第 1 讲　LINGO 软件的基本用法

司守奎
烟台市，海军航空工程学院数学教研室
Email：sishoukui@163.com

LINGO 软件是美国 LINDO 系统公司开发的一套专门用于求解最优化问题的软件。它为求解最优化问题提供了一个平台，主要用于求解线性规划、非线性规划、整数规划、二次规划、线性及非线性方程组等问题. 它是最优化问题的一种建模语言，包含有许多常用的函数供使用者编写程序时调用，并提供了与其他数据文件的接口，易于方便地输入，求解和分析大规模最优化问题，且执行速度快.。由于它的功能较强，所以在教学、科研、工业、商业、服务等许多领域得到了广泛的应用。

1.1　LINGO 软件简介

1.1.1　LINGO 软件的特点

LINGO 语言是一个综合性的工具，使建立和求解数学优化模型更容易，更有效。LINGO 提供了一个完全集成的软件包，包括强大的优化模型描述语言，一个全功能的建立和编辑模型的环境，和一套快速内置的求解器，能够有效地解决大多数优化模型。LINGO 包括如下的基本特点：

1.代数模型语言

LINGO 支持强大的集模型语言，它使得用户能够高效紧凑地表示数学规划模型。多数模型可以用 LINGO 的内置脚本进行迭代求解。

2.方便的数据选项

LINGO 使得从用户从费时费力的数据管理中解脱出来。它允许你直接从数据库和电子表格中获取信息建立模型，同样，LINGO 能够把解输出到数据库或电子表格，使你能更容易生成你选择的应用报告。完整的模型和数据的分离，提高模型的维护性和扩展性。

3.模型交互性或创建交钥匙工程的应用

你可以用 LINGO 建立或求解模型，或者你可以直接从所写的应用中直接调用 LINGO。为了提高模型的交互性，LINGO 提供了一个完整的建模、求解和分析模型的环境。为了建立交钥匙的解决方案，LINGO 可以调用用户所写的 DLL 和 OLE 应用接口。LINGO 可以直接调用 Excel 宏或数据库应用， LINGO 目前包括 C/C++、Fortran、Java、C#.net、VB.NET、ASP.NET、Visual Basic、Delphi 和 Excel 编程实例。

4.广泛的文档和帮助

LINGO 提供了所有你需要快速启动和运行的工具。你可以得到 LINGO 用户手册，它描述了程序的命令和功能。还包括 LINGO 优化建模的较大超级版本，讨论了所有类型的线性、整数和非线性优化问题的综合建模文档。LINGO 提供了许多现实世界的建模实例供用户修改和扩展。

5.强大的求解器和工具

LINGO 提供了一套全面快速的内置求解器，求解线性、非线性（凸与非凸）、二次、二次约束的和整数优化。你永远不必指定或加载一个单独的求解器，因为 LINGO 读取你的公式，自动选择一个合适的求解器。LINGO 中有如下一般描述的求解器和工具：

（1）一般非线性求解器

LINGO 提供了一般非线性和非线性整数的功能。非线性许可证选择要使用的非线性功能与 LINDO API。

（2）全局求解器

全局求解器结合了一系列的边界（例如，区间分析和凸分析）和范围减少技术（例如，线性规划和约束传播），在一个分支定界框架内找到非凸非线性规划的行之有效全局解。传统的非线性求解器只能求得次优的局部解。

（4）多初值求解器

多初值求解器能在非线性规划和混合整数非线性规划的解空间中智能地生成一系列候选初始点。一个传统的非线性求解器调用每个初始点，找到一个局部最优解。对于非凸非线

性规划模型，由多初值求解器求得的最好解的质量往往要优于传统的非线性求解器从单一初值求得的解。用户可以调节参数控制多初值点的最大个数。

（5）障碍求解器

障碍求解器是求解线性、二次和二次锥问题的一种可选方法。LINGO 使用最先进技术的障碍方法，为大型稀疏模型提供了超高速的算法。

（6）单纯形求解器

LINGO 提供的原始和对偶单纯形两个先进算法作为求解线性规划模型的主要工具。其灵活的设计允许用户通过改变几个算法参数，微调每一种算法。

（7）混合整数求解器

LINGO 的混合整数求解器功能扩展到线性、二次和一般非线性整数模型。它包含了一些先进的求解技术，如切割生成、树排序减少树的动态生长以及先进的启发式和预求解策略。

（8）随机求解器

随机规划求解器通过多阶段随机模型，提供了不确定条件下的决策机会。用户通过辨识内置的或用户定义的描述每一个随机变量的分布函数，来描述不确定性。随机求解器将优化模型，以最大限度地减少初始阶段的成本和预期的成本。高级采样模式也可用于近似连续分布。

（9）模型及求解分析工具

LINGO 对于不可行线性、整数和非线性规划，包括一套全面的分析调试工具，采用先进的技术来分离源于不可行的原始约束的最小子集。它也有工具来执行灵敏度分析以确定某些最优基的敏感性。

（10）二次识别工具

二次规划识别工具是一个有用的代数预处理器，自动确定任意非线性规划是否实际上是一个凸二次规划模型。然后二次规划模型可以通过更快的二次求解器，它是作为障碍求解器选项的一部分。当障碍求解器选择全局选项时，会自动识别二阶锥模型，和凸二次规划模型。

（11）线性化工具

线性化是一个综合性的重构工具，自动将许多非光滑函数和操作符（例如，最大和绝对值）变换到线性系列的数学等价表达式。许多非光滑模型可以完全线性化。这让线性求解器很快找到一个全局最优解，否则将是一个棘手的非线性问题。

1.1.2 LINGO 软件的界面介绍

下面简要地介绍 LINGO 软件的模型窗口、运行状态窗口和一些重要求解参数的设置。

1.模型窗口

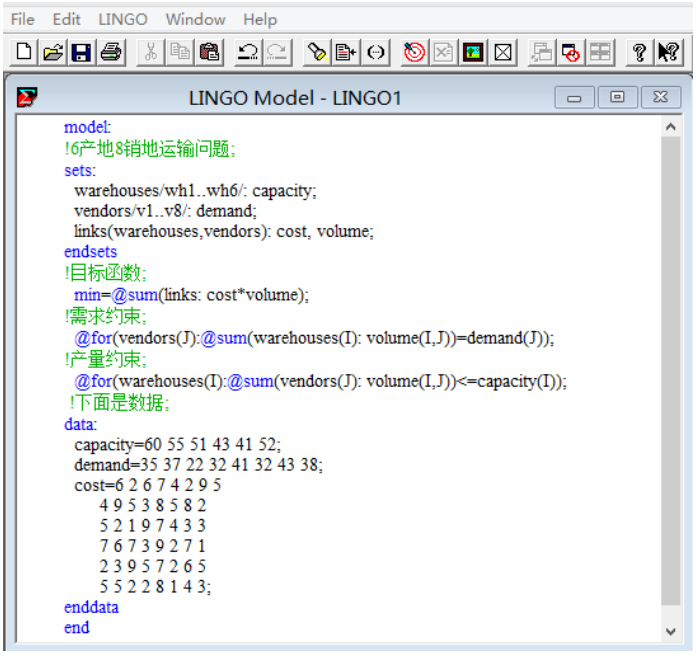


图 1. 1　LINGO 的模型窗口

LINGO 的模型窗口如图 1. 1 所示。模型窗口输入格式要求如下：

（1）LINGO 的数学规划模型包含目标函数、决策变量、约束条件三个要素。

（2）在 LINGO 程序中，每一个语句都必须要用一个英文状态下的分号“;”结束，一个语句可以分几行输入。

（3）LINGO 的注释以英文状态的感叹号“！”开始，必须以英文状态下的分号“;”结束；

（4）LINGO 的变量不区分字母的大小写，必须以字母开头，可以包含数字和下划线，不超过 32 个字符，。

（5）LINGO 程序中，只要定义好集合后，其他语句的顺序是任意的。

（6）LINGO 中的函数以“@”开头。

（7）LINGO 程序默认所有的变量都是非负的。

（8）LINGO 程序中“>或<”号与“≥或≤”号功能相同。

（9）LINGO 模型以语句“MODEL:”开始，以“END”结束，对于比较简单的模型，这两个语句可以省略。

LINGO 建模时需要注意如下几个基本问题：

（1）尽量使用实数变量，减少整数约束和整数变量。

（2）模型中使用的参数数量级要适当，否则会给出警告信息，可以选择适当的单位改变相对尺度。

（3）尽量使用线性模型，减少非线性约束和非线性变量的个数，同时尽量少使用绝对值、符号函数、多变量求最大最小值、取整函数等非线性函数。

（4）合理设定变量上下界，尽可能给出初始值。

2.LINGO 的求解器运行状态窗口

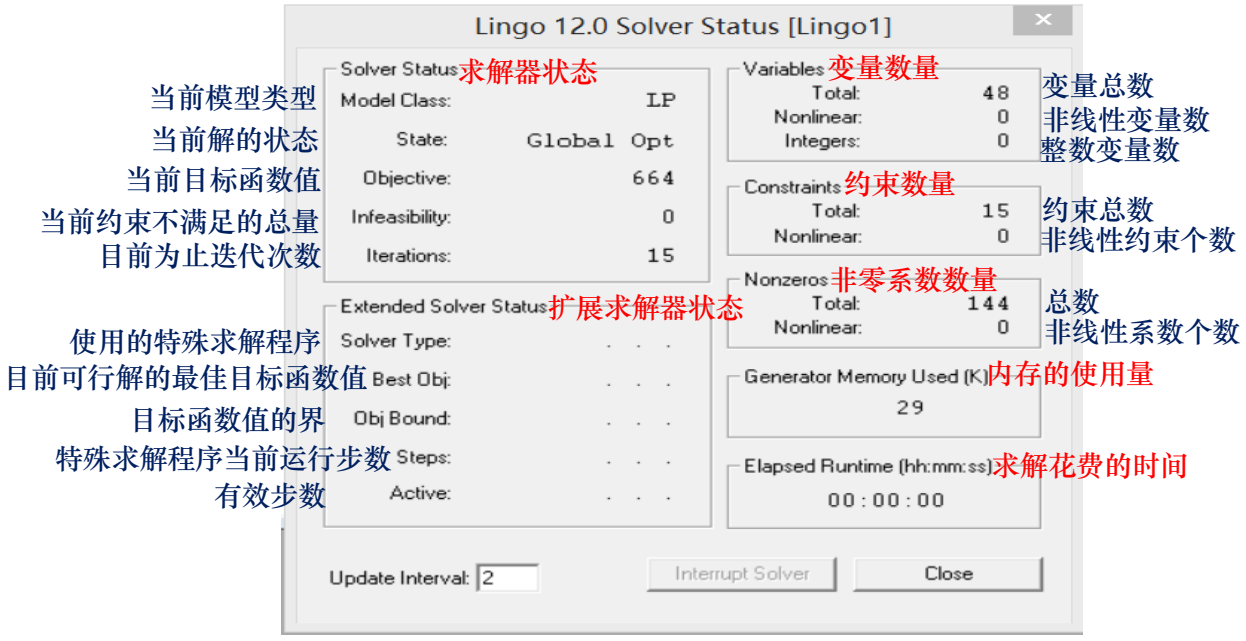


图 1. 2　LINGO 的求解器状态窗口

LINGO 求解器运行状态窗口如图 1. 2 所以。其中的两个状态框介绍如下：

（1）求解器状态框

“当前解的状态”有如下几种：

Global Optimum　全局最优解；

Local Optimum　局部最优解；

Feasible　可行解；

Infeasible　不可行解；

Unbounded　无界解；

Interrupted　中断；

Undetermined　未确定。

（2）扩展求解器状态

“使用的特殊求解程序”有如下几种：

B-and-B 分支定界算法。

Global 全局最优求解程序。

Multistart 用多个初始点求解的程序。

3.LINGO 求解的参数设置

LINGO10 软件管理的内存最大为 2G，如果你的计算机内存是 4G 的话，LINGO 的内存就设置为 2G，你的计算机内存是 8G 的话，也要设置成 2G。

LINGO 内存的设置是依次选择菜单 LINGO(第 3 个主菜单)→Options…→Model Generator。如图 1. 3 所示红色标注的部分。

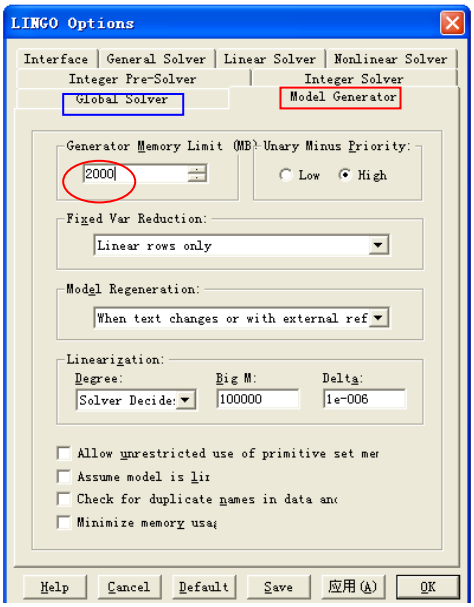


图 1. 3 求解器 Options 的一些设置

如果模型是非线性模型，且欲求全局最优解，要把求解器设置成“Global”。进入图 1. 3 中蓝色“Global Solver”后，在“Use Global Solver”前面打上“√”，设置完成后，要关闭 LINGO 软件，再重新启动 LINGO 软件。

1.1.3 初识 LINGO 程序

LINGO 程序书写实际上特别简捷，数学模型怎样描述，LINGO 语言就对应地怎样表达。首先介绍两个简单的 LINGO 程序。

例 1. 1 求解如下的线性规划问题：

$$\max \quad z = 72x_1 + 64x_2,$$

$$\text{s. t.} \begin{cases} x_1 + x_2 \le 50, \\ 12x_1 + 8x_2 \le 480, \\ 3x_1 \le 100, \\ x_1, x_2 \ge 0. \end{cases}$$

解 LINGO 求解程序如下：

```
max=72*x1+64*x2;
x1+x2<=50;
12*x1+8*x2<=480;
3*x1<=100;
```

注 1. 1：LINGO 中默认所有的变量都是非负的，在 LINGO 中就不需写出对应的约束。

例 1. 2 抛物面 $z = x^2 + y^2$ 被平面 $x + y + z = 1$ 截成一椭圆，求原点到这椭圆的最短距离。

解　该问题可以用拉格朗日乘子法求解。下面我们把问题归结为数学规划模型，用 LINGO 软件求解。

设原点到椭圆上点 (x,y,z) 的距离最短，建立如下的数学规划模型：

$$\min \sqrt{x^2+y^2+z^2},$$
$$\text{s.t.} \begin{cases} x+y+z=1, \\ z=x^2+y^2. \end{cases}$$

LINGO 求解程序如下：

```
min=(x^2+y^2+z^2)^(1/2);
x+y+z=1;
z=x^2+y^2;
@free(x); @free(y);
```

注 1.2：LINGO 中默认所有变量都是非负的，这里 x,y 的取值是可正可负的，所以使用 LINGO 函数 free。

例 1.3　求解如下的数学规划模型：

$$\min \sqrt{\sum_{i=1}^{100} x_i^2},$$
$$\text{s.t.} \begin{cases} \sum_{i=1}^{100} x_i = 1, \\ x_{100} = \sum_{i=1}^{99} x_i^2. \end{cases}$$

解　用 LINGO 求解上述数学规划问题，使用下面将介绍的集合和函数比较方便，使用集合的目的是为了定义向量，集合使用前，必须先定义；LINGO 程序中的标量不需要定义，直接使用即可。

LINGO 求解程序如下：

```
sets:
var/1..100/:x;
endsets
min=@sqrt(@sum(var(i):x(i)^2));
@sum(var(i):x(i))=1;
x(100)=@sum(var(i)|i#le#99:x(i)^2);
@for(var(i)|i#le#99:@free(x(i)));
```

注 1.3：如果不使用集合和函数，全部使用标量 x1,x2,…,x100，最后一个约束就要写 99 遍，@free(x1); …; @free(x99)。

1.1.4　线性规划问题的影子价格与灵敏度分析

以例 1.1 的线性规划模型

$$\max \quad z = 72x_1 + 64x_2,$$
$$\text{s.t.} \begin{cases} x_1 + x_2 \le 50, \\ 12x_1 + 8x_2 \le 480, \\ 3x_1 \le 100, \\ x_1, x_2 \ge 0. \end{cases}$$

为例。

1.影子价格

要进行灵敏度分析，必须选择如图 1.4 所示的画圈的选项，依次选择下列菜单 LINGO→Options……General Solver 下 Dual Computations 选择 Prices。

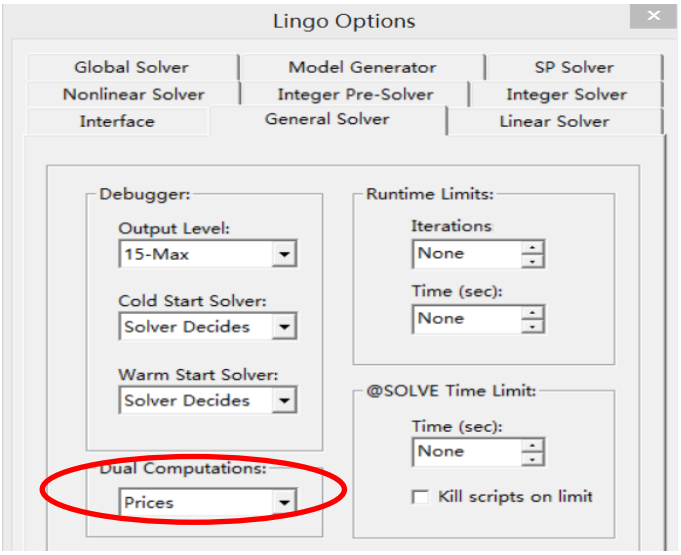


图 1.4　LINGO Options 设置

勾选了 Prices 选项后，运行 LINGO 程序，输出结果窗口中包含灵敏度分析，如图 1.5 所示。

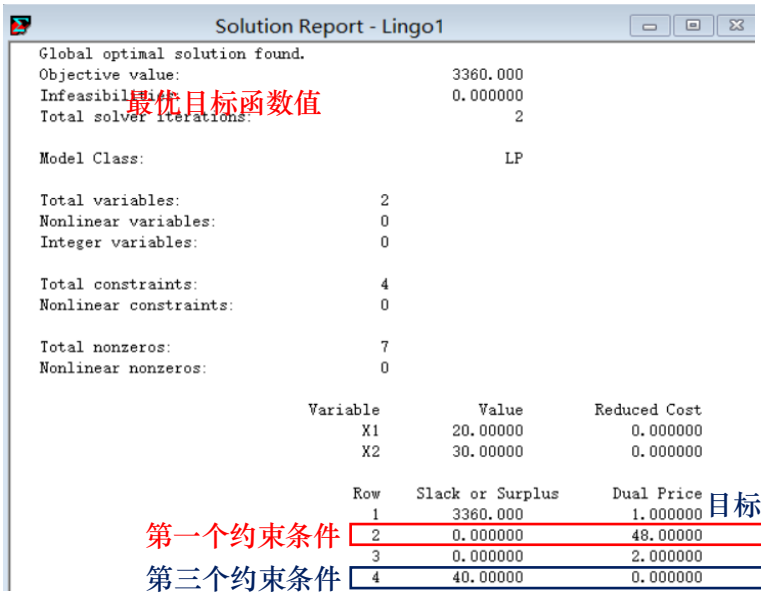


图 1.5　灵敏度分析

从结果可知，目标函数的最优值为 3360，决策变量 $x_1=20, x_2=30$。

（1）reduced cost 值对应于单纯形法计算过程中各变量的检验数。

（2）图 1.5 中红色方框表示第一个约束条件，Slack or Surplus 值为 0 表示该约束松弛变量为 0，约束等号成立，为紧约束或有效约束。蓝色方框表示第三个约束松弛变量为 40，不等号成立，有剩余。

（3）Dual Price 对应影子价格，红色方框表示当第一个约束条件右端常数项增加 1 个单位，即由 50 变为 51 时，目标函数值增加 48，即约束条件 1 所代表的资源的影子价格。蓝色方框表示，第三个约束条件右端常数项增加 1 个单位时，目标函数值不变。

2.确保最优基不变的系数变化范围

如果想要研究目标函数的系数和约束右端常数项系数在什么范围变化（假定其他系数保持不变）时，最优基保持不变。此时需要首先勾选图 1.6 所示的选项。

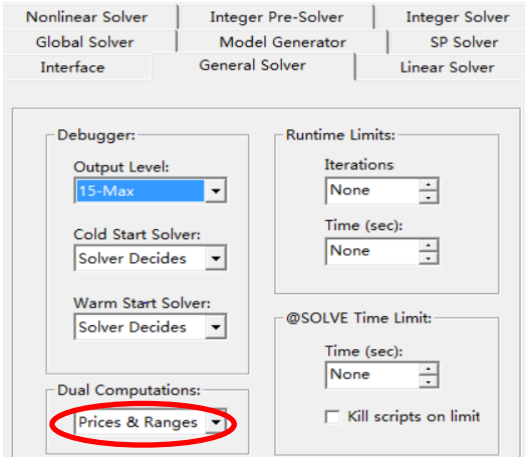
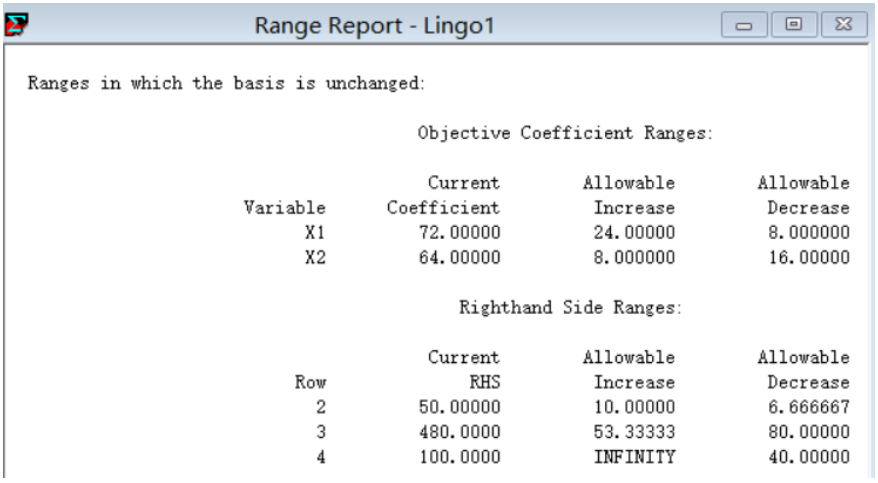


图 1. 6　LINGO Options 对话框

此时不需要重新运行程序，关闭输出窗口，从菜单命令 LINGO 中选中“Range”，即可得到如下输出窗口，如图 1. 7。



Range Report - Lingo1

Ranges in which the basis is unchanged:

Objective Coefficient Ranges:

Variable	Current Coefficient	Allowable Increase	Allowable Decrease
X1	72.00000	24.00000	8.000000
X2	64.00000	8.000000	16.00000

Righthand Side Ranges:

Row	Current RHS	Allowable Increase	Allowable Decrease
2	50.00000	10.00000	6.666667
3	480.0000	53.33333	80.00000
4	100.0000	INFINITY	40.00000

图 1. 7　灵敏度分析范围变化输出窗口

（1）Objective Coefficient Ranges 一栏反映了目标函数中决策变量的价值系数，可以看到 x_1 的系数是 72，x_2 的系数是 64，说明 x_1 要想确保当前最优基不变，在其他系数不变的情况下，x_1 系数的变化范围为 $(64,96)$，当 x_1 的系数在这个范围内变化时 ，最优解不变，但是最优目标函数值发生变化，同样，x_2 系数的变化范围为 $(48,72)$ 。

（2）Righthand Side Ranges 一栏反映了约束条件右端代表资源系数的常数项，可见第一个约束右端常数项在 $(43.333333,60)$ 变化时，最优基不变，但是最优解发生变化，目标函数值也相应地发生变化。由于第三个约束松弛变量为 40，有剩余，可见无论再如何增加该资源，只会使剩得更多，对解没有影响，但是如果减少量超过 40，就会产生影响。

1.2　LINGO 模型的基本组成

用 LINGO 语言编写程序来表达一个实际优化问题，称之为 LINGO 模型。下面以一个运输规划模型为例说明 LINGO 模型的基本组成。

例 1. 4　已知某种商品 6 个仓库的存货量，8 个客户对该商品的需求量，单位商品运价如表 1. 1 所示。试确定 6 个仓库到 8 个客户的商品调运数量，使总的运输费用最小。

表 1. 1　单位商品运价表

单位运价 客户 / 仓库	V1	V2	V3	V4	V5	V6	V7	V8	存货量
W1	6	2	6	7	4	2	5	9	60
W2	4	9	5	3	8	5	8	2	55
W3	5	2	1	9	7	4	3	3	51

w4	7	6	7	3	9	2	7	1	43
W5	2	3	9	5	7	2	6	5	41
W6	5	5	2	2	8	1	4	3	52
需求量	35	37	22	32	41	32	43	38	

解　设 $x_{ij}(i=1,2,\cdots 6;\ j=1,2,\cdots,8)$ 表示第 i 个仓库运到第 j 个客户的商品数量，c_{ij} 表示第 i 个仓库到第 j 个客户的单位运价，d_j 表示第 j 个客户的需求量，e_i 表示第 i 个仓库的存货量，建立如下线性规划模型

$$\min \quad \sum_{i=1}^{6}\sum_{j=1}^{8} c_{ij}x_{ij}\ ,$$

$$\text{s.t.}\ \begin{cases} \sum\limits_{j=1}^{8} x_{ij} \le e_i, \quad i=1,2,\cdots,6, \\ \sum\limits_{i=1}^{6} x_{ij} = d_j, \quad j=1,2,\cdots,8, \\ x_{ij} \ge 0, \quad i=1,2,\cdots,6;\ j=1,2,\cdots,8. \end{cases}$$

LINGO 模型一般由 3 个部分构成：集合，数据和初始，目标函数与约束条件。

1.2.1　集合定义部分

集合是一群相联系的对象，这些对象也称为集合的**成员**。LINGO 将集合（set）的概念引入建模语言，代表模型中的实际事物，并与数学变量及常量联系起来，是实际问题到数学的抽象。例 1. 4 中的 6 个产地可以看成是一个集合，8 个销地可以看成另一个集合。

LINGO 有两种类型的集合：原始集合(primitive　set)和派生集合(derived set)。

一个原始集合是由一些最基本的对象组成的，不能再被拆分成更小的组分。

一个派生集合是用一个或多个其它集合来定义的，也就是说，它的成员来自于其它已存在的集合。

集合部分是 LINGO 模型的一个可选部分。在 LINGO 模型中使用集合之前，必须在集合部分事先定义。集合部分以关键字“sets:”开始，以“endsets”结束。一个模型可以没有集合部分，或有一个简单的集合部分，或有多个集合部分。一个集合部分可以放置于模型的任何地方，但是一个集合及其属性在模型约束中被引用之前必须定义了它们。

1.定义原始集合

为了定义一个原始集合，必须详细声明：

- 集合的名称；
- 集合的成员（可选的）；
- 集合成员的属性（可选的）。

定义一个原始集合，用下面的语法：

setname[/member_list/][:attribute_list];

注意：用“[]”表示该部分内容可选。下同，不再赘述。

setname 是用来标记集合的名字，最好具有较强的可读性。集合名称必须严格符合标准命名规则：以字母为首字符，其后由字母（A-Z）、下划线、阿拉伯数字（0，1，…，9）组成的总长度不超过 32 个字符的字符串，且不区分大小写。

注 1. 4：该命名规则同样适用于集合成员名和属性名等的命名。

member_list 是集合成员列表。如果集合成员放在集合定义中，那么对它们可采取显式罗列和隐式罗列两种方式。如果集合成员不放在集合定义中，那么可以在随后的数据部分定义它们。

① 当显式罗列成员时，必须为每个成员输入一个不同的名字，中间用空格或逗号搁开，允许混合使用。

② 当隐式罗列成员时，不必罗列出每个集成员。可采用如下语法：

setname/member1..memberN/[: attribute_list];

这里的 member1 是集合的第一个成员名，memberN 是集合的最末一个成员名。LINGO 将自动产生中间的所有成员名。LINGO 也接受一些特定的首成员名和末成员名，用于创建一些特殊的集合，如表 1. 2。

表 1. 2　隐式罗列成员示例

隐式成员列表格式	示例	所产生集成员
1..n	1..5	1,2,3,4,5
StringM..StringN	Car2..car14	Car2,Car3,Car4,…,Car14
DayM..DayN	Mon..Fri	Mon,Tue,Wed,Thu,Fri
MonthM..MonthN	Oct..Jan	Oct,Nov,Dec,Jan
MonthYearM..MonthYearN	Oct2001..Jan2002	Oct2001,Nov2001,Dec2001,Jan2002

在例 1. 4 中需要定义仓库集合：

warehouses/1..6/: e;

其中 warehouses 是集合的名称，1..6 是集合内的成员，“..”是特定的省略号（如果不用省略号，也可以把成员一一罗列出来，成员之间用逗号或空格分开），表明该集合有 6 个成员，分别对应于 6 个仓库，e 是集合的属性，它可以看成是一个一维数组，有 6 个分量，分别表示各仓库的存货量。

仓库集合也可以定义为：

warehouses/W1　W2　W3　W4　W5　W6/: e;

或者

warehouses/W1..W6/: e;

在例 1. 4 中还需要定义客户集合：

vendors/V1..V8/: d;

该集合有 8 个成员，d 是集合的属性（有 8 个分量）表示各客户的需求量。

原始集合的属性相当于一维数组。

2.定义派生集合

为了定义一个派生集，必须详细声明：

- 集合的名字；
- 父集合的名字；
- 集合成员（可选）；
- 集合成员的属性（可选）。

可用下面的语法定义一个派生集合：

setname(parent_set_list)[/member_list/][:attribute_list];

setname 是集合的名字。parent_set_list 是已定义集合的列表，多个时必须用逗号隔开。如果没有指定成员列表，那么 LINGO 会自动创建父集合成员的所有组合作为派生集合的成员。派生集合的父集合既可以是原始集合，也可以是其他的派生集合。

例 1. 5　集合定义示例。

```
sets:
product/A B/;
machine/M N/;
week/1..2/;
allowed(product,machine,week):x;
endsets
```

LINGO 生成了三个父集合的所有组合共八组作为 allowed 集合的成员。如表 1. 3 所列。

表 1. 3　集合 allowed 的成员

编号	成员
1	(A,M,1)
2	(A,M,2)
3	(A,N,1)
4	(A,N,2)

5	(B,M,1)
6	(B,M,2)
7	(B,N,1)
8	(B,N,2)

成员列表被忽略时，派生集合成员由父集合成员所有的组合构成，这样的派生集合称为稠密集。如果限制派生集合的成员，使它成为父集合成员所有组合构成的集合的一个子集，这样的派生集合称为稀疏集。同原始集合一样，派生集合成员的声明也可以放在数据部分。一个派生集合的成员列表有两种方式生成：

① 显式枚举；

② 设置成员资格过滤器。

当采用方式①时，必须显式枚举出所有要包含在派生集合中的成员，并且枚举的每个成员必须属于稠密集。在例 1. 5 中，显式枚举派生集合的成员：

allowed(product,machine,week)/A M 1,A N 2,B N 1/;

如果需要生成一个大的、稀疏的集，那么显式枚举就很讨厌。幸运地是许多稀疏集的成员都满足一些条件以和非成员相区分。我们可以把这些逻辑条件看作过滤器，在 LINGO 生成派生集合的成员时把使逻辑条件为假的成员从稠密集中过滤掉。

在例 1. 4 中，为了表示数学模型中从仓库到客户的运输关系以及与此相关的运输单价 c_{ij} 和运量 x_{ij}，要定义一个表示运输关系的派生集合：

links(warehouses,vendors): c,x;

其中 c 和 x 是该派生集合的两个属性，分别表示运输单价 c_{ij} 和运量 x_{ij}。

例 1. 4 模型的完整集合定义如下：

```
sets:
   warehouses/1..6/: e;
   vendors/1..8/: d;
   links(warehouses,vendors): c,x;
endsets
```

综上所述，LINGO 中的集合类型见图 1. 8。

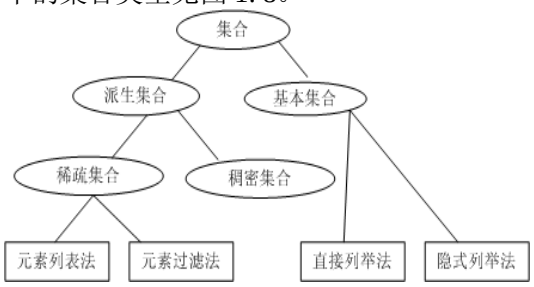


图 1. 8　集合类型示意图

1.2.2　模型的数据部分和初始部分

在处理模型的数据时，需要为集合指派一些成员并且在 LINGO 求解模型之前为集合的某些属性指定值。为此，LINGO 为用户提供了两个可选部分：输入集合成员和数据的数据部分（Data Section）和为决策变量设置初始值的初始部分（Init Section）。

1.模型的数据部分

数据部分以关键字“data:”开始，以关键字“enddata”结束。在这里，可以指定集合成员、集合的属性。其语法如下：

object_list = value_list;

其中 object_list 中包含要设置集合成员的集名、要指定值的属性名，用逗号或空格隔开。如果 object_list 中有多个属性名，那么它们必须定义在同一个集合上。如果 object_list 中有一个集合名，那么它必须是 object_list 中任何属性的父集合。

value_list 包含要分配给 object_list 中的对象的值，用逗号或空格隔开。注意属性值的个数必须等于集合成员的个数。看下面的例子。

例 1. 6

```
model:
sets:
SET1: X, Y;
endsets
data:
SET1 = A B C;
X = 1 2 3;
Y = 4 5 6;
enddata
end
```

在集 set1 中定义了两个属性 X 和 Y。X 的三个值是 1、2 和 3，Y 的三个值是 4、5 和 6。也可采用下面的复合数据声明实现同样的功能。

```
model:
sets:
SET1: X, Y;
endsets
data:
SET1 X Y = A 1 4
B 2 5
C 3 6;
enddata
end
```

要记住一个重要的事实是，当 LINGO 读取复合数据声明的值列表时，它将前 *n* 个值分别分配给对象列表中 *n* 个对象的第一个值，第二批 *n* 个值的依次分配给 *n* 个对象的第二个值，依次类推。换句话说，LINGO 期望的是列格式，而不是行格式的输入数据，这反映了关系数据库中记录和字段之间的关系。

例 1. 4 中的集合和数据部分可以如下定义：

```
sets:
   warehouses/1..6/: e;
   vendors/1..8/: d;
   links(warehouses,vendors): c,x;
endsets
data:   !数据部分;
e= 60 55 51 43 41 52;   !属性值;
d=35 37 22 32 41 32 43 38;
c= 6 2 6 7 4 2 5 9
4 9 5 3 8 5 8 2
5 2 1 9 7 4 3 3
7 6 7 3 9 2 7 1
2 3 9 5 7 2 6 5
5 5 2 2 8 1 4 3;
enddata
```

例 1. 4 中的集合和数据部分也可以如下定义：

```
sets:
warehouses: e;
vendors: d;
links(warehouses, vendors): c, x;
endsets
data: !数据部分;
warehouses = WH1 WH2 WH3 WH4 WH5 WH6;!集合成员;
vendors = V1 V2 V3 V4 V5 V6 V7 V8;
e = 60 55 51 43 41 52; !属性值;
d = 35 37 22 32 41 32 43 38;
c = 6 2 6 7 4 2 5 9
```

```
4 9 5 3 8 5 8 2
5 2 1 9 7 4 3 3
7 6 7 3 9 2 7 1
2 3 9 5 7 2 6 5
5 5 2 2 8 1 4 3;
enddata
```

2. 数据段的两点说明

（1）实时数据处理

在某些情况，对于模型中的某些数据并不是定值。譬如模型中有一个通货膨胀率的参数，我们想在 2%至 6%范围内，对不同的值求解模型，来观察模型的结果对通货膨胀的依赖有多么敏感。我们把这种情况称为实时数据处理。

在本该输入数的地方输入一个问号（?）。

例 1.7

```
data:
  interest_rate,inflation_rate=0.085  ?;
enddata
```

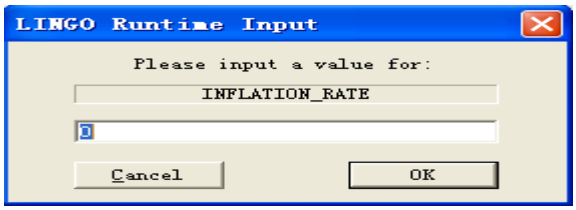


图 1.9　交互式输入对话框

每一次求解模型时，LINGO 都会提示为参数 inflation_rate 输入一个值。在 WINDOWS 操作系统下，将会接收到一个图 1.9 所示的对话框。

直接输入一个值再点击 OK 按钮，LINGO 就会把输入的值指定给 inflation_rate，然后继续求解模型。

（2）部分赋值

有时只想为一个集的部分成员的某个属性指定值，而让其余成员的该属性保持未知，以便让 LINGO 去求出它们的最优值。在数据声明中输入一个逗号（最前面或最后面）或多个相连的逗号表示该位置对应的集成员的属性值未知。

例 1.8

```
sets:
   years/1..5/: capacity;
endsets
data:
   capacity = ,34,20,,;
enddata
```

属性 capacity 的第 2 个和第 3 个值分别为 34 和 20，其余的 3 个值未知。

3.模型的初始部分

初始部分是 LINGO 提供的另一个可选部分。在初始部分中，可以输入初始声明（initialization statement），和数据部分中的数据声明相同。对实际问题的建模时，初始部分并不起到描述模型的作用，在初始部分输入的值仅被 LINGO 求解器当作初始点来用，并且仅仅对非线性模型有用。和数据部分指定变量的值不同，LINGO 求解器可以自由改变初始部分初始化变量的值。

一个初始部分以“init:”开始，以“endinit”结束。初始部分的初始声明规则和数据部分的数据声明规则相同。也就是说，我们可以在声明的左边同时初始化多个集属性，可以把集属性初始化为一个值。

例 1.9

```
init:
X = 0.999;
Y = 0.002;
endinit
```

Y <= @LOG(X);
X^2 + Y^2 <=1;
好的初始点会减少模型的求解时间。

1.2.3 目标函数和约束条件

例 1. 4 中的目标函数表达式 min $\sum_{i=1}^{6}\sum_{j=1}^{8}c_{ij}x_{ij}$ 用 LINGO 语句表示为：

```
min=@sum(links(i,j): c(i,j)*x(i,j));
```

式中，@sum 是 LINGO 提供的内部函数，其作用是对某个集合的所有成员，求指定表示式的和，该函数需要两个参数，第一个参数是集合名称，指定对该集合的所有成员求和；第二个参数是一个表达式，表示求和运算对该表达式进行。此处@sum 的第一个参数是 links(i,j)，表示求和运算对派生集合 links 进行，该集合的维数是 2，共有 48 个成员，运算规则是：先对 48 个成员分别求表达式 c(i,j)*x(i,j)的值，然后求和，相当于求 $\sum_{i=1}^{6}\sum_{j=1}^{8}c_{ij}x_{ij}$ ，表达式中的 c 和 x 是集合 links 的两个属性，它们各有 48 个分量。

注 1. 5　如果表达式中参与运算的属性属于同一个集合，则@sum 语句中索引（相当于矩阵或数组的下标）可以省略，假如表达式中参与运算的属性不同属于不同的集合，则不能省略属性的索引。例 1. 4 中的目标函数可以表示成：

```
min=@sum(links: c*x);
```

例 1. 4 中的约束条件 $\sum_{j=1}^{8}x_{ij}\le e_i, i=1,2,\cdots,6$ 实际上表示了 6 个不等式，用 LINGO 语言表示该约束条件，语句为：

```
@for(warehouses(i):@sum(vendors(j): x(i,j))<=e(i));
```

语句中的@for 是 LINGO 提供的内部函数，它的作用是对某个集合的所有成员分别生成一个约束表达式，它有两个参数，第一个参数是集合名，表示对该集合的所有成员生成对应的约束表达式，上述@for 的第一个参数为 warehouses，它表示仓库，共有 6 个成员，故应生成 6 个约束表达式；@for 的第二个参数是约束表达式的具体内容，此处再调用@sum 函数，表示约束表达式的左边是求和，是对集合 vendors 的 8 个成员求和，即对表达式 x(i,j)中的第二维 j 求和，亦即 $\sum_{j=1}^{8}x_{ij}$ ，约束表达式的右边是集合 warehouses 的属性 e，它有 6 个分量，与 6 个约束表达式一一对应。本语句中的属性分别属于不同的集合，所以不能省略索引 i，j。

同样地，约束条件 $\sum_{i=1}^{6}x_{ij}=d_j, j=1,2,\cdots,8$ 用 LINGO 语句表示为：

```
@for(vendors(j):@sum(warehouses(i): x(i,j))=d(j));
```

1.2.4 完整的模型

综上所述，例 1. 4 的完整 LINGO 模型如下：

```
model:
sets:
   warehouses/1..6/: e;
   vendors/1..8/: d;
   links(warehouses,vendors): c,x;
endsets
data:   !数据部分;
e= 60 55 51 43 41 52;   !属性值;
d=35 37 22 32 41 32 43 38;
c= 6 2 6 7 4 2 5 9
4 9 5 3 8 5 8 2
5 2 1 9 7 4 3 3
7 6 7 3 9 2 7 1
2 3 9 5 7 2 6 5
5 5 2 2 8 1 4 3;
enddata
```

```
min=@sum(links(i,j): c(i,j)*x(i,j));   !目标函数;
@for(warehouses(i):@sum(vendors(j): x(i,j))<=e(i));   !约束条件;
@for(vendors(j):@sum(warehouses(i): x(i,j))=d(j));
end
```

注 1.6　LINGO 模型以 model:开始，以语句 end 结束，这两个语句单独成一行。完整的模型由集合定义、数据段、目标函数和约束条件等部分组成，这几个部分部分的先后次序无关紧要，但集合使用之前必须先定义。

用鼠标点击工具栏上的“求解”按钮，就可以求出上述模型的解。计算结果表明：目标函数值为 664，最优运输方案见表 1.4。

表 1.4　最优运输方案

	V1	V2	V3	V4	V5	V6	V7	V8	合计
W1	0	19	0	0	41	0	0	0	60
W2	1	0	0	32	0	0	0	0	33
W3	0	11	0	0	0	0	40	0	51
w4	0	0	0	0	0	5	0	38	43
W5	34	7	0	0	0	0	0	0	41
W6	0	0	22	0	0	27	3	0	52
合计	35	37	22	32	41	32	43	38	

1.2.5　LINGO 语言的优点

从以上实例可以看出，LINGO 建模语言建立规划模型有如下优点：

（1）对大规模数学规划，LINGO 语言所建模型较简洁，语句不多。

（2）模型易于扩展，因为@for、@sum 等语句并没有指定循环或求和的上下限，如果在集合在集合定义部分增加集合成员的个数，则循环或求和自然扩展，不需要改动目标函数和约束条件。

（3）数据部分与其他部分分开，对同一模型用不同数据来计算时，只需改动数据部分即可，其他语句不变。

（4）“集合”是 LINGO 很有特色的概念，它表达了模型中的实际事物，又与数学变量及常量联系起来，是实际问题到数学量的抽象，它比 C 语言中的数组用途更为广泛，集合中的成员可以随意起名字，没有什么限制，集合的属性可以根据需要确定用多少个，可以用来代表已知常量，也可以用来代表决策变量。

（5）使用了结合以及@for、@sum 等集合操作函数以后可以用简洁的语句表达出常见的规划模型中的目标函数和约束条件，即使模型有大量决策变量和大量数据，组成模型的语句并不随之增加。

1.3　LINGO 的运算符和函数

1.3.1　LINGO 的常用运算符

1.算术运算符

算术运算实际上就是加、减、乘、除、乘方等数学运算，即数与数之间的运算，运算结果也是数，LINGO 中的算术运算符有以下 5 种：

+（加法），-（减号或负号），*（乘法），/（除法），^（求幂）。

2.逻辑运算符

逻辑运算就是运算结果只有“真”（true）和“假”（false）两个值的运算。

在 LINGO 中，逻辑运算符主要用于集循环函数的条件表达式中，来控制在函数中哪些集成员被包含，哪些被排斥。在创建稀疏集时用在成员资格过滤器中。

LINGO 具有 9 种逻辑运算符：

#and#（与）　　仅当两个参数都为 true 时，结果为 true；否则为 false；

#or#（或）　　仅当两个参数都为 false 时，结果为 false；否则为 true；

#not#（非）　　否定该操作数的逻辑值，＃not＃是一个一元运算符；

#eq#（等于）　　若两个运算数相等，则为 true；否则为 false；

#ne#（不等于） 若两个运算符不相等，则为 true；否则为 false；
#gt#（大于） 若左边的运算符严格大于右边的运算符，则为 true；否则为 false；
#ge#（大于等于） 若左边的运算符大于或等于右边的运算符，则为 true；否则为 false；
#lt#（小于） 若左边的运算符严格小于右边的运算符，则为 true；否则为 false；
#le#（小于等于） 若左边的运算符小于或等于右边的运算符，则为 true；否则为 false。
这些运算符的优先级由高到低为：
高 #not#；
#eq# #ne# #gt# #ge# #lt# #le#；
低 #and# #or#。

3.关系运算符

关系运算符表示的是“数与数之间”的大小关系，因此在 LINGO 中用来表示优化模型的约束条件。LINGO 中关系运算符有三种：

<（即<=，小于等于），=（等于），>（即>=，大于等于）.

请注意在优化模型中约束一般没有严格小于、严格大于关系。此外，请注意区分关系运算符与“数与数之间”进行比较的 6 个逻辑运算符的不同之处。

如果需要严格小于和严格大于关系，比如让 A 严格小于 B，那么可以把它变成如下的小于等于表达式：$A+\varepsilon <= B$，这里 ε 是一个小的正数，它的值依赖于模型中 A 小于 B 多少才算不等。

上述三类运算符的优先级如表 1. 5 所示，其中同一优先级按从左到右的顺序执行，如果有括号“（）”，则括号内的表达式有限进行计算。

表 1. 5 逻辑和关系运算符的优先级

优先级	运算符
最高	#not# -（负号）
	^
	* /
	+ -（减法）
	#eq# #ne# #gt# #ge# #lt# #le#
	#and# #or#
最低	< = >

1.3.2 基本的数学函数

内部函数的使用能大大减少用户的编程工作量，所有函数都以“@”符号开头。LINGO 中包括相当丰富的数学函数，我们直接在下面一一列出。

@abs(x)：绝对值函数，返回 x 的绝对值。
@acos(x)：反余弦函数，返回值单位为弧度。
@acosh(x)：反双曲余弦函数。
@asin(x)：反正弦函数，返回值单位为弧度。
@asinh(x)：反双曲正弦函数。
@atan(x)：反正切函数，返回值单位为弧度。
@atan2(y,x)：返回 y/x 的反正切。
@atanh(x)：反双曲正切函数。
@cos(x)：余弦函数，返回 x 的余弦值。
@cosh(x)：双曲余弦函数。
@sin(x)：正弦函数，返回 x 的正弦值，x 采用弧度制。
@sinh(x)：双曲正弦函数。
@tan(x)：正切函数，返回 x 的正切值。
@tanh(x)：双曲正切函数。
@exp(x)：指数函数，返回 e^x 的值。
@log(x)：自然对数函数，返回 x 的自然对数。
@log10(x)：以 10 为底的对数函数，返回 x 的以 10 为底的对数。
@lgm(x)：返回 x 的 gamma 函数的自然对数（当 x 为整数时，lgm(x)=log((x-1)!)）。

@mod(x,y)：模函数，返回 x 除以 y 的余数，这里 x 和 y 应该是整数。
@pi()：返回 pi 的值，即 3.14159265…。
@pow(x,y)：指数函数，返回 x^y 的值。
@sign(x)：如果 x<0 返回-1；如果 x>0，返回 1；如果 x=0，返回 0。
@floor(x)：取整函数，返回 x 的整数部分。当 x>=0 时，返回不超过 x 的最大整数；当 x<0 时，返回不低于 x 的最小整数。
@smax(list)：最大值函数，返回一列数（list）的最大值。
@smin(list)：最小值函数，返回一列数（list）中的最小值。
@sqr(x)：平方函数，返回 x 的平方。
@sqrt(x)：平方根函数，返回 x 的正平方根的值。

1.3.3 集合循环函数

集合循环函数是指对集合中的元素下标进行循环操作的函数，如前面我们用过的@for 和@sum 等。一般用法如下：

集循环函数遍历整个集进行操作。其语法为

@function(setname[(set_index_list)[|conditional_qualifier]]:expression_list);

其中：

function 是集合函数名，是 for，sum，min，max，prod 五种之一；

setname 是集合名；

set_index_list 是集合索引列表，不需使用索引时可以省略；

conditional_qualifier 是用逻辑表达式描述的过滤条件，通常含有索引，无条件时可以省略；

expression_list 是一个表达式，对@for 函数，可以是一组表达式，其间用逗号分隔。

五个集合循环函数的含义如下：

@for（集合元素的循环函数）：对集合 setname 的每个元素独立地生成表达式，表达式由 expression_list 描述，通常是优化问题的约束。

@sum（集合属性的求和函数）：返回集合 setname 上的表达式的和。

@max（集合属性的最大值函数）：返回集合 setname 上的表达式的最大值。

@min（集合属性的最小值函数）：返回集合 setname 上的表达式的最小值。

@prod（集合属性的乘积函数）：返回集合 setname 上的表达式的积。

例 1.10 求向量[5，1，3，4，6，10]前 5 个数的和。

```
model:
data:
   N=6;
enddata
sets:
   number/1..N/:x;
endsets
data:
   x = 5   1   3   4   6   10;
enddata
   s=@sum(number(I) | I #le# 5: x);
end
```

例 1.11 求向量[5，1，3，4，6，10]前 5 个数的最小值，后 3 个数的最大值。

```
model:
data:
   N=6;
enddata
sets:
   number/1..N/:x;
endsets
data:
```

```
   x = 5 1 3 4 6 10;
enddata
   minv=@min(number(I) | I #le# 5: x);
   maxv=@max(number(I) | I #ge# N-2: x);
end
```

例 1.12　事件 A,B,C 相互独立，且已知它们发生的概率 $P(A)=0.3$， $P(B)=0.5$，$P(C)=0.8$，求 A,B,C 中至少有一个事件发生的概率。

解　至少有一个事件发生的概率为

$$P(A\cup B\cup C)=1-P(\bar{A}\bar{B}\bar{C})=1-P(\bar{A})P(\bar{B})P(\bar{C})$$

$$=1-(1-0.3)(1-0.5)(1-0.8)=0.93.$$

计算的 LINGO 程序如下：

```
model:
sets:
components: p;
endsets
data:
p =0.3    0.5    0.8;
enddata
pp=1-@prod(components( i): 1-p(i));
end
```

1.3.4　集合操作函数

集合操作函数是指对集合进行操作的函数，主要有@in，@index，@wrap，@size 四种，下面分别简要介绍其一般用法。

1.@index 函数

使用格式为

@index([set_name,] primitive_set_element)

该函数返回在集 set_name 中原始集成员 primitive_set_element 的索引。如果 set_name 被忽略，那么 LINGO 将返回与 primitive_set_element 匹配的第一个原始集成员的索引。如果找不到，则给出出错信息。

请注意，按照上面所说的索引值的含义，集合 set_name 的一个索引值是一个正整数，即对集合中一个对应元素的顺序编号，且只能位于 1 和集合的元素个数之间。

例 1.13　架设定义了一个 姓名的集合（girls）和一个男孩姓名的集合（boys）如下：

```
sets:
girls/debbie,sue,alice/;
boys/bob,joe,sue,fred/;
endsets
a=@index(sue);   !返回值为 2;
b=@index(boys,sue);   !返回值为 3;
```

可以看到女孩和男孩中都有名为 sue 的小孩。这时，调用函数@index(sue)将返回索引值 2，这相当于@index(girls,sue)，因为集合 girls 的定义出现在集合 boys 之前。如果要找男孩中名为 sue 的小孩的索引，应该使用@ index(boys,sue)，这时将返回索引值 3。

2.@in 函数

使用格式为：

@in(set_name,primitive_index_1 [,primitive_index_2,…])

该函数用于判断一个集合中是否含有某个索引值。如果集合 set_name 中包含由索引 primitive_index_1 [,primitive_index_2,…]所表示的对应元素，则返回 1，否则返回 0。索引用“&1”、“&2”或@index 函数等形式给出，这里“&1”表示对应于第 1 个父集合的元素的索引值，“&2”表示对应于第 2 个父集合的元素的索引值。

例 1.14　我们定义一个学生集合 students（原始集合），然后由它派生一个及格学生的集合 passed 和一个不及格学生的集合 failed，可以定义如下：

```
sets:
students/zhao, qian, sun, li/;
passed(students)/qian, sun/;
failed(students)|#not# @in(passed,&1);
endsets
```

例 1.15　集合 s3 是由集合 s1，s2 派生的。

```
sets:
s1 / a b c/;
s2 / x y z/;
s3( s1, s2) / a,x a,z b,y c,x/;
endsets
```

现在我们想判断 s3 中是否包含元素（b,y），则可以利用以下语句：

x = **@in(s3, @index(s1, b), @index(s2, y))**;

在本例中，s3 中确实包含元素（b,y），所以上面语句的结果是 x=1（真）。你可能已经注意到，这里 x 既是集合 s2 的元素，后来又对 x 赋值 1，这样不会混淆吗？后者会冲掉前者吗？在 LINGO 中这种表达是允许的，因为前者的 x 是集合的元素，后者 x 是变量，二者逻辑上没有任何关系（除了同名外），所以不会出现混淆，更谈不上后者会冲掉前者的问题。

3.@wrap 函数

使用格式为：

@wrap(index,limit)

当 index 位于区间[1,limit]内时直接返回 index；一般地，返回值 j=index-k*limit，其中 j 位于区间[1,limit]，k 为整数。直观地说，该函数把 index 的值加上或减去 limit 的整数倍，使之落在区间[1,limit]中。

例 1.16（职员时序安排模型）一项工作一周 7 天都需要有人（比如护士工作），每天（周一至周日）所需的最少职员数为 20、16、13、16、19、14 和 12，并要求每个职员一周连续工作 5 天，试求每周所需最少职员数，并给出安排。注意这里我们考虑稳定后的情况。

解　设周一至周日分别安排 $x_i(i=1,2,\cdots,7)$ 个人开始上班，周一至周日需要的职员数记为 $r_i(i=1,2,\cdots,7)$，第 i 天工作的人数为

$$x_i + \text{一天前开始工作的人数+两天前开始工作的人数}$$
$$+\text{三天前开始工作的人数+四天前开始工作的人数},$$

借用上面的 LINGO 命令，第 i 天工作的人数可以写作

$$x_i + x_{@\mathrm{wrap}(i-1,7)} + x_{@\mathrm{wrap}(i-2,7)} + x_{@\mathrm{wrap}(i-3,7)} + x_{@\mathrm{wrap}(i-4,7)},$$

需要人数的约束条件可以写为

$$x_i + x_{@\mathrm{wrap}(i-1,7)} + x_{@\mathrm{wrap}(i-2,7)} + x_{@\mathrm{wrap}(i-3,7)} + x_{@\mathrm{wrap}(i-4,7)} \geq r_i .$$

因而，建立如下的数学规划模型

$$\min \sum_{i=1}^{7} x_i ,$$

$$\text{s.t.} \begin{cases} x_i + x_{@\mathrm{wrap}(i-1,7)} + x_{@\mathrm{wrap}(i-2,7)} + x_{@\mathrm{wrap}(i-3,7)} + x_{@\mathrm{wrap}(i-4,7)} \geq r_i, \\ x_i \geq 0 \text{且为整数.} \end{cases}$$

利用 LINGO 软件，求得每周最少需要 22 个员工，周一安排 8 人，周二安排 2 人，周三无需安排人，周四安排 6 人，周五和周六都安排 3 人，周日无需安排人。

计算的 LINGO 程序如下：

```
model:
sets:
   days/mon..sun/: r,x;
endsets
data:
      r = 20   16   13   16   19   14   12; !每天所需的最少职员数;
enddata
   min=@sum(days: x);   !最小化每周所需职员数;
```

```
    @for(days(i):@sum(days(j) |j #le# 5: x(@wrap(i-j+1,7))) >= r (i));
    @for(days:@gin(x));   !约束 x 为整型变量;
end
```

4.@size 函数

使用格式为：

@size(set_name)

返回数据集 set_name 中包含元素的个数。

例 1.17　某公司准备新开发一个产品，为了保证新产品的按期投入市场，公司要对新产品进行 PERT 分析。需要完成的作业由表 1.6 所示。

表 1.6　PERT 网的相关数据

作业	名称	计划完成时间	紧前作业
A	产品设计	10	-
B	需求预测	14	A
C	市场调查	3	A
D	产品定价	3	B，C
E	生产规划	7	B
F	开支预算	4	F
G	人员培训	10	D，F

（1）画出产品计划网络图；

（2）求完成新产品的最短时间，列出各项作业的最早开始时间、最迟开始时间。

解　（1）以 7 项作业作为顶点集，即顶点集 $V=\{\mathrm{A},\mathrm{B},\cdots,\mathrm{G}\}$，紧前作业到后续作业画出有向弧，弧集集合记作 $\tilde{E}$，画出的产品计划有向图见图 1.10。

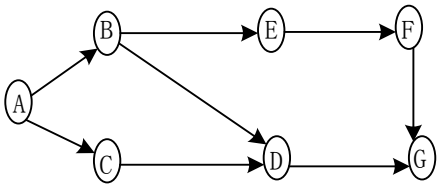


图 1.10　产品计划有向图

（2）用 $i=1,2,\cdots,7$ 分别表示作业 $\mathrm{A},\mathrm{B},\cdots,\mathrm{G}$，第 i 项工作的计划完成时间记为 $t(i)$，第 i 项工作的最早和最晚开工时间分别记为 $t_{es}(i),t_{ls}(i)$，第 i 项工作的松弛时间 $t_s(i)=t_{ls}(i)-t_{es}(i)$，则最早开工时间 $t_{es}(i)$ 和最晚开工时间 $t_{ls}(i)$ 的递推公式为：

$$t_{es}(1)=0,\quad t_{es}(j)=\max_{(i,j)\in\tilde{E}}\{t_{es}(i)+t(i)\},\quad j=2,3,\cdots,7;$$

$$t_{ls}(7)=t_{es}(7),\quad t_{ls}(i)=\min_{(i,j)\in\tilde{E}}\{t_{ls}(i)-t(j)\},\quad i=6,5,\cdots,1.$$

利用 LINGO 软件求得的计算计算结果见表 1.7。所以完成新产品的最短时间 $T=ls(7)+10=45$。

表 1.7　产品计划网络图的计算结果

作业	A	B	C	D	E	F	G
最早开工时间	0	10	10	24	24	31	35
最晚开工时间	0	10	29	32	24	31	35

计算的 LINGO 程序如下：

```
model:
sets:
tasks/A..G/:time,es,ls,slack;
pred(tasks,tasks)/A B,A C,B D,B E,C D,D G,E F,F G/;
endsets
data:
time = 10, 14, 3, 3, 7, 4, 10;
enddata
```

```
@for(tasks(j)|j #gt#1:es(j)=@max(pred(i, j):es(i) + time(i)));
@for(tasks(i)|i #lt#ltask:ls(i)=@min(pred(i,j):ls(j)-time(i)););
@for(tasks(i):slack(i)=ls(i)-es(i));
es(1)=0; !第一项工作的最早开始时间;
ltask=@size(tasks);  !作业的总数量;
ls(ltask)=es(ltask); !最后一项作业的最迟开始时间=最后一项工作的最早开始时间;
end
```

1.3.5 变量定界函数

变量定界函数对函数的取值范围加以限制，共有以下七种函数：

@gin(x)：限制 x 为整数。

@bin(x)：限制 x 为 0 或 1。

@free(x)：取消 x 的非负性限制，即 x 可以取任意实数值。

@bnd(L,x,U)：限制 L<=x<=U。

@sos1('set_name',x)：限制 x 中至多一个大于 0。

@sos2('set_name',x)：限制 x 中至多两个不等于 0，其他都为 0。

@sos3('set_name',x)：限制 x 中正好有一个为 1，其他都为 0。

@card('set_name',x)：限制 x 中非零元素的个数。

@semic(L,x,U)：限制 x=0 或 L<=x<=U。

例 1.18（背包问题）给定 n 种物品和一背包，物品 i 的重量是 w_i，其价值为 v_i，$i=1,2,\cdots,n$，背包的容量为 c，问应如何选择装入背包的物品，使得装入背包中物品的总价值最大。

假设现有 8 件物品，它们的重量分别为 3,4,6,7,9,10,11,12（kg），价值分别为 4,6,7,9,11,12,13,15（元），假设总重量限制不超过 30kg，试决策带哪些物品，使所带物品的总价值最大？

解　引进 0–1 变量

$$x_i=\begin{cases}1, & \text{物品}i\text{放入背包中},\\ 0, & \text{物品}i\text{不放入背包中}.\end{cases}$$

建立如下的 0-1 整数规划模型

$$\max\ z=\sum_{i=1}^{8}v_i x_i\ ,$$

$$\text{s.t.}\begin{cases}\sum_{i=1}^{8}w_i x_i\le c,\\ x_i=1\text{或}0,\ i=1,2,\cdots,8.\end{cases}$$

利用 LINGO 软件，求得的结果为带物品 2、4、5、6，总价值为 38。

计算的 LINGO 程序如下：

```
model:
sets:
items/1..8/:x,w,v;
endsets
data:
w=3,4,6,7,9,10,11,12;
v=4,6,7,9,11,12,13,15;
c=30;
enddata
max=@sum(items:v*x); !目标函数;
@sum(items:w*x)<=c;  !重量约束;
@for(items:@bin(x)); !0-1变量约束;
end
```

例 1.19（配对模型）某公司准备将 8 个职员安排到 4 个办公室，每室两人。根据以往观察，已知有些职员在一起时合作好，有些则不然，表 1. 8 列出了两两之间的不相容程度，数字越小代表相容越好，问如何组合可以使总相容程度最好？

表 1.8　职员之间两两组合的不相容程度

	1	2	3	4	5	6	7	8
1	-	9	3	4	2	1	5	6
2	-	-	1	7	3	5	2	1
3	-	-	-	4	4	2	9	2
4	-	-	-	-	1	5	5	2
5	-	-	-	-	-	8	7	6
6	-	-	-	-	-	-	2	3
7	-	-	-	-	-	-	-	4
8	-	-	-	-	-	-	-	-

注：因为甲与乙配对等同于乙与甲配对，故表中数字只保留对角线上方的内容。

解　用 c_{ij} 表示第 i 人和第 j 人的不相容程度，引进 0–1 变量

$$x_{ij}=\begin{cases}1, & i\text{与}j\text{组合},\\ 0, & i\text{不与}j\text{组合}.\end{cases}\quad (i<j)$$

则目标函数是总的不相容程度最小，约束条件是每人组合一次，即对于职员 i，必有

$$\sum_{j<i}x_{ji}+\sum_{k>i}x_{ik}=1.$$

于是建立 0–1 整数规划模型如下：

$$\min\sum_{i<j}c_{ij}x_{ij},$$

$$\text{s.t.}\begin{cases}\sum_{j<i}x_{ji}+\sum_{k>i}x_{ik}=1,\\ x_{ij}=0\text{或}1,\ i<j.\end{cases}$$

利用 LINGO 软件，求得最优组合方案为：(1,6)，(2,7)，(3,8)，(4,5)，总等级为 6。

计算的 LINGO 程序如下：

```
model:
sets:
ren/ 1..8/;
pairs(ren,ren)|&1 #lt# &2:c,x;
endsets
data:
c=9 3 4 2 1 5 6
      1 7 3 5 2 1
        4 4 2 9 2
          1 5 5 2
            8 7 6
              2 3
                4;
@text()=@table(x); !以表格形式把x的计算结果输出到屏幕;
enddata
min=@sum(pairs:c*x);
@for(ren(i):@sum(pairs(i,j):x(i,j))+@sum(pairs(j,i):x(j,i))=1);
@for(pairs(i,j): @bin(x(i,j)));
end
```

例 1.20　某公司要生产 6 种产品，需要使用 6 种设备。生产每种产品所用的各种设备台时、每件产品利润、固定费用，以及 6 种设备的总台时数见表 1.9。在现有设备总台时的约束下，如何安排生产才能使利润最大。

表 1.9　产品生产数据表

	产品 1	产品 2	产品 3	产品 4	产品 5	产品 6	设备总台时
设备 A	1	4	0	4	2	1	800

设备 B	4	5	3	0	1	0	1160
设备 C	0	3	8	0	1	0	1780
设备 D	2	0	1	2	1	5	1050
设备 E	2	4	2	2	2	4	1360
设备 F	1	4	1	4	3	4	1240
产品利润	30	45	24	26	24	30	
固定费用	35	20	60	70	75	30	

解　设第 i 种产品的生产量为 $x_i(i=1,2,\cdots,6)$，第 i 种产品的每件利润为 $c_i(i=1,2,\cdots,6)$，第 i 种产品的固定费用为 $d_i(i=1,2,\cdots,6)$；6 种设备分别编号为 $1,2,\cdots,6$；6 种设备的总台时分别记作 $b_i(i=1,2,\cdots,6)$，生产第 j 种产品使用第 i 中设备的台时数为 $a_{ij}(i=1,2,\cdots,9;\ j=1,2,\cdots,9)$。

引进 0－1 变量

$$y_j=\begin{cases}1, & \text{生产第}j\text{种产品},\\ 0, & \text{不生产第}j\text{种产品}.\end{cases}$$

建立如下的数学规划模型

$$\sum_{j=1}^{6}(c_jx_j-d_jy_j),$$

$$\text{s.t.}\begin{cases}\sum_{j=1}^{6}a_{ij}x_j\le b_i, & i=1,2,\cdots,6,\\ x_j\le 400y_j, & i=1,2,\cdots,6,\\ x_j\ge 0\text{且为整数}, & j=1,2,\cdots,6,\\ y_j=0\text{或}1, & j=1,2,\cdots,6.\end{cases}$$

利用 LINGO 软件求得 $x_1=96$，$x_2=0$，$x_3=195$，$x_4=0$，$x_5=191$，$x_6=94$。最大利润为 14764。

计算的 LINGO 程序如下：

```
model:
sets:
num/1..6/:b,c,d,x,y;
link(num,num):a;
endsets
data:
b=800   1160   1780   1050   1360   1240;
c=30     45   24   26   24   30;
d=35     20   60   70   75   30;
a=1  4    0    4    2    1
4    5    3    0    1    0
0    3    8    0    1    0
2    0    1    2    1    5
2    4    2    2    2    4
1    4    1    4    3    4;
enddata
max=@sum(num:c*x-d*y);
@for(num(i):@sum(num(j):a(i,j)*x(j))<b(i));
@for(num:x<400*y;@gin(x);@bin(y));
end
```

例 1.21（分段线性函数）已知分段线性函数过点 $(0,22),(5,10),(12,41),(20,49)$，图形见图 1.11，计算 $x=8.5$ 时对应的函数值 y。

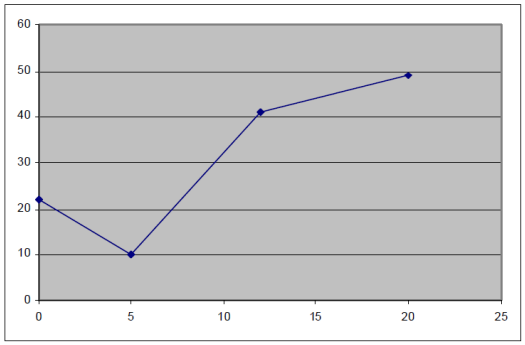


图 1.11　分段线性函数的图形

解　我们容易写出分段线性函数，然后代入 $x=8.5$ 计算即可，下面直接用 LINGO 软件计算，求得 $x=8.5$ 时，对应的函数值为 $y=25.5$。

我们先解释一下计算原理。

记分段线性函数为 $y(x)$。x 轴上的分点为 $x_1=0$，$x_2=5$，$x_3=12$，$x_4=20$，4 个分点对应的函数值分别为 y_1,y_2,y_3,y_4。当 x 属于第 1 个小区间 $[x_1,x_2]$ 时，记 $x=w_1x_1+w_2x_2$，$w_1+w_2=1$，$w_1,w_2\geq 0$，因为 $y(x)$ 在 $[x_1,x_2]$ 上是线性的，所以 $y(x)=w_1y_1+w_2y_2$。同样，当 x 属于第 2 个小区间 $[x_2,x_3]$ 时，$x=w_2x_2+w_3x_3$，$w_2+w_3=1$，$w_2,w_3\geq 0$，$y(x)=w_2y_2+w_3y_3$。当 x 属于第 3 个小区间 $[x_3,x_4]$ 时，$x=w_3b_3+w_4b_4$，$w_3+w_4=1$，$w_3,w_4\geq 0$，$y(x)=w_3y_3+w_4y_4$。

计算的 LINGO 程序如下：

```
model:
sets:
points/1..4/: w, u, v;
endsets
data:
v = 22 10 41 49; !四个点的函数值;
u = 0 5 12 20;   !四个点的自变量的取值;
enddata
x = 8.5; !自变量的赋值;
y= @sum(points(i): v(i) * w(i)); !计算对应的函数值;
x= @sum(points(i): u(i) * w(i));   !x 用权重的线性组合表示;
@sum(points( i): w(i))= 1;   !权重之和为 1;
@for(points(i): @sos2('sos2_set', w( i))); !权重是 sos2 类型，最多有两个相邻的权重非零;
end
```

注 1.7：当不使用 LINGO 的 SOS2 函数时，为了表示分段线性函数，我们就需要引进一组 0－1 变量。引入 0－1 变量 $z_k(k=1,2,3)$ 表示 x 在第几个小区间上，其中

$$z_k=\begin{cases}1, & x\text{在第}k\text{个小区间上},\\ 0, & \text{否则}.\end{cases}$$

这样，$w_1,w_2,w_3,w_4,z_1,z_2,z_3$ 应满足

$$w_1\leq z_1,\quad w_2\leq z_1+z_2,\quad w_3\leq z_2+z_3,\quad w_4\leq z_3,$$
$$w_1+w_2+w_3+w_4=1,\quad w_k\geq 0\ (k=1,2,3,4),$$
$$z_1+z_2+z_3=1,\quad z_1,z_2,z_3=0\text{或}1.$$

计算的 LINGO 程序如下：

```
model:
sets:
points/1..4/: w, u, v, z;
endsets
data:
v = 22 10 41 49; !四个点的函数值;
```

```
u = 0 5 12 20;   !四个点的自变量的取值;
enddata
x=8.5; !自变量的赋值;
n=@size(points); !计算元素的个数;
y= @sum(points(i): v(i)*w(i)); !计算对应的函数值;
x= @sum(points(i): u(i)*w(i));   !x 用权重的线性组合表示;
@sum(points( i): w(i))= 1;   !权重之和为 1;
w(1)<=z(1); w(n)<=z(n-1); z(4)=0; !z(4)是无用的;
@for(points(i)|i#gt#1 #and# i#lt#n:w(i)<=z(i-1)+ z(i));
@sum(points(i):z(i))=1;
@for(points(i):@bin(z(i)));
end
```

例 1.22（续**例 1.4**） 已知某种商品 6 个仓库的存货量，8 个客户对该商品的需求量，单位商品运价如表 1. 1 所示。如果某个仓库要供货，供货量要大于等于 15 且小于等于 30，试确定 6 个仓库到 8 个客户的商品调运数量，使总的运输费用最小。

表 1.10　单位商品运价表

单位运价 客户 / 仓库	V1	V2	V3	V4	V5	V6	V7	V8	存货量
W1	6	2	6	7	4	2	5	9	60
W2	4	9	5	3	8	5	8	2	55
W3	5	2	1	9	7	4	3	3	51
w4	7	6	7	3	9	2	7	1	43
W5	2	3	9	5	7	2	6	5	41
W6	5	5	2	2	8	1	4	3	52
需求量	35	37	22	32	41	32	43	38	

解　设 $x_{ij}(i=1,2,\cdots 6; j=1,2,\cdots,8)$ 表示第 i 个仓库运到第 j 个客户的商品数量，c_{ij} 表示第 i 个仓库到第 j 个客户的单位运价，d_j 表示第 j 个客户的需求量，e_i 表示第 i 个仓库的存货量，建立如下的非线性规划模型

$$\min \quad \sum_{i=1}^{6}\sum_{j=1}^{8} c_{ij}x_{ij} \ ,$$

$$\text{s.t.} \begin{cases} \sum_{j=1}^{8} x_{ij} \le e_i, \quad i=1,2,\cdots,6, \\ \sum_{i=1}^{6} x_{ij} = d_j, \quad j=1,2,\cdots,8, \\ 15 \le x_{ij} \le 30 \text{或} 0, \quad i=1,2,\cdots,6;\ j=1,2,\cdots,8. \end{cases}$$

利用 LINGO 软件，求得的最优调运方案见表 1. 11，最小运输费用为 761。

表 1.11　最优运输方案

	V1	V2	V3	V4	V5	V6	V7	V8	合计
W1	0	19	0	0	26	15	0	0	60
W2	15	0	0	0	15	0	0	15	45
W3	0	0	22	0	0	0	28	0	50
w4	0	0	0	15	0	0	0	23	38
W5	20	18	0	0	0	0	0	0	38
W6	0	0	0	17	0	17	15	0	49
合计	35	37	22	32	41	32	43	38	

计算的 LINGO 程序如下：

```
model:
sets:
   warehouses/1..6/: e;
   vendors/1..8/: d;
   links(warehouses,vendors): c,x;
endsets
data:   !数据部分;
e= 60 55 51 43 41 52;   !属性值;
d=35 37 22 32 41 32 43 38;
c= 6 2 6 7 4 2 5 9
4 9 5 3 8 5 8 2
5 2 1 9 7 4 3 3
7 6 7 3 9 2 7 1
2 3 9 5 7 2 6 5
5 5 2 2 8 1 4 3;
@text()=@table(x);   !把x以表格形式输出到屏幕;
enddata
min=@sum(links(i,j): c(i,j)*x(i,j));   !目标函数;
@for(warehouses(i):@sum(vendors(j): x(i,j))<=e(i));   !约束条件;
@for(vendors(j):@sum(warehouses(i): x(i,j))=d(j));
@for(links:@semic(15,x,30));
end
```

注 1.8：当不使用 LINGO 的半连续变量函数@semic 时，我们必须引进一组 0–1 变量才能把非线性约束

$$15 \le x_{ij} \le 30 \text{或} 0, \quad i=1,2,\cdots,6;\ j=1,2,\cdots,8. \tag{1.1}$$

线性化。

引进 0–1 变量

$$y_{ij}=\begin{cases}1, & \text{第}i\text{个仓库向第}j\text{个客户供货,}\\ 0, & \text{第}i\text{个仓库不向第}j\text{个客户供货.}\end{cases}$$

把(1.1)式线性化为

$$\begin{cases}15y_{ij} \le x_{ij} \le 30, & i=1,2,\cdots,8;\ j=1,2,\cdots,6,\\ y_{ij}=0 \text{或} 1, & i=1,2,\cdots,8;\ j=1,2,\cdots,6.\end{cases} \tag{1.2}$$

计算的 LINGO 程序如下：

```
model:
sets:
   warehouses/1..6/: e;
   vendors/1..8/: d;
   links(warehouses,vendors): c,x,y;
endsets
data:   !数据部分;
e= 60 55 51 43 41 52;   !属性值;
d=35 37 22 32 41 32 43 38;
c= 6 2 6 7 4 2 5 9
4 9 5 3 8 5 8 2
5 2 1 9 7 4 3 3
7 6 7 3 9 2 7 1
2 3 9 5 7 2 6 5
5 5 2 2 8 1 4 3;
@text()=@table(x);   !把 x 以表格形式输出到屏幕;
enddata
min=@sum(links(i,j): c(i,j)*x(i,j));   !目标函数;
```

```
@for(warehouses(i):@sum(vendors(j): x(i,j))<=e(i));   !约束条件;
@for(vendors(j):@sum(warehouses(i): x(i,j))=d(j));
@for(links:15*y<x; x<=30*y; @bin(y));
end
```

1.3.6 财务会计函数

目前 LINGO 提供了两个金融函数。

1．@fpa(I,N)

返回如下情形的净现值：单位时段利率为 I，连续 N 个时段支付，每个时段支付单位费用。若每个时段支付 x 单位的费用，则净现值可用 x 乘以@fpa(I,n)算得。@fpa 的计算公式为

$$@\text{fpa}(\text{I},\text{N})=\sum_{n=1}^{\text{N}}\frac{1}{(1+\text{I})^n}=\left(1-\left(\frac{1}{1+\text{I}}\right)^{\text{N}}\right)\Big/\text{I}$$

例 1.23（贷款买房问题）贷款金额 50000 元，贷款年利率 5.31%，采取分期付款方式（每年年末还固定金额，直至还清）。问拟贷款 10 年，每年需偿还多少元？

解　设贷款的总额为 A_0 元，年利率为 r，总贷款时间为 N 年，每年的等额还款额为 x 元。设第 n 年的欠款为 A_n（$n=0,1,\cdots,N$），则有递推关系

$$A_{n+1}=(1+r)A_n-x,\quad n=0,1,\cdots,N-1.$$

于是有

$$\begin{aligned}A_1&=(1+r)A_0-x,\\A_2&=(1+r)A_1-x,\\&\cdots\cdots\cdots\cdots\cdots,\\A_N&=(1+r)A_{N-1}-x,\end{aligned}$$

可以递推地得到

$$\begin{aligned}A_N&=A_0(1+r)^N-x[(1+r)^{N-1}+(1+r)^{N-2}+\cdots+(1-r)+1]\\&=A_0(1+r)^N-x\frac{(1+r)^N-1}{r},\end{aligned}$$

因而得到贷款总额 A_0、年利率 r、总贷款时间 N 年、每年的还款额 x 的如下关系

$$A_N=A_0(1+r)^N-x\frac{(1+r)^N-1}{r}=0,\tag{1.3}$$

所以每年的还款额

$$x=\frac{A_0(1+r)^N r}{(1+r)^N-1}.\tag{1.4}$$

代入数据，计算得每年需偿还 $x=6573.069$ 元。

计算的 LINGO 程序如下：

```
A0=50000; r=0.0531; N=10;
A0=x*@fpa(r,N);   !利用 LINGO 函数计算;
x2=A0*(1+r)^N*r/((1+r)^N-1);   !利用还款额公式(1.4)计算
```

2. @fpl(I,n)

返回如下情形的净现值：单位时段利率为 I，第 n 个时段支付单位费用。@fpl(I,n)的计算公式为

$$(1+\text{I})^{-\text{n}}.$$

细心的读者可以发现这两个函数间的关系：

$$@\text{fpa}(\text{I},\text{n})=\sum_{\text{k}=1}^{\text{n}}@\text{fpl}(\text{I},\text{k}).$$

例 1.24　验证 $@\text{fpa}(0.05,10)=\sum_{\text{k}=1}^{10}@\text{fpl}(0.05,\text{k})$.

解　验证的 LINGO 程序如下：

```
sets:
num/1..10/;
endsets
r=0.05;
a=@fpa(r,10);
b=@sum(num(i):@fpl(r,i));
```

1.3.7 概率函数

1.*@NORMINV(P, MU, SIGMA)*

返回均值为 ***MU***，标准差为 ***SIGMA*** 的正态分布的分布函数反函数在 ***P*** 处的值 z_p，即若 $X \sim N(\mu,\sigma^2)$，则返回值 z_p 满足 $P\{X \le z_p\} = p$， z_p 称为随机变量 X 的 p 分位数。

2. *@NORMSINV(P)*

返回值为标准正态分布的 ***P*** 分位数。

3. *@PBN(P, N, X)*

返回值为二项分布 $B(N,P)$ 的分布函数在 X 处的取值。当 N 或 X 不是整数时，采用线性插值进行计算。

4. *@PCX(N, X)*

返回值为自由度为 N 的 $\chi^2(N)$ 分布的分布函数在 X 处的取值。

5. *@PEB(A, X)*

当到达负荷（强度）为 A，服务系统有 X 个服务器且允许无穷排队时的 Erlang 损失概率。

6. *@PEL(A, X)*

当到达负荷（强度）为 A，服务系统有 X 个服务器且不允许排队的 Erlang 损失概率。

7. *@PFD(N, D, X)*

自由度为 N 和 D 的 F 分布的分布函数在 X 点的取值。

8. *@PFS(A, X, C)*

当负荷上限为 A，顾客数为 C，并行服务器数量为 X 时，有限源的 Poisson 服务系统的等待或返修顾客数的期望值，其中极限负荷 A 是顾客数乘以平均服务时间，再除以平均返修时间。

9. *@PHG(POP, G, N, X)*

超几何分布的分布函数。也就是说，返回如下概率：当总共有 POP 个产品，其中 G 个是正品时，那么随机地从中取出 N 个产品（$N \le POP$），正品不超过 X 个的概率。当 POP, G, N 或 X 不是整数时，采用线性插值进行计算。

10. *@PPL(A, X)*

Poisson 分布的线性损失函数，即返回 $\max(0, Z - X)$ 的期望值，其中 Z 为均值为 A 的 Poisson 分布随机变量。

11. *@PPS(A, X)*

Poisson 分布函数，即返回均值为 A 的 Poisson 分布的分布函数在 X 点的取值，当 X 不是整数时，采用线性插值进行计算。

12. *@PSL(X)*

标准正态分布的线性损失函数，即返回 $\max(0, Z - X)$ 的期望值，其中 Z 为标准正态分布随机变量。

13. *@PSN(X)*

标准正态分布的分布函数在 X 点的取值。

14. *@PTD(X)*

自由度为 N 的 t 分布的分布函数在 X 点的取值。

15. *@QRAND(SEED)*

产生服从(0, 1)区间的多个拟均匀随机数，其中 *seed* 为种子，默认时取当前计算机时间为种子。该函数只允许在模型的数据部分使用，它将用拟随机数填满集属性。通常，声明一个 $m \times n$ 的二维表，m 表示运行实验的次数，n 表示每次实验所需的随机数的个数。在行内，

随机数是独立分布的；在行间，随机数是非常均匀的。这些随机数是用“分层取样”的方法产生的。

例 1.25　产生 (0,1) 区间上 4×2 个拟均匀分布的随机数。

解　产生随机数的 LINGO 程序如下：

```
model:
data:
   M=4; N=2; seed=1234567;
enddata
sets:
   rows/1..M/;
   cols/1..N/;
   table(rows,cols): x;
endsets
data:
   x=@qrand(seed);
enddata
end
```

16. ***@RAND(SEED)***

返回 0 与 1 之间的一个伪均匀随机数，其中 seed 为种子。

为了说明怎样产生服从任意给定分布的随机数，我们引进一个定理。

定理 1.1　若随机变量 ξ 的概率密度函数和分布函数分别为 $f(x), F(x)$，则随机变量 $\eta = F(\xi)$ 的分布就是区间 $[0,1]$ 上的均匀分布。因此，若 R_i 是 $[0,1]$ 中均匀分布的随机数，那么方程 $\int_{-\infty}^{x_i} f(x)dx = R_i$ 的解 x_i 就是所求的具有概率密度函数为 $f(x)$ 的随机数。

证明　ξ 的分布函数

$$F(x) = \int_{-\infty}^{x} f(x)dx,$$

不失一般性，设 $F(x)$ 是严格单调增函数，存在反函数 $x = F^{-1}(y)$。下面我们证明随机变量 $\eta = F(\xi)$ 服从 $[0,1]$ 上的均匀分布，记 η 的分布函数为 $G(y)$，由于 $F(x)$ 是分布函数，它的取值在 $[0,1]$ 上，从而当 $0 < y < 1$ 时

$$G(y) = P\{\eta \le y\} = P\{F(\xi) \le y\} = P\{\xi \le F^{-1}(y)\} = F(F^{-1}(y)) = y,$$

因而 η 的分布函数为

$$G(y) = \begin{cases} 0, & y \le 0, \\ y, & 0 < y < 1, \\ 1, & y \ge 1. \end{cases}$$

η 服从 $[0,1]$ 上的均匀分布。

R 为 $[0,1]$ 区间均匀分布的随机变量，则根据定义，随机变量 $\xi = F^{-1}(R)$ 的分布函数为 $F(x)$，分布密度为 $f(x)$，这里 $F^{-1}(x)$ 是 $F(x)$ 的反函数。

所以，只要分布函数 $F(x)$ 的反函数 $F^{-1}(x)$ 存在，由 $[0,1]$ 区间均匀分布的随机数 R_t，求 $x_t = F^{-1}(R_t)$，即解方程

$$F(x_t) = R_t,$$

就可得到分布函数为 $F(x)$ 的随机数 x_t。

例 1.26　利用@rand 产生 15 个服从均值为−10、标准差为 2 的正态分布的随机数，15 个自由度为 2 的 t 分布的随机数。

```
model:   !产生一列正态分布和t分布的随机数;
sets:
   series/1..15/: u, znorm, zt;
endsets
   u(1)=@rand(0.1234);   !第一个均匀分布随机数是任意的;
```

```
@for(series(I)|I #GT# 1:u(I)=@rand(u(I-1))); !产生其余的均匀分布的随机数;
@for(series( I):
znorm(I)=@norminv(u(I),-10,2); !正态分布随机数;
@ptd(2,zt(I))=u(I);  !和自由度为2的t分布随机数;
@free(znorm(I)); @free(zt(I))); !ZNORM 和ZT可以是负数;
end
```

1.3.8 输入输出函数

这些函数用于控制输入数据和输出结果，包括以下五个函数。

1. @FILE (filename)

当前模型引用其他 ASCII 码文件中的数据或文本时可以采用该语句，其中 filename 为存放数据的文件名（可以带有文件路径，没有指定路径时表示在当前目录），该文件中记录之间用“~”分开。我们将在第二章详细介绍。

2. @ODBC

这个函数提供 LINGO 与 ODBC（open data base connection，开放式数据库连接）的接口。

3. @OLE

这个函数提供 LINGO 与 OLE（object linking and embedding，对象链接与嵌入）的接口。

4. @POINTER（N）

在 Windows 下使用 LINGO 的动态链接库（dynamic link library，DLL），直接从共享的内存中传送数据。

5. @TEXT（filename）

用于数据段中将解答结果保存到文本文件 filename 中，当省略 filename 时，结果送到标准的输出设备（通常就是屏幕）。filename 中可以带有文件路径，没有指定路径时表示在当前目录下生成这个文件。

1.3.9 结果报告函数

1.@DUAL (variable_or_row_name)

当参数为变量名时，返回解答中变量的 Reduced Cost，即变量的检验数；当参数是行名时，返回该约束行的 Dual Price，即影子价格。

2.@ITERS ()

这个函数在程序的数据段（data）和计算段（calc）使用，调用时不需要任何参数，总是返回 LINGO 求解器计算所使用的总迭代次数。例如：

```
@text()=@ITERS ()
```

将迭代次数显示在屏幕上。

3. @NEWLINE (*n*)

这个函数在输出设备上输出 n 个新行（n 为一个正整数）。

4. @WRITE(obj1[, …, objn])

这个函数只能在数据段中使用，用于输出一系列结果（obj1,…，objn）到一个文件，或电子表格（如 Excel）、或数据库，这取决于@write 所在的输出语句中左边的定位函数。例如：

```
data:
@text()=@write('The ratio of x to y is: ',x/y);
enddata
```

其中 x, y 是该模型中的变量，若计算结束时 x=10，y=5，则上面语句的作用是在屏幕上输出：

```
The ratio of x to y is: 2
```

5. @WRITEFOR(setname[(set_index_list) [| cond_qualifier]]: obj1[, …, objn])

这个函数只能在数据段和计算段中使用，它可以看作是函数@write 在循环情况下的推广，它输出集合上定义的属性对应的多个变量的取值。

```
data:
    @text()=@writefor(links(i,j)|volume(i,j)#gt#0:'从仓库',i,'运输',volume(i,j),
    '件货物到顾客',j,@newline(1));
enddata
```

对应的输出效果示意如下：

从仓库1运输19件货物到顾客2
从仓库1运输41件货物到顾客5
从仓库2运输1件货物到顾客1
从仓库2运输32件货物到顾客4
从仓库3运输11件货物到顾客2
从仓库3运输40件货物到顾客7
从仓库4运输5件货物到顾客6
从仓库4运输38件货物到顾客8
从仓库5运输34件货物到顾客1
从仓库5运输7件货物到顾客2
从仓库6运输22件货物到顾客3
从仓库6运输27件货物到顾客6
从仓库6运输3件货物到顾客7

6. 符号“*”

在@write 和@writefor 函数中，可以使用符号“*”表示将一个字符串重复多次，用法是将“*”放在一个正整数 n 和这个字符串之间，表示将这个字符串重复 n 次。

7. **@FORMAT (*value, format_descriptor*)**

@format 函数用在@write 和@writefor 语句中，作用是格式化数值和字符串的值以文本形式输出。Value 是要格式化输出的数值和字符串的值，***format_descriptor*** 与 C 语言的输出格式是一样的。

```
@text()=@write( '仓库     顾客          数量',@newline(1));
@text()=@write( '-------------------------',@newline(1));
@text()=@writefor(links(i,j)|x(i,j)#gt# 0:3*' ', i, 4*' ', j, 4*' ',
@format( x( i, j), '8.0f'),@newline( 1));
```

对应的输出效果对应如下：

```
仓库     顾客          数量
-------------------------
   1     2            19
   1     5            41
   2     1             1
……
```

例 1.27　画出正态分布函数图形的示意图。

```
DATA:
H=49; ！ 图形的高度;
W=56; ！ 图形的宽度;
R=2.4; ！ 区间半径;
ENDDATA
SETS:
S1/1..H/: X, FX;
ENDSETS
@FOR(S1(I): @FREE(X); ! X取值可以是非负的;
X(I)=-R + (I-1)*2*R/(H-1); ！ 计算X坐标;
FX(I)=@PSN(X(I))); ！ 计算Y坐标=@psn(X);
DATA:
@TEXT()=@WRITE('Graph of @PSN() on the interval [-',R,',+',R,']:',@NEWLINE(1)); !输出图的头部;
@TEXT()=@WRITE('| 0 ',(W/2-5)*'-',' 0.5 ',(W/2-5)*'-', '1.0 X(i)',@NEWLINE(1));
@TEXT()=@WRITEFOR( S1(I): '|',(W * FX(I)+1/2) *'*',
@IF(X(I)#LT# 0,',' '),(W-(W*FX(I)+1/2)+3)*' ',@FORMAT(X(I), '.1f'),@NEWLINE(1));
!图的中间循环输出部分;
@TEXT()=@WRITE('|0',(W/2-5)*'-','0.5',(W/2-5)*'-','1.0',@NEWLINE(1)); !输出图的尾部;
ENDDATA
```

8. **@NAME (var_or_row_reference)**

@name 以文本方式返回变量名或行名，@name 只能在数据段（data）和计算段（calc）中使用。

例 1.28　@name 函数应用示例。

```
model:
sets:
   warehouses/W1..W6/: e;
   vendors/V1..V8/: d;
   links(warehouses,vendors): c,x;
endsets
data:   !数据部分;
e= 60 55 51 43 41 52;   !属性值;
d=35 37 22 32 41 32 43 38;
c= 6 2 6 7 4 2 5 9
4 9 5 3 8 5 8 2
5 2 1 9 7 4 3 3
7 6 7 3 9 2 7 1
2 3 9 5 7 2 6 5
5 5 2 2 8 1 4 3;
@text()=@writefor(links(i,j)|x(i,j)#gt#0:@name(x),'     ',x,@newline(1));
enddata
min=@sum(links(i,j): c(i,j)*x(i,j));   !目标函数;
@for(warehouses(i):@sum(vendors(j): x(i,j))<=e(i));   !约束条件;
@for(vendors(j):@sum(warehouses(i): x(i,j))=d(j));
end
```

输出结果如下所示：

```
X( W1, V2)     19
X( W1, V5)     41
X( W2, V1)     1
X( W2, V4)     32
X( W3, V2)     11
X( W3, V7)     40
X( W4, V6)     5
X( W4, V8)     38
X( W5, V1)     34
X( W5, V2)     7
X( W6, V3)     22
X( W6, V6)     27
X( W6, V7)     3
```

9. **@RANGED (variable_or_row_name)**

为了保持最优基不变，目标函数中变量的系数或约束行的右端项允许减少的量。

10. **@RANGEU (variable_or_row_name)**

为了保持最优基不变，目标函数中变量的系数或约束行的右端项允许增加的量。

11. **@STATUS ()**

返回 LINGO 求解模型结束后的最后状态。返回值的含义见表 1.12。

表 1.12　@STATUS ()返回值的含义

返回值	含义
0	Global Optimum（全局最优解）
1	Infeasible（没有可行解）
2	Unbounded（目标函数无界）
3	Undetermined（不确定，求解失败）
4	Feasible（可行解）

5	Infeasible or Unbounded（不可行或无界）
6	Local Optimum（局部最优解）
7	Locally Infeasible（局部不可行）
8	Cutoff（目标函数达到了指定的误差水平）
9	Numeric Error（约束中遇到了无定义的数学操作）

12. **@STRLEN(string)**

返回字符串的长度。

13. **@TABLE('attr|set')**

以表格形式显示属性值或集成员的值。

14. **@TIME()**

返回的是生成模型和求解模型所用的全部运行时间，单位为秒。

1.3.10 其他函数

1. @IF (logical_condition, true_result, false_result)

当逻辑表达式 logical_condition 的结果为真时，返回 true_result，否则返回 false_result.

例如，生产甲乙两种产品，数量分别为 x 和 y 。生产甲产品的固定成本为 100，每件产品的变动成本为 2。生产乙产品的固定成本为 60，每件产品的变动成本为 3。至少生产甲乙两种产品 30 件，如何生产才能使总成本最小。

LINGO 模型如下：

```
MIN=COST;
COST=XCOST+YCOST;
XCOST=@IF(X#GT#0,100,0)+2*X;
YCOST= @IF(Y#GT#0,60,0)+3*Y;
X+Y>=30;
```

有经验的建模者知道，如果不适用 LINGO 的@IF 函数，需要引进 0-1 变量把上述模型进行线性化。而使用了 LINGO 函数@IF，LINGO 可以把上述模型线性化。

2. @WARN ('text', logical_condition)

如果逻辑表达式 logical_condition 的结果为真，显示'text'信息。

3. @USER (user_determined_arguments)

该函数允许用户调用自己编写的 DLL 文件或对象文件，涉及到其他应用的接口，这里就不介绍了。

1.4 LINGO 子模型

1.4.1 子模型的定义和求解

在 LINGO 9.0 及更早的版本中，在每个LINGO 模型窗口中只允许有一个优化模型，可以称为主模型（MAIN MODEL）。在LINGO 10.0及以后的版本中，每个LINGO 模型窗口中除了主模型外，用户还可以定义子模型(SUBMODEL)。子模型可以在主模型的计算段中被调用，这就进一步增强了LINGO的编程能力。

子模型必须包含在主模型之内，即必须位于以“MODEL:”开头、以“END”结束的模块内。同一个主模型中，允许定义多个子模型，所以每个子模型本身必须命名，其基本语法是：

SUBMODEL *mymodel*:

可执行语句（约束+目标函数）;

ENDSUBMODEL

其中*mymodel*是该子模型的名字，可执行语句一般是一些约束语句，也可能包含目标函数，但不可以有自身单独的集合段、数据段、初始段和计算段。也就是说，同一个主模型内的变量都是全局变量，这些变量对主模型和所有子模型同样有效。

如果已经定义了子模型 *mymodel*，则在计算段中可以用语句“@SOLVE(*mymodel*);”求解这个子模型。

同一个LINGO主模型中，允许定义多个子模型。

例 1.29 用 LINGO 求下列方程组

$$\begin{cases} x^2 + y^2 = 4, \\ x^2 - y^2 = 1. \end{cases}$$

的所有解。

```
model:
submodel maincon:   !定义方程子模型;
x^2+y^2=4;
x^2-y^2=1;
endsubmodel
submodel con1:   !定义附加约束子模型;
@free(x);x<0;
endsubmodel
submodel con2:   !定义附加约束子模型;
@free(y); y<0;
endsubmodel
submodel con3:   !定义附加约束子模型;
@free(x); @free(y);
x<0; y<0;
endsubmodel
calc:
@solve(maincon);   !调用子模型;
@solve(maincon,con1);
@solve(maincon,con2);
@solve(maincon,con3);
endcalc
end
```

求上述 LINGO 模型时，需要把求解设置为全局求解器。依次调用 4 个子模块，求得的方程组的解依次为

$x=1.581139, y=1.224745$； $x=-1.581139, y=1.224745$；

$x=1.581139, y=-1.224745$； $x=-1.581139, y=-1.224745$.

例 1.30　分别求解以下 4 个优化问题：

（1）在满足约束 $x^2+4y^2\le 1$ 且 x, y 非负的条件下，求 $x-y$ 的最大值；

（2）在满足约束 $x^2+4y^2\le 1$ 且 x, y 非负的条件下，求 $x+y$ 的最小值；

（3）在满足约束 $x^2+4y^2\le 1$ 且 x, y 可取任何实数的条件下，求 $x-y$ 的最大值；

（4）在满足约束 $x^2+4y^2\le 1$ 且 x, y 可取任何实数的条件下，求 $x+y$ 的最小值。

可以编写如下 LINGO 程序：

```
model:
submodel obj1:
max=x-y;
endsubmodel
submodel obj2:
min=x+y;
endsubmodel
submodel con1:
x^2+4*y^2<=1;
endsubmodel
submodel con2:
@free(x); @free(y);
endsubmodel
calc:
@write('问题1的解：', @newline(1)); @solve(obj1,con1);
@write('问题2的解：', @newline(1)); @solve(obj2,con1);
@write('问题3的解：', @newline(1)); @solve(obj1,con1,con2);
@write('问题4的解：', @newline(1)); @solve(obj2,con1,con2);
endcalc
end
```

这个程序中定义了4个子模型，其中OBJ1和OBJ2只有目标（没有约束），而CON1和CON2

只有约束（没有目标）。在计算段，我们将它们进行不同的组合，分别得到针对问题（1）~（4）的优化模型进行求解。但需要注意，每个@solve命令所带的参数表中的子模型是先合并后求解的，所以用户必须确保每个@solve命令所带的参数表中的子模型合并后是合理的优化模型，例如最多只能有一个目标函数。

运行程序后，求得

问题（1）的解，$x=1, y=4.92523\times10^{-9}$，目标函数的最大值为 1；

问题（2）的解，$x=0, y=0$，目标函数的最小值为 2.403504×10^{-10}；

问题（3）的解，$x=0.8944272, y=-0.2236068$，目标函数的最大值为 1.118034；

问题（4）的解，$x=-0.8944272, y=-0.2236068$，目标函数的最小值为−1.118034。

例 1.31　当参数 $a=0,1,2,3,4$；$b=2,4,6,7$ 时，分别求下列的非线性规划问题。

$$\min \quad 4x_1^3-ax_1-2x_2,$$

$$\text{s.t.}\begin{cases} x_1+x_2\le 4,\\ 2x_1+x_2\le 5,\\ -x_1+bx_2\ge 2,\\ x_1,x_2\ge 0. \end{cases}$$

解　a 的取值有 5 种可能，b 的取值有 4 种可能，(a,b) 的取值总共有 20 种组合，这就需要求解 20 个非线性规划问题。利用 LINGO 的子模块功能，只要编写一个 LINGO 程序就可以解决问题。计算的 LINGO 程序如下：

```
model:
sets:
var1/1..5/:a0;   !a0用于存放a的取值;
var2/1..4/:b0;   !b0用于存放b的取值;
var3/1 2/:x;
endsets
data:
a0=0 1 2 3 4;
b0=2 4 6 7;
enddata
submodel sub_obj:   !定义目标函数子模型;
[obj] min=4*x(1)^3-a*x(1)-2*x(2);   !为了下面引用目标函数的值，这里对目标函数进行了标号;
endsubmodel
submodel sub_con:   !定义约束条件子模型;
x(1)+x(2)<4;
2*x(1)+x(2)<5;
-x(1)+b*x(2)>2;
endsubmodel
calc:
@for(var1(i):@for(var2(j):a=a0(i);b=b0(j);@solve(sub_obj,sub_con);   !调用目标函数和约束条件子模型，即求解数学规划;
@write('a=',a0(i),',b=',b0(j),'时，最优解x1=',x(1),',x2=',x(2),',最优值为
',obj,'.',@newline(2))));   !输出最优解和最优值;
endcalc
end
```

1.4.2　求背包问题的多个最好解的例子

例 1.32（续例 1. 18）求背包问题的多个最好解。

对于例 1.18 的最优问题，最优解并不是唯一的。如果我们希望找到所有的最优解（最优值为 38 的所有解），有没有办法呢？更一般地，能否找出前 K 个最好的解？供决策者选择。

例 1.18 的程序中只有主模型，我们也可以将这个模型定义为子模型，然后在计算段 calc 中进行求解，相应的 LINGO 程序如下：

```
model:
sets:
items/1..8/:x,w,v;
endsets
data:
w=3,4,6,7,9,10,11,12;
v=4,6,7,9,11,12,13,15;
c=30;
enddata
submodel knapsack:
max=@sum(items:v*x); !目标函数;
@sum(items:w*x)<=c;   !重量约束;
@for(items:@bin(x)); !0-1变量约束;
endsubmodel
calc:
@solve(knapsack);
endcalc
end
```

求解上述LINGO模型，得到的结果与例1.18不用子模型时相同。

为了得到第2个最好的解，我们需要再次求解子模型knapsack，但必须排除再次找到刚刚得到的解x(2)=x(4)=x(5)=x(6)=1（其他x(i)=0）。因此，我们需要在第2次求解子模型knapsack时，增加一些约束条件（一般称为“割”）。生成“割”的方法可能有很多种，这里我们介绍一种针对0-1变量的特殊处理方法。

对于我们刚刚得到的解x(2)=x(4)=x(5)=x(6)=1（其他x(i)=0），显然满足

$$x(1)-x(2)+x(3)-x(4)-x(5)-x(6)+x(7)+x(8)=-4;$$

这个等式左边就是将刚刚得到的解中取1的x(i)的系数定义为-1，取0的x(i)的系数定义为1，然后求代数和；等式右边就是解中取1的x(i)的个数的相反数。

为了防止再次求解子模型knapsack时这个解再次出现，就是要防止x(2)，x(4)，x(5)，x(6)同时取1的情况出现。下面的约束就可以保证做到这一点：

$$x(1)-x(2)+x(3)-x(4)-x(5)-x(6)+x(7)+x(8)>=-3;$$

这个约束就是将上面等式中的右端项增加了 1，将等号“=”改成了“>=”。显然，这个约束排除了 x(2)，x(4)，x(5)，x(6)同时取 1 的情况，由于 x(i)只能取 0 或 1，这个约束除了排除 x(2)，x(4)，x(5)，x(6)同时取 1 的情况外，没有对原可行解空间增加任何新的限制。

可以想象，增加这个约束后，新的最优解一定与 x(2)=x(4)=x(5)=x(6)=1（其他 x(i)=0）不同。这种处理方法具有一般性，可以用于找出背包问题的前 N 个最好解。

具体的程序如下（以下程序中取 N=7）：

```
model:
data:
N=7;
```

```
enddata
sets:
items/1..8/:x,w,v;
soln/1..N/:rhs;  !rhs表示根据每个最优解生成"割"时的右端项;
links(soln,items):cof;  !割的系数,即1或-1;
endsets
data:
w=3,4,6,7,9,10,11,12;
v=4,6,7,9,11,12,13,15;
c=30;
enddata
submodel knapsack:
[obj]max=@sum(items:v*x); !目标函数;
@sum(items:w*x)<=c;  !重量约束;
@for(items:@bin(x)); !0-1变量约束;
@for(soln(k)|k# lt#ksofar:@sum(items(j):cof(k,j)*x(j))>=rhs(k));  !排除已经得到的解;
endsubmodel
calc:
@for(soln(ks):ksofar=ks; @solve(knapsack);  rhs(ks)=1;  !对ks=1,2,...,K进行循环,
ksofar表示当前正在计算的是第几个最优解;
@write('      ',ks,'      ',@format(obj,'3.0f'),':');
@writefor(items(j):'      ',x(j));  @write(@newline(1));
@for(items(j):@ifc(x(j)#gt#0.5:cof(ks,j)=-1;rhs(ks)=rhs(ks)-1; @else cof(ks,j)=1)));
endcalc
end
```

运行上述程序好后，求得的前 7 个最好解如下：

```
1     38:    0    1    0    1    1    1    0    0
2     38:    1    1    1    1    0    1    0    0
3     38:    1    1    0    0    0    0    1    1
4     37:    1    1    0    0    0    1    0    1
5     37:    0    1    1    1    0    0    0    1
6     37:    1    1    1    1    1    0    0    0
7     37:    0    1    1    0    1    0    1    0
```

可见，前 7 个最好的解中，最优值为 obj=38 的解一共有 3 个，而 obj=37 的解至少有 4 个。

习题 1

1.1　用 LINGO 求解下列线性规划：

（1） $\max z = 6x_1 + 2x_2 + 3x_3 + 9x_4$，

$$\text{s.t.}\begin{cases}5x_1+6x_2-4x_3-4x_4\le 2,\\ 3x_1-3x_2+2x_3+8x_4\le 25,\\ 4x_1+2x_2-x_3+3x_4\le 10,\\ x_i\ge 0,\ i=1,2,3,4.\end{cases}$$

（2）$\min\ z=10x_1+2x_2+x_3+8x_4+6x_5$，

$$\text{s.t.}\begin{cases}x_1+x_3=100,\\ x_2+x_4=200,\\ x_3+x_5=300,\\ x_4+x_5=500,\\ x_1+2x_2+x_3+x_4-x_5\ge -400,\\ 2x_1+3x_4+4x_5\ge -220.\end{cases}$$

（3）$\max\ z=8x_1+6x_2+5x_3+9x_4+3x_5$，

$$\text{s.t.}\begin{cases}2x_1+9x_2-x_3-3x_4-x_5\le 20,\\ x_1-3x_2+2x_3+6x_4+x_5\le 30,\\ x_1+2x_2-x_3+x_4-2x_5\le 10,\\ a_i\le x_i\le b_i,\ i=1,2,3,4,5.\end{cases}$$

其中 $a=[a_1,a_2,a_3,a_4,a_5]=-[10,50,15,20,30]$，$b=[b_1,b_2,b_3,b_4,b_5]=[20,50,60,30,10]$。

1.2　用 LINGO 求方程组

$$\begin{cases}x^2+y^2=2,\\ 2x^2+x+y^2+y=4.\end{cases}$$

的所有实数解。

1.3　求解下列非线性规划：

（1）$\max\ z=\sum_{i=1}^{100}\sqrt{x_i}$，

$$\text{s.t.}\begin{cases}x_1\le 10,\\ x_1+2x_2\le 20,\\ x_1+2x_2+3x_3\le 30,\\ x_1+2x_2+3x_3+4x_4\le 40,\\ \sum_{i=1}^{100}(101-i)x_i\le 1000,\\ x_i\ge 0,\ i=1,2,\cdots,100.\end{cases}$$

（2）$\max\ z=(x_1-1)^2+\sum_{i=1}^{99}(x_i-x_{i+1})^2$，

$$\text{s.t.}\begin{cases}x_1+\sum_{i=2}^{100}x_i^2=200,\\ \sum_{i=1}^{50}x_{2i}^2-\sum_{i=1}^{50}x_{2i-1}^2=10,\\ \left(\sum_{i=1}^{33}x_{3i}\right)\left(\sum_{i=1}^{50}x_{2i}\right)\le 1000,\\ -5\le x_i\le 5,\ i=1,2,\cdots,100.\end{cases}$$

1.4　已知 $\boldsymbol{A}=[\boldsymbol{A}_1,\boldsymbol{A}_2,\cdots,\boldsymbol{A}_6]=\begin{bmatrix}89.39 & 86.25 & 108.13 & 106.38 & 62.40 & 47.19\\ 64.3 & 99 & 99.6 & 96 & 96.2 & 79.9\end{bmatrix}$，

$$\boldsymbol{B}=[\boldsymbol{B}_1,\boldsymbol{B}_2,\cdots,\boldsymbol{B}_6]=\begin{bmatrix}25.2 & 28.2 & 29.4 & 26.4 & 27.2 & 25.2\\ 223 & 287 & 317 & 291 & 295 & 222\end{bmatrix}.$$

利用 LINGO 的子模块，对于 $j_0=1,2,\cdots,6$，求解如下 6 个线性规划问题：

$$\max \boldsymbol{B}_{j_0}^T\boldsymbol{Y}\ ,$$

$$\text{s.t.}\ \begin{cases}\boldsymbol{A}_j^T\boldsymbol{X}-\boldsymbol{B}_j^TY\geq 0, \quad j=1,2,\cdots,6,\\ \boldsymbol{A}_{j_0}^T\boldsymbol{X}=1,\\ \boldsymbol{X}\geq 0, Y\geq 0.\end{cases}$$

其中 $\boldsymbol{X}=[x_1,x_2]^T$，$\boldsymbol{X}\geq 0$ 表示 $x_1,x_2\geq 0$；$\boldsymbol{Y}=[y_1,y_2]^T$，$\boldsymbol{Y}\geq 0$ 表示 $y_1,y_2\geq 0$.