换次数较少：由于每次兑换都要花 T lei。

以上感性的认识就勾画出了我们“压缩”算法的一个大致轮廓。不难通过理论证明，得到如下定理：

[定理 4.1]

若{a_i}序列中有一段连续的子序列 $a_i, a_{i+1}, \ldots, a_j$（$i<j$），满足该子序列的部分和

$$a_i,$$
$$a_i + a_{i+1},$$
$$a_i + a_{i+1} + a_{i+2},$$
$$\ldots$$
$$a_i + a_{i+1} + \ldots + a_{j-1}$$

均为正数，而总和

$$a_i + a_{i+1} + \ldots + a_j$$

非正数。

则当在第 i 天收获了 a_i 欧元后，一定有某个最优方案不急于兑换，而是继续在第$(i+1)$天收得 a_{i+1} 欧元，第$(i+2)$天收得 a_{i+2} 欧元，直到第 j 天收到 a_j 欧元，再看情况决定是否兑换。

从感性上看，这个定理显然是正确的。但还是让我们用理论证明一下：

[证明 4.1]

用反证法，考虑最简单的情况，假设某个最优方案在$[i, j]$区间中，仅有一个兑换点：在第 k 天（$i \leqslant k<j$）。[4]

那么设从上次兑换点到第$(i-1)$天为止，共余下了 x 欧元，如下图所示：

1. x 为正数[5]：那么显然到第 k 天的兑换点我们将余下更大的正数，取消这个兑换点，将其并入后一个兑换点一起兑换，既提高了手中欧元的价值，又省去了一次兑换手续费 T，何乐而不为？
2. x 为负数：将这个兑换点提前至第$(i-1)$天，提前脱手了负数的欧元，显然可以付出较少的 lei 的代价；而将$[i, k]$区间内的正数归入下一个兑换点，也可以得到更大的收益。

综上两种情况，无论如何我们都可以取消或者移走这个兑换点，却提高了收入，这与假设命题中方案的最有性不符，推出矛盾，从而[定理 4.1]得证。

[定理 4.1]即说明了本题中存在的可压缩性：它实际上告诉我们，如果 a_1 是正数，则第一天结束时一定不急于兑换，等到第二天有 a_2 的收入，若当前手中的欧元$(a_1 + a_2)$仍是正数则还是不兑换，再等到第三天……直到第 k 天结束若尚未兑换的欧元

$$a_1 + a_2 + \ldots + a_p$$

不再是正数，再考虑是否兑换。这么多天的收入在兑换问题上是等价的，它们作为一个个个体的性质是相同的，因为根据定理他们一定都在同一个兑换点被兑换。即可以把这一段数字**压缩成一个点**：令

[4] 多次兑换的同理可以证明。

[5] 或者可以是 0。

$$b_1 = a_1 + a_2 + \ldots + a_p$$

表示我们认为这 p 天的欧元是在同一个时间点 b_1 收到的，可以作为一个压缩后的点与其它每个收入点**同样的**被处理：稍有不同的是，这些钱的兑换要在 $t_1 = p$ 天后进行，而不是原来的每天后。所以我们的替代法则为：用二元组(b_1, t_1)代替原来 p 天的收入。

这样，我们继续考察 a_{p+1} 开始的每个点，累计从该点开始收入的欧元，直到新的点 a_q 出现了非正数的和，那么就将 a_{p+1} 直至 a_q 这些(q-p+1)个点压缩成为一个新的收入点：

$$b_2 = a_{p+1} + a_{p+2} + \ldots + a_q$$

同时记录下 $t_2 = q$，表示压缩处理后第二个兑换点的具体时间。用(b_2, t_2)代替。

这样一次处理下去，我们可以得到以下 M 个压缩后的收入点和兑换点：

$$(b_1, t_1), (b_2, t_2), (b_3, t_3), \ldots, (b_M, t_M)$$

值得强调的是，最后处理到 a_N 时可能会有手中欧元为正数的情况，将这些钱在第 N 天（最后一天）兑换即可获得它们的最大价值。

将目光转移到压缩后的任务上，和原任务类似，我们还是要求对数列 b_i 一个最优分割兑换方案，构造它们的部分和 S'并写出新的动态规划方程：

$$g(m) = \max_{0 \le i < m} \{ g(i) + (S'_m - S'_i) * t_m - T \}$$

可以看出由于任意 $b_i \leqslant 0$，部分和 S'_m 一定是非上升序列：优化决策过程的条件“S'_m 的非严格单调性”已经满足！

那么利用附录所述的方法，可以构造出 O(M)的动态规划算法，再加上之前的 O(N)的压缩预处理。我们得到了一个 O(N + M) = O(N)的线性算法！

至此，通过压缩处理，本题已经圆满解决。

[小结]

回顾一下[例四]的分析过程，我们已经掌握了一些基本的动态规划优化方法，如本题中的“斜率优化法”。解决类似的问题本应不难，然而题设中一个障碍条件“每次收入的欧元 a_i 正负任意”却与斜率优化的原理即操作函数的单调性产生了极大的冲突。看似矛盾不可调和，然而利用压缩法，将连续的部分和是正数的收入子序列“压缩”，直到成为一个不大于零的收入点，奇迹般的化解了矛盾，打开了僵局，为下一步优化营造了条件。压缩法在本题中证明了每一个待压缩收入集合中的收入点在兑换问题上的等价性，继而可以压缩，忽略它们作为个体的性质，从而达到简化问题的目的。

[例五]合并数列问题（ZOJ p1794 Merging Sequence Problem 改编）

给出 K 个数列，假设每个数列的长度分别为 n_1，n_2，n_3，…，n_K。而这 K 个数列分别为：

$$a_{1,1}, a_{1,2}, a_{1,3}, \cdots, a_{1,n_1}$$
$$a_{2,1}, a_{2,2}, a_{2,3}, \cdots, a_{2,n_2}$$
$$\vdots$$
$$a_{K,1}, a_{K,2}, a_{K,3}, \cdots, a_{K,n_K}$$

注意这里每个数都可能是负数。

将这 K 个数列**归并**为一个新的数列 S，所谓归并，即选择一个非空的输入数列，将其第一项放入 S 的尾部，并在数列中删去该项；重复这一过程，直到所有的输入数列都是空数列。

要求寻找一个归并后的数列 S，新的数列 S 的长度为 $N=n_1 + n_2 + n_3 + \ldots + n_K$。且 S 的各个部分和

$$Sum_0, Sum_1, Sum_2, \ldots, Sum_N$$

中，最大的最小。

数据范围：

$K \leqslant N \leqslant 10^5$。

[分析]

初看这题似乎只有 $O(K*N^K)$的动态规划算法，但这个算法太不现实了，下面就让我们有意识的尝试用压缩法来分析这题。

[观察压缩要点]

首先我们看一下这题中的可压缩性具体表现为什么：

假设某一个数列中存在一段连续的子串 u，

$$u_1, u_2, u_3, \ldots, u_p$$

如果可以证明当归并过程中选择了 u 的第一项放入 S，就一定可以连续的选至 u 的最后一项而不会丢失寻找最优解的可能性：即 u 一定会在某个最优串 S 中连续出现。根据可压缩性的定义，如果 u 满足上述性质，那么这一段子串内部各项的互相联系即它们的相互位置关系已经失去价值，也就是说，u 内各项的相互联系不会与外部因素相互作用而影响最优结果，即 u 可压缩。

再分析一下本题中的替代法则，分析如何保留这一段子列对其他数值的影响。

首先，如下图，u 子串中的某一个部分和可能成为 S 序列里最大的部分和：不过也只有 u 的最大部分和即它在数列中的**相对峰值**有这个可能性。

第二，只要 u 子串不是在 S 数列的末尾，其总和一定会对以后的部分和产生影响，间接的影响到部分和的最大值：这也是我们所关心的。

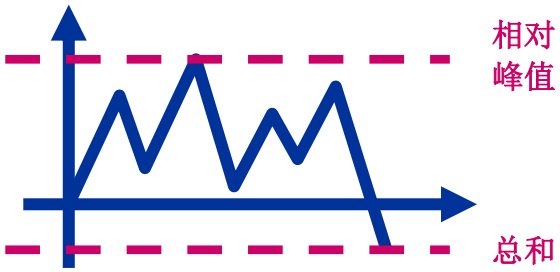

图 7

综合以上两点，我们连续的两个数的子串 *a, b* 陈述一个可以被压缩的子串 *u* 的所有令人感兴趣的特征，其中 *a* 为非负数，表示 *u* 的相对峰值，*b* 是一个非正的修正值，使(*a* + *b*)等于子列 *u* 的总和。从图象上看，即将上图简化为下图的形式。这就是本题中压缩的替代法则。

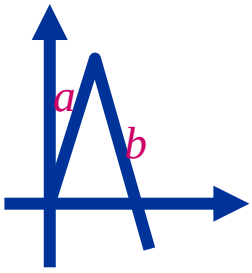

图 8

注意形如上文中含两个数字的子串 *a, b*，其中 *b*≤0 显然是可压缩的：即他们在最优串 *S* 中一定连续出现，不妨称之为一“对”（**couple**）。

这样的一“对”不仅可以在替代法则中代替一段待压缩的子串，在以后的定理证明中也可以代替一些未知的数列的基本特征。我们称一“对” *a, b* 为正，当它对以后部分和的影响(*a*+*b*)为正数，否则若影响为负数则称之为负；特别的，定义 *a, b* 为“零对”当且仅当 *a*+b=0。

下面我们就来分析在这题中存在着哪些可压缩性：在什么情况下，某一个数列中的一段子串可以被压缩。

[寻找可压缩性：第一阶段压缩]

首先不难发现下面的定理：

[定理 5. 1]

如果在某个输入数列中，存在连续的子串

$$u_1，u_2，u_3，\ldots，u_p$$

设其部分和为

$$Q_1，Q_2，Q_3，\ldots，Q_p$$

且 $Q_1，Q_2，Q_3，\ldots，Q_{p-1}$ 均是非负数，只有 $Q_p < 0$。则当可以选择 u_1 加入 *S* 数列时，要么暂时不选，要么将连续的 u_1 直至 u_p 一起加入 *S*。**即 *c* 子串可压缩**。

要证明这个定理，先来看一个引理：

[引理 5. 1. 1]

如果在某个输入数列中，存在连续的两“对”：

$$a_1, b_1, a_2, b_2$$

其中 a_1, b_1 非负，a_2, b_2 为负，如下图所示。则可以将它们压缩成新的一“对”。

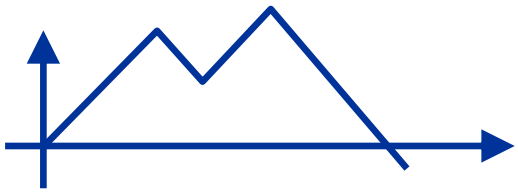

图 9

[证明 5.1.1]

使用调整法，假设在 S 中，有如下三“对”形成的子列：

$$a_1, b_1, x, y, a_2, b_2$$

其中 x, y 表示隔开 a_1, b_1 和 a_2, b_2 的一段数。

分两种情况讨论：

1. 若 $x+y\leqslant 0$，则将子列调整为：

$$x, y, a_1, b_1, a_2, b_2$$

此时 a_1, b_1 和 a_2, b_2 连续。而由于“对” a_1, b_1 非负即 $a_1+b_1\geqslant 0$，调整后 x 的绝对高度不会升高；由于 $x+y\leqslant 0$，调整后 a_1 的绝对高度也不会变得更高。

2. 若 $x+y>0$，则将子列调整为：

$$a_1, b_1, a_2, b_2, x, y$$

此时 a_1，b_1 和 a_2，b_2 连续。而由于 $a_2+b_2<0$，调整后 x 的绝对高度降低；由于 $x+y>0$，调整后 a_2 的绝对高度同样也会降低。

综上两种情况，无论如何都可以将 S 中的 a_1, b_1 和 a_2, b_2 调整到连续的位置而 S 的峰值却不会增高。因此[引理 2.1.1]成立。

利用这个引理，就可以证明定理的结论了。

[证明 5.1]

首先认为这 p 个数字是 p 个特殊的“对”，即一个数字被“压缩”后形成的“对”。那么这 p 个数字构成的数列就变成了 p 个“对”形成的数列。

根据假设，只要这 p 个“对”还没有被最终压缩成一个，那么第一个“对”总是正的，而由于 p 个数字的总和是负数，因此总能找到一个负的“对”。鉴于此，只要数列还没有被压缩成的一个“对”，我们总可以找到一个正的“对”后接一个负“对”，并根据[引理 5.1.1]将这两个”对”压缩成为一个新的“对”；重复着一个过程，直到只剩一个“对”时，就构造证明了这个定理。

现在我们可以这样依次处理每一个输入数列：从第一个数开始，累计部分和，依次向后考察每一个数，若加入部分和后部分和依然为正，则继续处理，否则若部分和为负，可将目前处理的这一段数根据[定理 5.1]“压缩”成一个“对”。将部分和清零后继续处理。

至此，我们已经完成了第一阶段的压缩。在这一阶段完成后，每一个输入的数列的前半部分都变成了许多负的“对” a_i, b_i，即每一个正数后都跟有一个绝对值更大的负数。而后半部分则是这样的一列数：它的任意部分和都是正数。

可以证明，在最优的合并数列 S 中，所有输入数列的“后半部分”将恰好是

S 的后半部分，即最优合并方案一定是将所有输入数列中的前半部分按某个顺序都归入 S 后再考虑那些任意部分和都是正数的子列[6]。

因此这个问题可以分成两个互不影响的部分：先归并所有数列的前半部分，再考虑那些部分和为正的子列。进一步的分析可以发现，问题的后一部分其实和前一部分等价，S 数列的总和确定，若将所有部分和为正的子列逆向观察，则一定也可以被压缩成多个负的“对”从而进行合并。所以我们只要专心得考虑问题的第一个部分就可以了。

[寻找可压缩性：第二阶段压缩]

[定理 5.2]

如果在某个输入数列中，存在连续的负的“对”

$$a_1, b_1, a_2, b_2$$

且峰值在 a_1，即 $a_1 \geqslant a_1 + b_1 + a_2$，那么这两个“对”在 S 中必定连续出现，即**可压缩**。

这个定理的证明也很简单：

[证明 5.2]

假设某一个最优方案的一个子列为

$$a_1, b_1, x, y, a_2, b_2$$

则这一段的峰值为 $\max\{a_1, a_1 + b_1 + x, a_1 + b_1 + x + y + a_2\}$，而做调整

$$a_1, b_1, a_2, b_2, x, y$$

后的峰值为

$$\max\{a_1, a_1 + b_1 + a_2, a_1 + b_1 + a_2 + b_2 + x\}$$

由于 $a_2 + b_2 \leqslant 0$，得到

$$\begin{cases} a_1 \leq a_1 \\ a_1 + b_1 + a_2 \leq a_1 \\ a_1 + b_1 + a_2 + b_2 + x \leq a_1 + b_1 + x \end{cases}$$

➔ $\max\{a_1, a_1 + b_1 + a_2, a_1 + b_1 + a_2 + b_2 + x\} \leqslant \max\{a_1, a_1 + b_1 + x, a_1 + b_1 + x + y + a_2\}$

即调整后 a_1, b_1 和 a_2, b_2 连续出现，且峰值不会变得更大：同样是一个最优方案。命题得证。

[推论 5.2.1]

如果在某个输入数列中，存在连续的多个负的“对”

$$a_1, b_1, a_2, b_2, a_3, b_3\ldots$$

且峰值在 a_1，即 $a_1 \geqslant a_1+b_1+a_2, a_1+b_1+a_2+b_2+a_3\ldots$，那么这个子串可以压缩。

以[推论 5.2.1]为依据，我们对所有具有第二条可压缩性的子串实施压缩，将得到新的 K 个数列。每个新数列中的数字符号一定是正负交错，而绝对值是递增的，因此每一个“对”的相对峰值都是递增的。如下图所示。

[6] 证明较为简单，故略去。

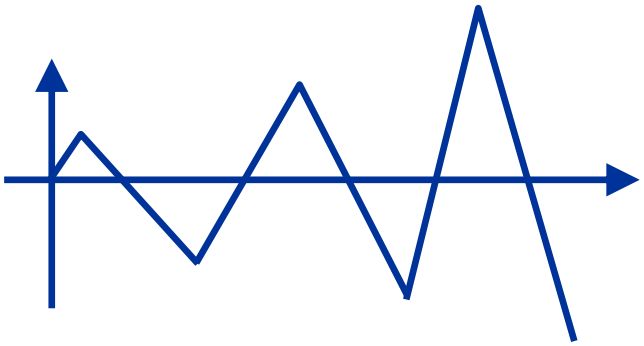

图 10

[贪心法解题]

经过上述第二阶段的压缩处理，现在这个问题就以一个十分简洁、优美的形式呈现在我们面前了：有 K 个数列，且每个数列都是由若干个总和为负的“对”组成，且它们的相对峰值单调递增，要求将它们归并成为新数列 S，且数列 S 的最大部分和尽量小。

至此，我们可以使用一个简单的贪心算法了。由于每一个“对”的总和为负，因此将相对峰值较高的“对”尽量向后放，可以尽量降低其绝对峰值，这是贪心的原则。而由于每个数列中“对”的相对峰值严格递增，因此对这些数列执行一次按相对峰值高度大小为序的归并排序，就可以得到一个新数列 S，其中“对”的相对峰值也是递增的：这正好完全符合我们的贪心原则。从感性上看这个算法是正确的，而简单的推导也可以证明我们的观点，我们把这个问题就留给各位同学了。

算法实现上，在每次归并时，都选择供选“对”中相对峰值最小的一个加入 S，而通过一个堆，则可以做到在 $O(\log_2 K)$的时间内取最小。

因此压缩法不仅为本题找到了多项式级别的算法，而且还是一个时间复杂度仅为 $O(N\log_2 K)$的优秀算法。[7]

[小结]

在上例中，在第一阶段的压缩之后，我们手中的 K 个数列由杂乱无章变得稍有规律，其每一个数列中的正数后总会跟有一个绝对值更大的负数；但这还不足以实施贪心法，因此又有了第二阶段的压缩，进一步提炼真正有用的信息，得到的新数列中的非结尾的负数后又跟着一个绝对值更大的正数。这样数列中一个个“对”的相对峰值呈现美妙的单调性，这正是贪心法需要的条件。

可见，在很多问题中由于输入数据过于纷乱，我们可以采取多阶段压缩、甚至是迭代式压缩的方法，分层次的丢弃冗余，提炼精华，就像一个金字塔，一层一层的逼近，最终到达解决问题的顶峰。

[7] 至于如何实现，大家可以参考论文附件中关于本题的程序。

总结

让我们先来回顾一下文中的主要例题。在[例一]单源最短路问题中，我们使用压缩法将源点集合压缩成为一个新的源，从而由多源问题得到了单源的问题进而得以解决；在[例二]球队问题中，压缩法将每个强连同分量缩成一个个点，从而将复杂的、任意的球员通讯联系图，简化成为一个有向无环图；在[例五]合并数列问题中，根据两条不同的可压缩性，符合条件的子串都可以被压缩成 a, b 两个数形成的一个“单峰”，由此 K 个任意的输入数列在压缩转化后显示出了美妙的单调性从而有了后来的贪心法解题。

在这些例题中，压缩法都是将一个个复杂的事物用较为简单的来代替，从而化简了问题。这正是数学思维中的一条经典思想：**化归思想**。

说到化归思想，大家一定对我们在做数学题时使用的换元法不陌生。例如解下面一个方程

$$x+\sqrt{x-1}=7$$

的时候，可以设 $t=\sqrt{x-1}$ 代换得到关于 t 的方程

$$t^2+1+t=7$$

从而得到原方程的解。这里的假设 $t=\sqrt{x-1}$ 就是做了一步化归：化无理为有理，化抽象为形象，进而化复杂为简单为最终解决问题铺平道路。可以看出，我们上文中讨论的压缩法也有类似的神奇特点。

然而，换元法做的仅仅是形式上的变化，是一个完全等价的变化：针对上面的例子，其换元的前提就是 t 和 $\sqrt{x-1}$ 相等。

而压缩法则不然，不妨回顾一下压缩法的两个要点：可压缩性和替代法则。可压缩性实际上承认了一个集合中冗余信息的存在性，即集合中元素的相互关系是无意义的。而替代法则则为我们提供了一个实际操作方案，通过一个现实的方法，保留下信息的精华部分。

可以看出，压缩法的两个要点都在围绕着信息的利用大做文章，而这是换元法所不具备的特性。

我们生活在一个信息爆炸的社会中，快速准确的获取信息已不是什么难事，然而如何防止信息污染却是一个日益严重的全球化问题。信息学竞赛应运而生，其宗旨在于教会我们如何合理有效的利用信息。而信息学竞赛中的压缩法继承了经典数学中的化归思想，又创新的加入了“信息化”因素，将两种思想有机的结合，通过舍去冗余的信息达到化繁为简、化难为易的目的，在很多复杂化的问题中有着广阔的应用前景。

因此，当我们遇到问题，感觉题设条件过多，信息量过大，无从下手时，不妨试一试压缩法，因为我们压缩法的目标正是：

压去冗余 缩得精华

附录

附录一：关于[例四]中的斜率优化法

考虑[例四]中的动态规划方程：

$$f(n) = \max_{0\le i<n}\{f(i)+(S_n-S_i)*n-T\}$$

当 n 确定时，不妨设

$$g(i) = f(i)+(S_n-S_i)*n-T$$

表示每一个决策，则

$$f(n) = \max_{0\le i<n}\{g(i)\}$$

对于两个不同的决策 $0\leqslant i, j<n$，设 $S_i > S_j$，那么 $g(i) > g(j)$的充分必要条件则为：

$$f(i)+(S_n-S_i)*n-T > f(j)+(S_n-S_j)*n-T$$

$$f(i) - f(j) > n(S_i - S_j)$$

$$\frac{f(i)-f(j)}{S_i-S_j} > n$$

可以看出，如果将每个决策看作一个对应的点

$$P_i[S_i,\quad f(i)]$$

如正文中图 11 所示。

则两个决策 P_i 和 P_j 相比，若 P_i 对应的点在 P_j 的右边，那么 P_i 优于 P_j 当且仅当直线 P_iP_j 的斜率大于 n。参考作者去年论文中的[例二]最大平均值问题的优化方法，可以证明在待选的决策点中，下凸点（凹点）的存在是没有意义的：无论 n 的取值如何都不会使其变成最优的决策点。因此我们只需要维护一个上凸函数即可。

如果可以确定每次添加的点都在原先所有点的一侧（这正是正文中分析所要做的），则可以参见作者去年论文，对这个上凸折线进行平摊复杂度为 O(1) 的插入维护和取最优操作。

附录二：论文附件

Baltic Olympiad in Informatics 2003 Euro（欧元兑换）英文原题


Euro. doc

Baltic Olympiad in Informatics 2003 Euro（欧元兑换）程序，时间复杂度 O(*N*)：

euro. pas

[例五]合并数列问题，程序，时间复杂度 O($N\log_2 K$)：

MergeSequence. pas

附录三：关于 MergeSequence. pas 程序的输入格式

输入文件 MergeSequence.in 的第一行包含一个整数 K，表示待归并数列的个数，以下 K 行，每行描述一个数列。每行的第一个数 n_i 表示该数列中的数字数目，之后 n_i 个数，顺序描述这个数列。

参考文献

《信息学奥林匹克竞赛指导——图论的算法与程序设计(PASCAL 版)》 吴文虎 王建德 编著

《实用算法的分析与程序设计》 吴文虎 王建德 编著

《浅谈数形结合思想在信息学竞赛中的应用》 作者论文于 2004 年信息学奥林匹克竞赛冬令营

Zhejiang University Online Judge：http://acm.zju.edu.cn