浅谈必要条件的应用

广东北江中学 方奇

【关键字】 命题 必要条件 效率 精确

【摘要】

必要条件是命题之间的一种逻辑关系。在解题过程中若能利用好必要条件，将有助于我们提高算法效率。针对必要条件，本文先阐述了其定义，然后结合其特点集中分析了它在解各类问题中的应用，指出利用必要条件关键在于“减少冗余，体现本质”。最后总结出一些寻找必要条件的经验。

【正文】

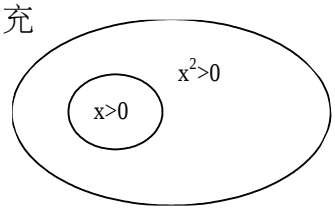
在现实生活中，许多问题错综复杂，体现在信息学问题上，就会导致我们难以从纷繁的条件关系中建立有效的数学模型进行求解。有时，准确地应用必要条件，有助于我们揭示问题的本质或简化原有模型，从而找到高效的解决方法。

下面，我们先来看看什么是必要条件。

一 必要条件的定义

数学上，判断真假的语句叫做命题。常用小写拉丁字母 p q r s……来表示。如果由命题 p 经过推理可以得出命题 q，也就是说，“如果 p 成立，那么 q 成立”，则记为 p=>q

一般地，如果已知 p=>q，那么我们说，p 是 q 的充分条件，q 是 p 的必要条件。例如，“$x^2>0$”是“x>0”的必要条件。



在这里，“$x^2>0$”的解集包含了“x>0”的解集。$\{x \mid x>0\} \in \{x \mid x^2>0\}$。用文氏图表示:

二 必要条件的应用

在信息学问题中，必要条件有着广泛的应用，主要体现在下面两个方面。

§1 缩小求解范围

大部分问题所要求解的往往是符合某一些条件的方案或具体数值。将这些条件命题记为 p，其解集用 A 表示。回想我们的解题思路，当然不能一下就知道 A 的确切解，而总是通过已知条件 q 在较大的既定范围内 B 运用各种方法，如枚举、搜索、动态规划、贪心等等从中进行筛选，最终找到符合 p 的解集 A。

在这一过程中,B 包括了 A，A 必也符合条件 q,有 p=>q，q 是 p 的必要条件。我们从比较“宽松”的条件 q 入手，在 B 中寻找 A，使其满足比较“苛刻”的条件 p。可以看到，开始界定的必要条件 q 的“宽松”程度，即 B 的范围大小直接影响着解决问题的效率。当 q 越逼近于 p，B 越趋近于 A 时，混合物中真金的含量比例越大，那我们要清除的沙子就会越少，效率自然也就越高了。可是，这必须建立在可行的基础上。因为，如果条件 q 在判断处理上难于实现或是要花费大量时间，那么无论 q 再怎么精确，都是毫无意义的了。

于是，这就启发我们要在坚持“容易实现”这一原则下，尽力寻找“精确”的必要条件, 以缩小求解范围，提高出解速度。

[例 1]

问题描述

已知二分图 G(V,E)，问哪些边是构成完备匹配所必须的。共 2n 个顶点，n≤100。

问题分析

设必须边的集合为 A。当且仅当某边 e 被去掉后，子图 g 不存在完备匹配，则有 e∈A。这可算是条件 p。思考到这里，可以得到算法①

枚举图 G 中所有边 E

- 对于某边 e,在原图中去掉，g=G-e
- 对 g 求最大匹配
- 若不存在完备匹配，则表示 e∈A，否则反之。

用匈牙利算法求最大匹配，时间复杂度是 $O(n^2)$，枚举所有边需 $O(n^2)$。于是此算时间复杂度为 $O(n^4)$，过于庞大。

在上述算法中，我们按照常规思路，一开始就不经意地把初始范围 B 设成全题边集 E,必要条件 q 为“满足是 E 中的边即可”。这样，便把求解的范围定大了。

其实，若边 e∈A,那么 e 必属于任意一个完备匹配中，这是显然的。换句话说，就是任意一个完备匹配都包含了所有必须边集 A。于是，我们可以将算法①改进如下:算法②

先求出 G 中的一个完备匹配，设为 B

枚举 B 中所有边 E

- 对于某边 e,在原图中去掉，g=G-e
- 对 g 求最大匹配
- 若不存在完备匹配，则表示 e∈A，否则反之。

在算法②中，实际上是把初始范围 B 缩小为一个完备匹配，必要条件 q 精确为“边属于完备匹配 B”。这么一来，时间复杂度下降为 $O(n^3)$

从[例 1]中我们可以看到利用好必要条件，对于解这类存在性问题有很大帮助。其实，应用必要条件缩小求解范围，其巨大威力还是体现在搜索算法上。

一般，搜索算法如深度优先、广度优先这类不带启发性的盲目搜索，时间复杂度往往是指数级的，根本无法忍受。于是搜索过程中如何限制其范围，减少搜索量变成了关键。在这里，使用必要条件剪去一些没有希望的搜索分支，便是俗称的“剪枝”。

[例 2]《移棋子》(Puzzle) GDOI98

问题描述

试题见附录

问题分析

这是一道典型的搜索题。根据题设，可知:

⑴ 在有限步内把原始状态转移至目标状态，或确定无解

⑵ 移动步数已知

⑶ 各个移动方向步数已知，但序列不知

⑷ 解必须满足⑵和⑶限制

采用深度优先搜索，逐一产生各种可能的移动序列，输出满足题设的即可。

显然，上述全搜索在时间上令人无法忍受，我们可从以下几方面来剪枝:

1 各移动方向的数目

由于各移动方向的数目是已知的，因此一旦某个移动方向的个数超出限制，则可不搜索此方向。

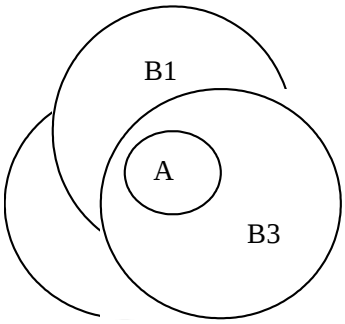
2 空格的位置和各方向的数目

由于目前状态和目标状态都已知，各方向的数目也已知，因此可确定目前状态的空格能否由此方向集合移至目标状态位置。即必须满足

（目标状态空格横坐标-当前空格横坐标=4 方向所剩数目-3 方向所剩数目）and （目标状态空格纵坐标-当前空格纵坐标=2 方向所剩数目-1 方向所剩数目）

3 各字符的位置和各方向数目

同 2，我们可以根据目前状态和目标状态以及各方向的数目，确定目前状态的各字符能否由此方向集合移至目标状态位置。



在上述思考过程中，我们将 p 定义为“操作序列能使初态达到终态”，满足 p 的所有操作序列集合设为 A,q1、q2、q3 代表 3 项剪枝，分别满足 3 项剪枝的操作序列集合设为 B1、B2、B3。有 p=>q1 p=>q2 p=>q3 A∈B1∩B2∩B3。用文氏图表示

这里，共运用了 3 个必要条件。从图中可以看出，三管齐下的作用下，求解范围缩得更小更精确。当然，要注意的事，这 3 项剪枝都容易实现，且没有耗费太多的时间。（不然就的得不偿失了）通过这些优化，搜索量大大减少，程序速度有了很大提高。

必要条件在动态规划方面也有应用。

[例 3]《Musketeers》 POI 9901

问题描述

n 个人围成一圈，按顺序编上号 1 至 n。决斗在相邻两人间进行，胜者继续留在圈中，负者被淘汰出圈外。现不知道决斗进行的顺序，只知道共进行了 n-1 场决斗。给出 n 个人之间的胜负关系，问哪些人可能成为最后的胜者 。N≤100。

问题分析

这题与 NOI95《石子合并》一题有点类似。仿照《石子合并》一题，可得到一种运用动态规划的思路。

E[i,j]记录 i、j 之间的胜负情况。当 i 胜 j 时 e[I,j]=true;当 j 胜 I 时 e[i,j]=false。

A[i,j,k]记录 k 是否能在 i 至 j 这些连续的人之间取胜（k 在其中），肯定时为 true，反之为 false。

状态转移方程为

$$A[i,j,k]=\begin{cases}\text{true} & \text{存在}(a[i,x,y]\text{ and }a[x+1,j,k]\text{ and }e[k,y])\text{ or }(a[u,j,v]\text{ and }a[i,u-1,k]\text{ and }e[k,v])=\text{true} \\ & i\le y\le x<k \qquad k<u\le v\le j \\ \text{false} & \end{cases}$$

此算法时间复杂度达到 $O(n^5)$，空间复杂度为 $O(n^3)$，太大了，必须改进。

注意到编号为 x 的人能从所有人中胜出，必要条件是他能与自己“相遇”（把环看成链，x 点拆成两个，中间的人全部被淘汰出局，x 保持不败）。这样，在连续几个人的链中，只须考虑头尾两个人能否胜利会师，中间的则不予考虑，从而少了一维状态表示量。设 meet[i,j]记录 i 和 j 能否相遇，能则为 true，否则为 false。状态转移方程为

$$meet[i,j]=\begin{cases}\text{true} & \text{存在}meet[i,k]\text{ and }meet[k,j]\text{ and }(e[i,k]\text{ or }e[j,k])=\text{true} \\ & i<k<j \\ \text{false} & \end{cases}$$

这样，时间复杂度下降为 $O(n^3)$，空间复杂度下降为 $O(n^2)$。

在此题中，由于前后两次考虑胜负的必要条件不同，导致了两次动态规划的途径也不一样。显然，后一次由于必要条件更为精确，减少了冗余运算，算法无论时空上都有了飞跃。

以上几道例题，分析了利用必要条件缩小求解范围这一特点在枚举、搜索、动态规划三类题型中的应用。可以看到，由于必要条件选择的精确，前后变化是显著的。

如果说，“缩小求解范围”这一特点，还只是在原有算法上加以优化改进，那么下面我们将看到，精确选择必要条件还能揭示问题本质，引导我们设计出新的更准确体现问题内在联系的更高效的算法。

§2 揭示题目本质

[例 4]《科学实验》(Experiment) GDOI2001

问题描述

试题见附录。

问题分析

若把 n1、n2 块陨石碎片抽象成点，不考虑其他，那么本题就是一个求二分图最佳匹配的问题。可以用匈牙利算法或是费用流来解决。时间复杂度为 $O(n1^2n2^2)$,当 n1、n2 取最大值 500 时，必定承受不了。

其实，将本题单纯地看成是匹配问题，还只是注意到了矛盾的普遍性，并没有发现矛盾的特殊性。细心思考，可以发现，该二分图的边上的权并不是随机安排的，而是等于关联该边地两点的重量之差的绝对值。这是本题模型与一般二分图最大的区别。

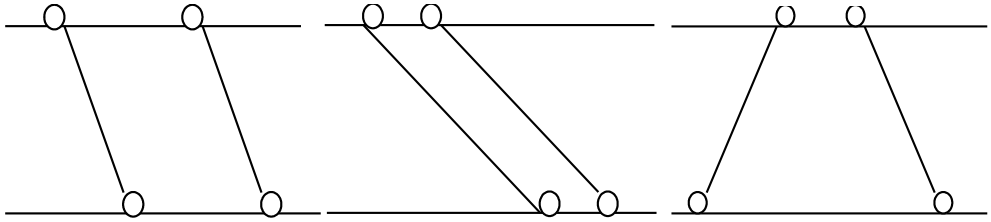


图 1　　图 2　　图 3

这一特点意味着什么呢？经过画坐标轴，将 A 星和 B 星的陨石重量分别排序后发现，满足最佳匹配的一个必要条件是“构成匹配的边不相交”（图 2 相交与不相交两种情况的效果一样，于是规定只选择不相交的方案），也就是说，设某一次实验从 A 星上取得重量为 a1 的碎片，从 B 星上取得重量为 b1 的碎片；另一次实验取得 a2、b2 两块碎片。若要使实验效果最佳，必不存在 a1<a2 b1>b2 或 a1>a2 b1<b2 这两种情况。

有了这个条件，问题便同时符合了无后效性和最优化原理，可以用动态规划思想求解，具体如下：

先将两组数分别按从小到大排序，设 n1≤n2（若 n1>n2,则将两组数交换）

数组 A[i,j]表示第一组前 I 块碎片和第二组前 j 块碎片实验的最佳效果

状态转移方程为

$$A[i,j]=\begin{cases}\min\{A[i,j-1],A[i-1,j-1]+|w1[i]-w2[j]|\} & j>i\\ A[i-1,j-1]+|w1[i]-w2[j]| & j=i\end{cases}$$

其中，w1 记录第一组碎片的重量；w2 记录第二组碎片的重量。

新算法排序耗时 $O(n1\log n1+n2\log n2)$,动态规划耗时 $O(n1n2)$,所以总体来说时间复杂度为 $O(n1n2)$。这正是必要条件切中题目要害的结果。

三 寻找必要条件的方法

通过上面几道典型例题的分析思考，我总结出以下几点寻找必要条件的经验方法。

㈠ 从特性中归纳共性

正如[例 1]中 B 集合的确定，这充分体现了“矛盾的普遍性寓于特殊性之中，并通过特殊性表现出来”这一哲学原理。其实这和我们思维习惯是一致的。当我们碰到一道难题时，总是尝试从最简单的特殊情况入手，逐渐深入，最终分析归纳出一般规律。

㈡ 从多方面寻找必要条件

有些时候，特别是在搜索算法终，单一的剪枝其力度是远远不够的。当内在因素相互影响难以总体把握的时候，我们可以尝试从多个侧面分析寻找必要条件，把问题肢解，最后在将各方面的必要条件综合起来使用。这样往往能起到一种合力的效果。[例 2]《移棋子》便是一个典型。

㈢ 与原有模型比较，分析异同

本质往往在比较中显现。像[例 4]《科学实验》一题，正是在比较中发现了新模型与原有二分图模型的根本区别——“边不相交”这一必要条件。从而引导我们运用动态规划思想进行解答。

方法是灵活多样的。我们在寻找必要条件时应该坚持“具体问题具体分析”的原则，不拘一格，灵活处理。

四 结语

必要条件实际上是逻辑推导的一种理论依据，也是思考过程中的一种方法体现。它在许多方面都有着广泛的应用。解题时，若能寻找出精确的必要条件，一方面能帮助我们揭示问题本质，一方面又能“缩小求解范围”，大大提高算法效率。

如何寻找精确的必要条件，其本身并没有固定的方法准则，这就要求我们在分析问题时，要勤于思考，善于发现，在实践中总结提高。

【附录】

1 《移棋子》题目

有一个 5×5 的方格棋盘，棋盘上放着 24 粒不同的棋子，分别用英文大写字母 A、B、……、X 来表示：棋盘上还有一个方格空着，用空格表示。

游戏的每一步是将空格上方、下方、左方或是右方的棋子移入空格，这四种操作分别用 1、2、3、4 来表示。

如果给出棋盘的初始状态和一定顺序的有限操作序列，就可以得到唯一的目标状态，例如：下图中的初状态经过操作序列“144223”到达终态。

初态 终态

T	R	G	S	J
X	D	O	K	I
M		V	L	N
W	P	A	B	E
U	Q	H	C	F

T	R	G	S	J
X	O	K	L	I
M	D	V	B	N
W	P		A	E
U	Q	H	C	F

但是，原来正确的操作序列顺序给人打乱了，初态按照被打乱操作序列并不能得到终态（仅仅是顺序打乱了，各类型操作总数不变）。

现在的任务是求出原来的操作序列。（如果有多解，给出其中任意一种即可）

输入格式：

从当前目录下的文本文件“PUZZLE.DAT”中读入数据。该文件的第一行到第五行是棋盘的初态，每行有五个字符。第六行到第十行是棋盘的终态，每行有五个字符。第十一行是一个正整数，表示操作序列长度 L（L≤50）。第十二行有 L 个字符，表示被打乱后的操作序列。每行个数据之间无空格。每行首尾无多余空格。

输出格式：

输出到当前目录下文件“PUZZLE.OUT”。若问题有解，则输出一行 L 个字符，表示原来正确的操作序列；若无解，则输出“0”

输入输出举例：

输出文件：PUZZLE.DAT

```
TRGSJ
XDOKI
M VLN
WPABE
UQHCF
TRGSJ
XOKLI
MDVBN
WP AE
UQHCF
6
442231
```

输出文件：PUZZLE.OUT

```
144223
```

2 《Musketeers》 POI VI Stage 1 Problem 1

In the time of Louis XIII and his powerful minister cardinal Richelieu in the Full Barrel Inn *n* musketeers had consumed their meal and were drinking wine. Wine had not run short and therefore the musketeers were eager to quarrel, a drunken brawl broke out, in which each musketeer insulted all the others.

A duel was inevitable. But who should fight who and in what order? They decided (for the first time since the brawl they had done something together) that they would stay in a circle and draw lots in order. A drawn musketeer fought against his neighbor to the right. A looser "quit the game" and to be more precise his corpse was taken away by servants. The next musketeer who stood beside the looser became the neighbor of a winner.

After years, when historians read memories of the winner they realized that a final result depended in a crucial extent on the order of duels. They noticed that a fence practice had indicated, who against who could win a duel. It appeared that (in mathematical language) the relation "**A** wins **B**" was not transitive! It could happen that the musketeer **A** fought better than **B**, **B** better than **C** and **C** better than **A**. Of course, among three of them the first duel influenced the final result. If **A** and **B** fight as the first, **C** wins eventually. But if **B** and **C** fight as the first, **A** wins finally. Historians fascinated by their discovery decided to verify which musketeers could survive. The fate of France and the whole civilized Europe indeed depended on that!

Task

N persons with consecutive numbers from *1* to *n* stay in a circle. They fight *n-1* duels. In the first round one of these persons (e.g. with the number *i*) fights against its neighbor to the right, i.e. against the person numbered *i+1* (or, if *i=n*, against the person numbered *1*). A looser quits the game, and the circle is tighten so that the next person in order becomes a winner's neighbor. We are given the table with possible duels results, in the form of a matrix. If $\mathbf{A}i,j = 1$ then the person with the number *i* always wins with the person *j*. If $\mathbf{A}i,j = 0$ the person *i* looses with *j*. We can say that the person *k* may win the game if there exists such a series of *n-1* drawings, that *k* wins the final duel.
Write a program which:

- reads matrix **A** from the text file MUS.IN,
- computes numbers of persons, who may win the game,
- writes them into the text file MUS.OUT.

Input

In the first line of the text file MUS.IN integer *n* which satisfies the inequality *3<=n<=100* is written. In each of the following *n* lines

appears one word consisting of *n* digits 0 or 1. A digit on *j*-th position in *i*-th line denote **A***i,j*. Of course **A***i,j* = 1 - **A***j,i*, for *i*<>*j*. We assume that **A***i,i* = 1, for each *i*.

Output

In the first line of the output file MUS.OUT there should be written *m* - the number of persons, who may win the game. In the following *m* lines numbers of these persons should be written in ascending order, one number in each line.

Sample Input

```
7
1111101
0101100
0111111
0001101
0000101
1101111
0100001
```

Sample Output

```
3
1
3
6
```

Remark

The order of duels: 1-2, 1-3, 5-6, 7-1, 4-6, 6-1 gives a final victory to the person numbered 6. You can also check that only two persons more (1 and 3) may win the game.

3 《科学实验》题目

最近，我国科学家通过星球探测器分别从 A 星和 B 星上取得了 n1、n2 块结构相类似的陨石碎片，需要做 n=min(n1,n2)次化学实验通过对比来研究它们，每次仁取 A,B 星上各一颗碎片进行化学实验，由于在实验中经过化学变化，碎片在实验后不可再用。另外，由于此种陨石碎片的特性受重量的影响很大，所以为了使实验效果达到最佳，科学家们希望能选出一种方案，使得每次实验的两颗陨石重量之差的绝对值的总和最小。

现在，请你求出该最小总和为多少。

输入格式：

输入文件为当前目录的 Exp.dat；

第一行为两个整数：n1 n2

接下来的 n1 行，每行一个实数，表示 A 星上各颗陨石的重量。

然后有 n2 行，每行一个史书，表示 B 星上各颗陨石的重量。

注意：n1,n2≤500;每颗陨石碎片的重量不超过 90。

输出格式：

输出文件为当前目录的 Exp.out;

只有一个实数，表示所求得的最小总和。

样例输入	样例输出
4 2 44.9 50.0 77.2 86.4 59.8 58.9	23.8