探索构造法解题模式

【关键字】构造法　数学模型

【摘要】

本文通过一些实例探讨构造法在信息学竞赛解题中的应用，首先阐述了数学方法在解题中的巧妙应用，引进了数学建模的思想。较详细地讨论建立模型的方法，包括直接构造问题解答的模型，图论模型，网络流模型以及组合数学模型。介绍了构建模型的基本方法和基本思路。同时也分析了数学模型的类型和作用。

【正文】

引言

“构造法”解题，就是构造数学模型解决问题。信息学竞赛中，它的应用十分广泛。构造恰当的模型或方法，能使问题的解决，变得非常简洁巧妙。

就我们现在所能接触的问题而言，构造的数学模型，从数学方法的分类来看，它是属初等模型、优化模型这两种。

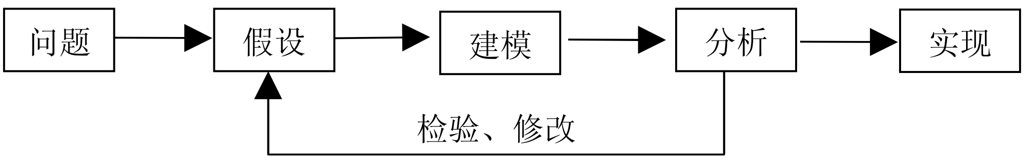
一般地，数学模型具有三大功能：

1．解释功能：就是用数学模型说明事物发生的原因；

2．判断功能：用数学模型判断原来的知识，认识的可靠性。

3．预见功能：利用数学模型的知识、规律和未来的发展，为人们的行为提供指导或参考。

构造法解题的思路或步骤可以归纳为：



本文的目的，在于利用构造数学模型的思想，构建我们对问题的解法。

数学的巧妙应用

数学是研究现实世界数量关系和空间形式的科学，数学的特点不仅在于概念的抽象性、逻辑的严密性、结论的明确性，而且在于它应用的广泛性。我们讲数学方法是指把错综复杂的问题简化、抽象为合理的数学结构的方法。

我们以具体的问题为例析，解释这些观点的应用，通过这些问题展示了数学的奇妙作用，让我们体会利用数学方法来解决问题时的一种乐趣。

〖问题 1〗跳棋问题

设有一个 n×n 方格的棋盘，布满棋子。跳棋规则如下：

1. 每枚棋子跳动时，其相邻方格（有公共边的方格）必须有一枚棋子为垫子，才能起跳；
2. 棋子只能沿水平或垂直方向跳动；
3. 棋子跳过垫子进入同一方向的空格，并把垫子取出棋盘。

把 n×n 方阵棋盘扩展成 m×m，试求出最小的 m，使得棋子能依规则跳动，直到棋盘内只剩下一枚棋子，并给出一种跳棋方案。

本题若用盲目搜索法解决，对 n=4,5 或许能行，但也要很高的费用。试用构造法求解。下面我们把 n 按除以 3 的余数进行分类讨论：

A. n=3k（k∈N）

我们把 n×n 的棋盘分成 k×k 个 3×3 的网格，并重复排列这种 3×3 的网格，延伸至整个平面。 每个 3×3 的网格按下列方式进行着色：

1	2	3
2	3	1
3	1	2

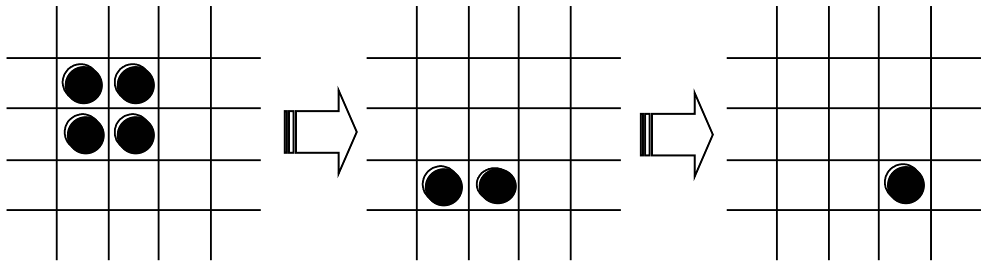
每次跳棋都是从 3 色中的两色跳至另一色。设每次跳棋跳至 1、2、3 色的次数分别为 a、b、c 次。不妨设最终剩下的一枚棋子是 1 色。可得：

$$\begin{cases} 3k^2 + a - b - c = 1 & ① \\ 3k^2 - a + b - c = 0 & ② \\ 3k^2 - a - b + c = 0 & ③ \end{cases}$$

① － ②，得 2(a+b)=1，由于 a,b 都是整数，本式不成立。所以对于 n=3k 的情况，不可能最终只剩下一枚棋子。

B. n=3k+2（k∈N）

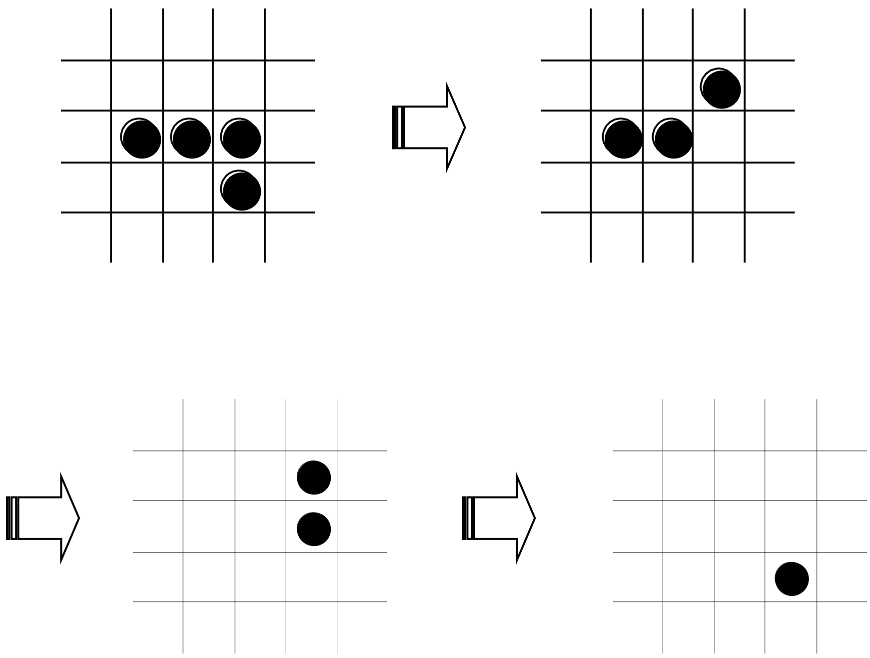
我们先从较小的 n 开始，研究是否有规律可循。同时为了对较大的 n 能得到



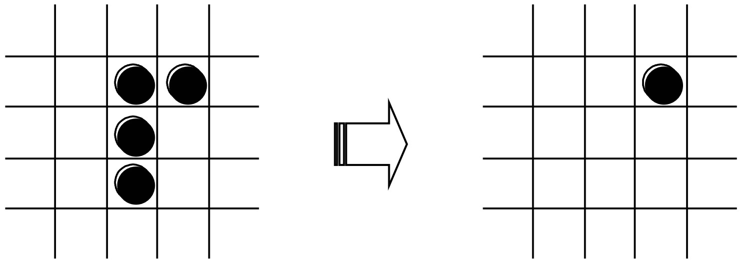
解法的规律，我们构造几种基本形状的跳法。当 n=2 时，解法如下：

我们把它定义为基本形状 A。

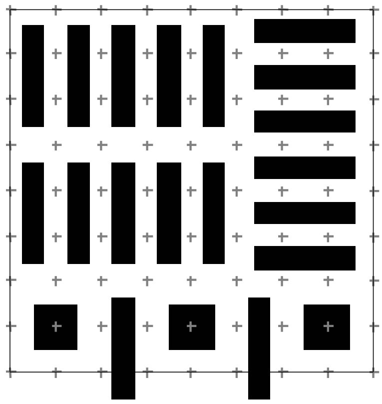
(1) 基本形状 B



(2) 基本形状 C

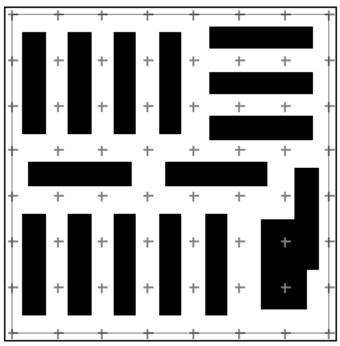


以上两种基本跳法告诉我们：对于任意的连续 3 个棋子，只要另有一个棋子辅助，就可以连续地消去 3 个棋子。有了这几种基本跳法，就可以开始构造本题的解答了。我们以 n=8 为例来说明。如下图所示，其中的正方形表示基本形状跳法 A，竖形长条表示基本形状跳法 C，横形长条表示基本形状跳法 B。其跳法是：首先从左到右消去左上边的竖形长条，接着从上到下消去右上边的横形长条，最后从左到右顺序消去下边的正方形与竖形长条。



C. n=3k+1（k∈N）

类似于情况 B，我们仍用基本跳法来构造解答。以 n=7 为例，从上到下，从左到右，逐个消去两种基本形状，最后剩下右下角的一个刀把形，再独立完成。



观察上述过程可以发现，只需把原棋盘向四周扩展 1 行，变成$(n+2)(n+2)$即可。

上题是构造法运用的一个典型例子。构造法不像搜索、动态规划等，有固定的模式可套用，而完全需要依据实际情况进行状态分析、构图分析对问题进行抽象性处理，这些，通常需要有良好的数学功底和创造性思维能力，严谨的科学精神。

〖问题 2〗圆桌吃饭问题

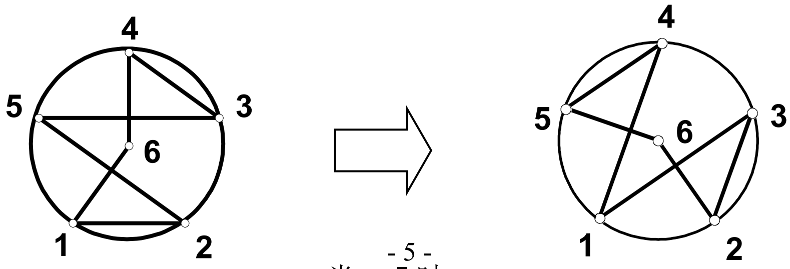
n 个人围着一张圆桌吃饭，每个人都不愿意两天与同一人为邻，问最多能坐多少天，并给出一种排列方案？

为了清楚的理解问题的实质，我们以图的模型来描述它：设 $G=(V,E)$为一完全图，$|V|=n$。图中的每个顶点代表一个人，连结顶点的边表示人之间的相邻关系。因此，每种围绕圆桌的吃饭方案就成为图中的一条哈密尔顿回路。设 $L=\langle v_1,v_2,\ldots,v_n\rangle$为 G 中的一条哈密尔顿回路，其中所含的边的集合记为 $e(L)$。试求 m 与 $L_1,L_2,\ldots,L_m$，使得 $e(L_i)\cap e(L_j)=\phi$，并且 m 达到最大值。

为了求得 m 的最大值，我们可以先估算一下 m 上界。完全图 G 共有 $n(n-1)/2$ 条边，每条哈密尔顿回路含有 n 条边，可见 m 的上界是$(n-1)/2$。当 n 为奇数时，$m=(n-1)/2$；当 n 为偶数时，$m=n/2-1$。那么这个上界是不是精确上界呢？如能构造出 m 条哈密尔顿回路，也就证明了它是本题的答案。

现给出构造方法：作一圆，把圆周分成 n-1 等分，标上 n-1 个刻度，将顶点 1 至 n-1 依次排列在圆周上，顶点 n 放在圆心。先从圆心出发，向任意点连一条线，再从这点出发，沿圆周向左右两个方向迂回连线，直到连完圆周上所有的点，再连回圆心。这样就构造出一条哈密尔顿回路。保持所有的顶点位置不变，把所有连线围绕圆心逆时针方向旋转一个刻度，得到一条新的哈密尔顿回路。这样连续旋转$(n-1)\text{div }2$ 次，就得到了$(n-1)\text{div }2$ 条回路。

下面来证明此算法的正确性。这只要证明所有的边旋转时都不重叠即可。观察下图，可以把所有边分为两大类，圆的半径和弦，弦又可以按它的长度分为



当 n=5 时

(n-1)div 2 类。显而易见，不同类别的边在旋转时是不会重叠的。那么相同类别的边会不会重叠呢？其实每类边至多只有两条，且又处在圆中相对的位置上，所以当所有的边都旋转半周以后，是不会重叠的。

其算法描述如下：

```
Procedure Table;
k:=0; I:=1; J:=n-1;
repeat
   inc(k);
   if odd(k)
      then  r[k]:=I;
               inc(I);
      else  r[k]:=J:
               dec(J);
until I>J;
for l:=1 to (n-1) div 2 do
   for i:=1 to k do
      write((d[i]-1+l)mod(n-1)+1,' ');
   writeln(n);
```

〖问题 3〗千足虫问题

千足虫"ishongololo"，身长，色黑而亮，是一种多脚的节肢动物。考虑一个长为 K 宽为 W，高为 H 的各面相互垂直的固体（K,W,H≤32），它从(1,1,1)开始爬进长方体果实，并会吃光它所经过的路上的一切果实。有下列限定条件：

1．千足虫严格地占据 1 个空的小立方块 block。

2．千足虫每次吃完一个小立方块 block。

3．千足虫不能进入以前自己进入过的小立方块 block。

4．千足虫不能进入未吃过的小立方块 block，也不能爬到果实之外。

5．千足虫只能吃掉或只能爬入相邻的 block，此外该 block 还必须没有别的面暴露于已被吃光的 block。

请编程求一个千足虫的动作序列，使它尽可能多地吃掉小立方块。

Toxic 解题思路：

本题是IOI’97的一道试题，作为国际竞赛中的难题，本题有相当的技巧。首先表现在这是一道三维空间题，需要强的空间想象能力；其次，这题状态较为复杂，数量巨大。显然，搜索只能作为对小数据作探索性实验时使用的算法。原题是依据解答的近似程度评分，这又降低了解题的难度。

我们先讨论一种简单的情况，H=1。

平面的Toxic路线构造

问题要求在三维长方体中寻找路径。简单起见，我们从平面的Toxic问题出发（当MaxZ = 1时）。怎样才能在一个平面矩形中吃到尽可能多的立方体呢？由于走过的路径不能第二次经过，所以要避免路线过早地将图形分为两部分，

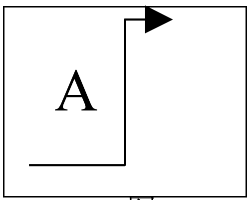


图 x1

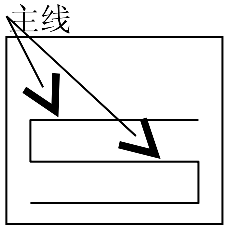
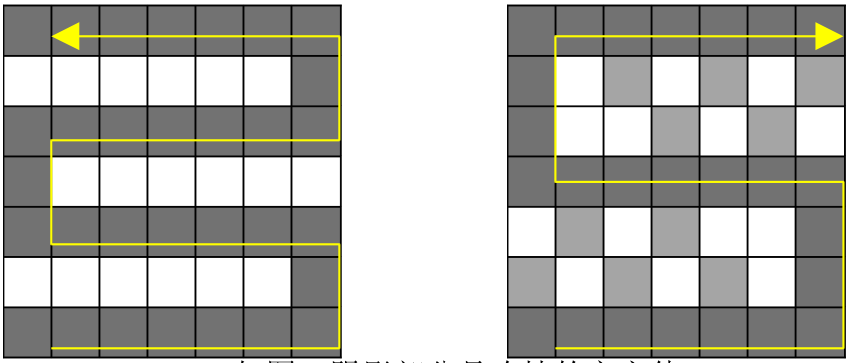


图 x2

如图x1的路线就是一个“失败”的例子，区域A再也无法到达。所以我们考虑基本路线采用“蜿蜒”前进的策略，如图x2。

基本路线确定以后，还需要确定两条“主线”之间的间隙。显然，主线之间隔开1~2行较好，超过2行就会有整行的空行无法吃到。图x3-a与x3-b是主线间隔分别为1、2行时构造出的解。间隔1行时主线较为密集，但是沿途不能吃其他的立方体；间隔2行虽然主线只占总行数的1/3，但是沿途可以“间隔”地吃立方体（图x3-b的浅阴影）。比较发现，间隔2行能吃到大约2/3的立方体，间隔1行只能吃到大约1/2的立方体，所以我们采用间隔2行的策略。



a 如图，阴影部分是吃掉的立方体b

图 x3

通过和搜索程序的结果比较，在平面上构造出的解和最优解相当接近。

立体的 Toxic 路线构造

有了平面的解，我们再以平面构造的解为基础，构造立体的千足虫路线。

平面构造的解是否可以直接套用在空间长方体上呢？如果千足虫在每一层都能象图 x3-b 一样移动，那么解应当是相当优的。遗憾的是千足虫在某一层爬过以后，会使得相邻的层有一些立方体无法吃到。准确地说，当立方体(x, y, z)被吃掉以后，如果千足虫继续再 Z 轴坐标为 z 的平面内移动，那么(x, y, z+1)以及(x, y, z-1)这两个立方体就都吃不到了。可以看出，吃掉方格影响的相邻层，这种影响不会“隔层”，也就是说(x, y, z)被吃掉不会影响到 Z 轴坐标为 z+i(|i|>1)的层面。这就启发我们，可以让千足虫在奇数层上以图 x3-b 的方式移动，当在奇数层从一个顶点移动到另一个顶点时，向上走（吃）2 个立方体。暂时不考虑偶数层的吃法，这样各层的移动互不干扰，至少是一个可行解。我们把这种吃法定为立体 Toxic 问题解的基本路线。

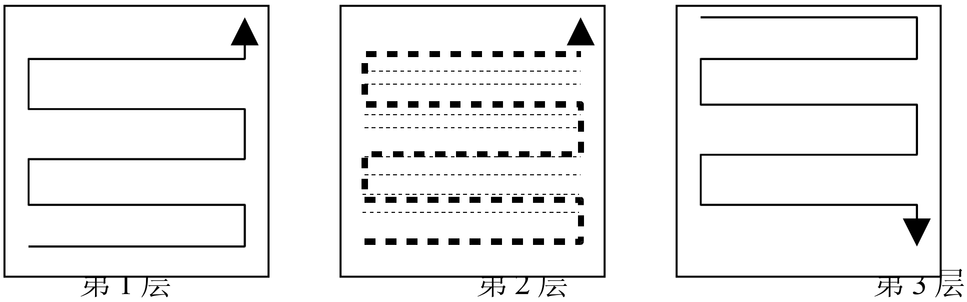


图 x4

如果仅仅在奇数层上移动、吃立方体，那么总共能吃到的立方体应当是全部的 1/3 左右。我们考虑再吃一些立方体。基本路线在每个偶数层只停留（吃掉）1 格，偶数层大部分的立方体没有被吃掉。我们以第 1 层为例，考虑偶数层的一般吃法。由于第 1 层吃掉一些立方体，第 1 层相应地有若干方格无法吃到。如图 x4，第 2 层的粗虚线部分在第 3 层是无法吃到的，细虚线部分有大约 1/2 的部分是无法吃到的。第 2 层的立方体只能是千足虫在 1、3 两层的主线上移动时吃。由于到了第 3 层已经不会因为吃第 2 层的立方体而影响主线的顺利延伸（如果从第 1 层吃第 2 层的果实则不然），所以在第 3 层应当将第 2 层的立方体尽可能多第吃。这样，就可以在第 2 层 1/3 的行（对应于第 3 层的主线）间隔地吃到 1/2 的果实。如果所有的奇数层都向下尽可能多地吃，那么可以多吃到所有立方体大约 1/12 的立方体，现在一共可以吃到大约 1/3 + 1/12 = 5/12 的立方体了。

沿途“拾遗”

在奇数层我们之所以不往向上 1 层吃，是担心影响向上 2 层主线的延伸。当所有路线确定以后，我们可以在不影响路线的前提下由奇数层往向上 1 层吃（在基本路线确定以前则不能随便这样吃）。这样，又可以吃到大约总数 1/12 的立方体（略少于 1/12）。实际上，由于基本路线的确定，具体编程时可以在不影响主线和遵守吃立方体规则的情况下尽可能多地吃立方体。

至此，构造基本完成。我们的方法大约可以吃到所有立方体中的 1/2。

坐标变换

我们确定的策略以平面为基础。当 MaxX=MaxY=MaxZ 时，下底面的选择无关紧要，而当三者不同时，下底面的选择影响到最终吃的立方体数。所以我们通过坐标变换将 3 个不同的底面作为下底面，这样就可以在尽量不增加编程复杂度的情况下得到更优的解。经过测试，下底面的选择对解的优劣影响在最大情况下可以达到 0~15%。

本题的编程实现有一定的难度，具体情况详见附录。

几种模式的构建

以上我们讨论了构造法解题的一种类型，即直接构造问题解答，这只是构造法运用的一种简单类型。它只能针对问题本身，探索其独有性质，不具备可推广性。如果要进一步探索，得到各种问题的共性，就要把构造法的思想上升到数学建模的高度。构造法作为一种解题方法，更多地体现的是数学建模的思想。对于在这部分中讨论的数学模型，我们按照其主要作用分为两大类：

1．认识事物的模型

这类模型是对现实世界的一个抽象。它提取了其中的有效信息，用简明的方式表达出来。它可以是一条代数公式、一幅几何图形，也可以是一个物理原理、化学方程式，甚至是一种自然现象。只要是对客观事物起到抽象概括作用，有利于我们开拓思维的，都可以算是这类模型。

2．指导算法的模型

本类模型通常是经过前人的大量研究，具有丰富、和谐的性质和定理的一门理论。在信息学竞赛中，较常用的有图论模型、组合数学模型、流网络模型等。这类模型有各式经典算法可供套用，对我们算法的设计起决定作用。

下面举一些实例来说明各种模型的建立和具体应用。

〖问题 4〗01 串问题

给定 N、L_0、A_0、B_0、L_1、A_1、B_1，设计一个长度为 N 的 01 串，使得对于任何连续的长度为 L_0 的子串，0 的个数大等于 A_0 且小等于 B_0，对于任何连续的长度为 L_1 的子串，1 的个数大等于 A_1 且小等于 B_1。

本题是 NOI’99 中的一道试题，考试时许多同学都是采用深度搜索算法，加上限界截枝，5 个数据可在规定时间内解出 4 个正确答案。现在反思本题，是否存在高效快捷的多项式时间算法呢？首先我们用数学中多元方程与不等式的思想构造本题的第一个模型。

模型 4.1

设所求的 01 串为 S[1..n]，S[i] ∈{‘0’,’1’}，X[i](I∈{0..n})表示 S 的前 I 个字符中’1’的个数。要使该数据模型 X[0..n]成为原题的解答，还需满足下列条件：

A．它是一个 01 串

$$\begin{cases} X[0] = 0 \\ X[0] \le X[1] \le X[0]+1 \\ X[1] \le X[2] \le X[1]+1 \\ \vdots \\ X[n-1] \le X[n] \le X[n-1]+1 \end{cases}$$

B．它满足子串 01 个数限制

S 的所有长度为 L 的子串中’1’的个数为 X[i+L]-X[i](0≤i≤n-L)。因此有

$$\begin{cases} A_0 \le X[L_0] - X[0] \le B_0 \\ A_0 \le X[L_0+1] - X[1] \le B_0 \\ \vdots \\ A_0 \le X[n] - X[n- L_0] \le B_0 \end{cases}$$

$$\begin{cases} L_1\text{-}B_1 \le X[L_1] - X[0] \le L_1\text{-}B_0 \\ L_1\text{-}B_1 \le X[L_1+1] - X[1] \le L_1\text{-}B_0 \\ \vdots \\ L_1\text{-}B_1 \le X[n] - X[n- L_1] \le L_1\text{-}B_0 \end{cases}$$

求以上不等式组的一组整数解。若无整数解，则原问题也是无解。

模型 4.1 是属于上面所说的第一类模型，它用简明扼要的形式化语言重述了原题。让我们来分析一下这个模型。仔细观察上述条件，发现它有以下特点：

a．除 X[0]=0 外，其余的条件都是由“≤”联接的不等式；

b．每个不等式都是含有两个未知数，一个常数的一次不等式；

c．可以通过移项，把每个不等式所含的两个未知数分别移到不等号的两边，并使它们的系数都等 1。

可见，所有的不等式都可以整理成这样的形式

$X[i] + C \le X[j]$ （i、j =0..n）

它给我们以很大的启发，但我们仍然不能从这个模型直接得到高效算法。

上述不等式类似于连结两点的一条有向边。因此，我们联想到信息学解题中常用的图论知识。

模型 4.2

首先，构造一个有 n+1 个顶点的有向图 G，把其顶点编号为 0..n。若有不等式 X[i] + C≤X[j]，则添加一条从点 i 指向 j 的有向边，其权为 C。以下我们分根据图 G 的性质分两种情况进行讨论。

(1) 图 G 中不含正圈

这时，图的顶点间存在着最长路径。令 D[i]表示从顶点 0 到顶点 i 的最长路径长度。对于图中每条从点 i 指向点 j 的权为 C[i,j]有向边，有性质 D[i]+C[i,j]≤D[j]。这可用反证法证明：若有 D[i]+C[i,j]＞D[j]，即从点 0 至点 i 再到点 j 的一条路径长度大于到点 j 的最长路径长度，这与 D[j]的定义矛盾。显然，D[i]=0。这样，让 X[i]=D[i]，X 完全符合所有的限制条件，即为原不等式组的一组解。

(2) 图 G 中含有正圈

设一个正圈为 v1→v2→v3→…→vm→v1，其路径权和为 C＞0。有

$$\begin{cases} X[v1] + C1 \le X[v2] \\ X[v2] + C2 \le X[v3] \\ \\ X[vm] + Cm \le X[v1] \end{cases}$$

相加，得 $X[1] + C \le X[1]$

$$C \le 0$$

这与上述假设矛盾。所以此时原不等式组无解。

对于上述第一种情况，可采用动态规划解决。事实上，由于最长路径长度最大为 n，只要在 n 次迭代内未得到最短路径，就可以判定图中含有正圈，也就是上述第二种情况。算法如下：

```
procedure 01_string;
由不等式构造有 n+1 个顶点的有向图 G;
X[0..n]      0;
K    0;
Repeat
   Inc(k);
   modify    false;
   for I    0 to n do
```

```
      for J    0 to n do
          if x[i]+c[i,j]≤x[j] then
              modify    true;
              x[j]    x[i]+c[i,j];
Until not modify or (k＞n);
If k＞n
    then Return(无解)
    else 依据 X[0..n]构造 01 串;
```

事实上，本题的 G 边数很少，连结每个顶点的至多只有 4 条边。所以 G 在存储结构上，不必使用邻接矩阵。可采用下列定义

G : array [1..n,1..4] of integer;

上述算法中对图的有关操作（如枚举连结顶点的每条边），只要略加修改即可。

综观本题解题的解题过程，包括了两次构造模型。第一次，是在原题的基础上构造不等式模型；第二次，是在不等式模型的基础上构造图论模型。本题构造的实质也就是不断把原问题转化为等价的新问题，直到产生出易于求解的问题为止。

〖问题 5〗整数拆分问题

给定整数 N，求把 N 拆分成 k 个小于 n（n＞N/2）的非负整数的方法总数。

该问题实际上是求不定方程 X1+X2+…+Xk=N（其中 0≤Xi≤n-1）的解的个数。由于只要求解的总数，可考虑构造母函数的的方法。

解状态可以转化为：求多项式 $S=(x^0+x^1+\ldots+x^{n-1})^k$ 中 x^N 的系数 A_N。

设多项式 $G=x^0+x^1+\ldots+x^{n-1}$，G=（$1-x^n$）/(1-x)，所以 $S=G^k$。由级数

$$(1+x)^m=\sum_{i=0}^{\infty}\frac{m(m-1)\cdots(m-i+1)}{i!}x^i$$

得：

$$(1-x)^{-k}=\sum_{i=0}^{\infty}\frac{-k(-k-1)\cdots(-k-i+1)}{i!}(-x)^i=\sum_{i=0}^{\infty}\frac{k(k+1)\cdots(k+i-1)}{i!}x^i=\sum_{i=0}^{\infty}C_{k+i-1}^{i}x^i$$

$$\therefore s=(1-x^n)^k(1-x)^{-k}$$
$$=(1-kx^n+C_k^2x^{2n}+\cdots+(-1)^kx^{nk})(1+kx+C_{k+1}^2x^2+\cdots)$$

设：$(1+kx+C_{k+1}^{2}x^{2}+\cdots)$ 的每项系数为：b_I

$(1-kx^{n}+C_{k}^{2}x^{2n}+\cdots+(-x)^{nk})$ 的每项系数为：c_I

∵N<2n

∴x^N 系数为：$\begin{cases} b_0 \bullet c_N + b_1 \bullet c_{N-n} = c_{k+N-1}^{N} - kc_{k+N-n-1}^{N-n} (N \ge n) \\ C_{k+N-1}^{N} (N < n) \end{cases}$

由于 n 的值可能比较大，所以 N 的值也可能比较大，求解中需要进行高精度计算。

本题利用母函数法，得到问题的数学模型，再通过数学方法求解，进而只要进行高精度求组合数就可得到问题的解。

模型与算法

无论是直接构造问题解答，还是构造数学模型，其算法都是我们最关心的问题。如何设计一个有较低编程复杂度和时空复杂度且结构清晰的算法，是非常重要的。以下列几点出发来考虑：

1．选择的模型必须尽量多体现问题的本质特征。但这并不意味着模型越复杂越好，累赘的信息会影响算法的效率。

2．模型的建立不是一个一蹴而就的过程，而要经过反复地检验、修改，在实践中不断完善。

3．数学模型通常有严格的形式，但编程实现时则可不拘一格。

我们以实例深入讨论。

〖问题 6〗旅馆经营问题

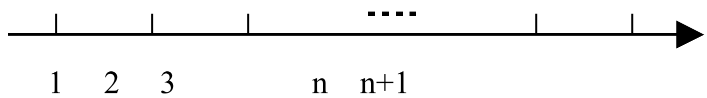
Patiya 城是一座美丽的旅游城市，每年假期都有众多游客来到 Patiya 度假 。Royal Rose 酒店就坐落在 Patiya 的海滨，它共有 R 间客房。假期将要来临，酒店收到了许多各大旅游公司的客房订单，但是毕竟酒店的客房数有限，不能容纳所有的游客。现在假设你是这家酒店馆的经理，请你有选择地接受一部分订单，使你的酒店可以从这些订单中得到最大的收益。每份订单给出了它的起止时间和预定房间数。一旦你接受了某份订单，在订单规定的时间内，你的酒店就必须为旅客提供整洁的房间，否则你将因为失去信誉而被辞退。

我们首先要在原题的基础上，剔除与解题无关的信息，建立第一类模型。

模型 6.1

建立时间轴，设从第一个旅客进驻酒店，直到最后一个旅客离开饭店，共

为期 n 天。把第一天开始的时刻编号为 1，一、二两天分隔的时刻编号为 2，最后到把最后一天结束的时刻编号为 n+1。把每份订单用时间轴上的一条以这些点为端点的线段来表示，设以顶点 i,j（i<j）为端点的线段条数为 c[i,j]。要求在这些线段中选择一部分来覆盖时间轴，使得每段线段的被覆盖次数不超过 R，并且覆盖的线段总长度最长。



模型 6. 1 把问题转化为了一个更加直观的形式，状态更加明确了。在此基础上，我们首先考虑到的是搜索算法。比如说，对每一线段的次数逐一枚举,寻找最优解。盲目搜索是一种基本算法，其解题过程对问题的分析还停留在表层。尽管搜索还可以做很多优化，但它仍然是一个时间复杂度为指数阶的算法。为了得到高效算法，对本题的求解做了一个尝试，使用贪心算法求解：

```
procedure Hotel_2;
1    设置线段集 S 为所有覆盖线段的集合;
2    设置线段集 A 为空集;
3    for I:=1 to R do
4          从 S 寻找一组互相不重叠的线段的集合 V，使它们在时间轴上覆盖的
     长度之和最长;
5          S     S – V;
6          A     A + V;
7    输出 A;
```

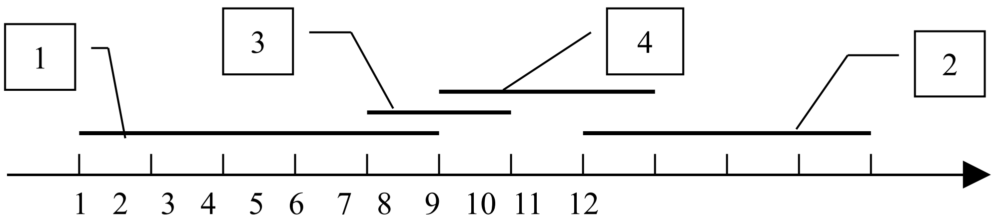
上述算法的第 4 步可以用动态规划实现。用 d[I]表示覆盖(1,I)的一组互相不重叠的线段的最大长度和。

D[1]=0

$$D[i] = \max_{j=1}^{i-1} \{D[j] + i - j\} (c0[i,j] > 0)$$

c0[I,J]表示在剩余线段集 S 中线段(I,J)的次数。

我们对这诱人的算法抱有很大的希望，但是算法的正确性毕竟要接受实践的检验。暂不证明这个算法正确性（最优性），我们考虑是否能举出一个反例否定它。



一个反例

如上图，若采用上述贪心算法，第一次选择的是线段 1 和 2，第二次选择的是线段 4，这样总长度为 12。但实际上本题的最优解是 4 条线段都选择，总长度为 14。可见上述贪心只是一个近似算法。于是我们又将上述算法的第 4 步改为：

从 S 寻找一组线段的集合 V，使得 A+V 中每条线段的覆盖次数不超过 I，并且它们在时间轴上覆盖的长度之和最长;

这个算法初看似乎可以对付上述反例，但我们仍可以举出类似的反例否定它。

经过对贪心算法的探讨，我们发现求解过程中各种数量之间的制约关系较为复杂，一般的图论模型无法描述。因此我们想到了迄今为止最复杂的“流网络”的模型。是否能利用“流网络极值”问题的高效算法解决本题呢？尝试用如下方式构造本题的“流网络”：

模型 6.2

(1) 构造 n+1 个顶点(从 1 到 n+1 编号)，代表时间轴上的 n+1 个标度，并给源和汇分别编号为 0 和 n+2;

(2) 从点 I 到 I+1(0≤I≤n+1)，画一条容量上限为+∞，费用为 0 的弧。

(3) 对每条线段<I,J>（起点和终点分别为 I 和 J），画一条从点 I 到点 J 的弧，其容量上限为 c[I,J]，费用为它的长度的弧。它的流量表示这条线段的覆盖次数。

对上面构造的流网络求流量为 R 的最大费用流，其流量对应的覆盖方式就是原问题的解。

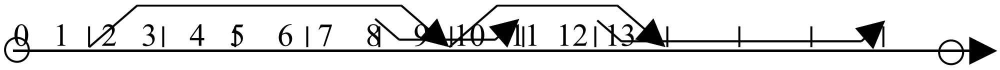
算法描述：

proceudre Hotel_3;

1 构造流网络;

```
for I:=1 to R do
    寻找一条具有最大费用的增流路径；
输出流网络对应的解答
```

仍然以上面那个例子来说明求最大费用流的过程。



如图，第一次增流的路径为<0,1,6,7,8,12,13>，第二次增流的路径为<0,1,2，3,4,5,7,6,9,10,11,12,13>，这时的最大费用为 14。通过对求最大费用流过程的剖析与它和贪心算法的对比，发现了它们的本质区别在于允许<7,6>的逆向流过程。

总结

构造与建模是一个复杂的抽象过程。我们要善于透视问题的本质，寻找突破口，进而选择适当的模型。构造的模型可以帮助我们认识问题，不同的模型从不同的角度反映问题，可以引发不同的思路，起到引导发散思维的作用。但认识问题的最终目的是解决问题，模型的固有性质可帮我们建立算法，其优劣可以通过时空复杂度等指标来衡量，但要以最终实现的程序为标准。所以模型不是一成不变的，同样要通过各种技术不断优化。模型的产生虽然是人脑思维的产物，但它仍然是客观事物的在人脑中的反映。所以要培养良好的建模能力，还必须靠在平时的学习中积累丰富的知识和经验。

【参考书目】
算法与数据结构　（傅清祥　王晓东 编著）
信息学（计算机）奥林匹克入门　（江文哉主编）
信息学奥林匹克杂志