



Lingo 编程应用

Lingo Programming Application



主讲人：刘瑞华

时 间：2023.06.08

其他函数 (1)

(1) @warn('文字信息', 逻辑表达式)

- 如果逻辑表达式的值为真，则显示指导文字信息，一般用于提示；

(2) @if(逻辑表达式, 表达式为真时的值, 表达式为假时的值)

- 该函数根据逻辑表达式的结果是真还是假，决定返回值，常用来表示分段函数；



其他函数 (2-1)

例3.4 生产计划安排问题. 某企业用 A, B 两种原油混合加工成甲、乙两种成品油销售。

数据见下表，表中百分比是成品油中原油 A 的最低含量。

成品油甲和乙的销售价与加工费之差分别为 5 和 5.6 （单位：千元/吨），原油 A , B 的采购价分别是采购了 x （单位：吨）的分段函数 $f(x)$ 和 $g(x)$ （单位：千元/吨），该企业的现有资金限额为 7200 千元，生产成品油乙的最大能力为 2000 吨。假设成品油全部能销售出去，试在充分利用现有资金和现有库存的条件下，合理安排采购和生产计划，使企业的收益最大。

$$f(x) = \begin{cases} 4x, & 0 \leq x \leq 500, \\ 500 + 3x, & 500 < x \leq 1000, \\ 1500 + 2x, & x > 1000. \end{cases} \quad g(x) = \begin{cases} 3.2x, & 0 \leq x \leq 400, \\ 240 + 2.6x, & 400 < x \leq 800, \\ 880 + 1.8x, & x > 800. \end{cases}$$

	甲	乙	现有库存	最大采购量
A	≥50%	≤60%	500	1650
B			800	1200

其他函数 (2-2)

解：

- 原油 A, B 的采购量分别为 x, y ;
- 原油 A 用于生产成品油甲、乙的数量分别为： x_{11}, x_{12} ;
- 原油 B 用于生产成品油甲、乙的数量分别为： x_{21}, x_{22} ;
- 采购原油 A, B 的费用分别为 $f(x), g(y)$;
- 目标函数：收益最大
- 约束条件：
 - 采购量约束： $x \leq 1650, y \leq 1200$;
 - 生产能力约束： $x_{12} + x_{22} \leq 2000$;
 - 原油含量约束： $x_{11} \geq x_{21}, x_{12} \geq 1.25x_{22}$;
 - 成品油与原油的关系： $x_{11} + x_{12} \leq x + 500, x_{21} + x_{22} \leq y + 800$;
- 资金约束： $f(x) + g(y) \leq 7200$;

规划模型：

$$\max z = 5(x_{11} + x_{21}) + 5.6(x_{12} + x_{22}) - f(x) - g(y),$$

$$s.t. \begin{cases} x_{11} + x_{12} \leq x + 500, \\ x_{21} + x_{22} \leq y + 800, \\ x_{12} + x_{22} \leq 2000, \\ x_{11} \geq x_{21}, x_{12} \geq 1.25x_{22}, \\ x \leq 1650, y \leq 1200, \\ f(x) + g(y) \leq 7200. \end{cases}$$

其他函数 (2-3)

model:

```
max = 5 * ( x11 + x21 ) + 5.6 * ( x12 + x22 ) - f -  
g;
```

```
x11 + x12 <= x + 500;  
x21 + x22 <= y + 800;  
x12 + x22 <= 2000;  
x11 >= x21;  
x12 > 1.5 * x22;  
x <= 1650;  
y <= 1200;  
f + g <= 7200;
```

```
f = @if( ( x #LE# 500 ), 4 * x, @if( ( x #le# 1000 ),  
    500 + 3 * x, 1500 + 2 * x ) );  
g = @if( ( y #le# 400 ), 3.2 * y, @if( ( y #le#  
    800 ),  
    240 + 2.6 * y, 880 + 1.8 * y ) );
```

end

Global optimal solution found.

Objective value:	13000.00
Objective bound:	13000.00

Infeasibilities:	0.000000
Extended solver steps:	4
Total solver iterations:	39
Elapsed runtime seconds:	0.08

Model Class:	MILP
--------------	------

Total variables:	40
Nonlinear variables:	0
Integer variables:	24

Total constraints:	71
Nonlinear constraints:	0

Total nonzeros:	188
Nonlinear nonzeros:	0

Linearization components added:

Constraints:	60
Variables:	32
Integers:	24

Variable	Value	Reduced Cost
X11	900.0000	0.000000
X21	900.0000	0.000000
X12	1200.000	0.000000
X22	800.0000	0.000000
F	4700.000	0.000000
G	2500.000	0.000000
X	1600.000	0.000000
Y	900.0000	0.000000

其他常用函数 (3-1)

◆ @file 函数

- 调用外部文件中的数据，可放在任何地方；
- 语法格式为：@file('filename');

◆ 例 (new 3.3) : 具体就见例 2.1

- 结束标记 (~) 之间的数据文部分称为**记录**；
- 如果数据文件没有结束标记，则整个文件被看作单个记录；
- 模型第一次调用 @file 函数时，lingo 打开数据文件，读取第一个记录；依次类推。

```
MODEL:
  SETS:
    WH / @file( 'new 3.3.txt' ) / : AI;
    VD / @file( 'new 3.3.txt' ) / : DJ;
    LINKS( WH, VD ): C, X;
  ENDSETS

  DATA:
    AI = @file( 'new 3.3.txt' );
    DJ = @file( 'new 3.3.txt' );
    C = @file( 'new 3.3.txt' );
  ENDDATA

  min = @SUM( LINKS( i, j ) : C( i, j ) * X( i, j ) );

  @FOR( WH( i ) :
    @SUM( VD( j ) : X( i, j ) ) <= AI( i ) );
  @FOR( VD( j ) :
    @SUM( WH( i ) : X( i, j ) ) = DJ( j ) );

END
```

其他常用函数 (3-2)

◆ @text 函数, @ole 函数

- @text 函数是输出操作, 输出 txt 文件;
- @ole 函数时从 excel 中输入、输出数据;

◆ 例 (高级 3.3) : 具体就见例 2.1

- 需要在 excel 里面定义 range;
- range 是自左而右, 自上而下来读;
- range 中定义的名字与 lingo 中定义的变量要求一致。

```
MODEL:
  SETS:
    WH / W1 .. W6 / : AI;
    VD / V1 .. V8 / : DJ;
    LINKS( WH, VD ): wv, X;
  ENDSETS

  DATA:
    ai,dj,wv = @ole( '运输模型 3.1.xlsx' );
  ENDDATA

  min = @SUM( LINKS( i, j ) : wv( i, j ) * X( i, j ) );

  @FOR( WH( i ) :
    @SUM( VD( j ) : X( i, j ) ) <= AI( i ) );
  @FOR( VD( j ) :
    @SUM( WH( i ) : X( i, j ) ) = DJ( j ) );

  data:
    @text('ys.txt') = x;
    @ole( '运输模型 3.1.xlsx' ) = x;
  enddata
END
```

稠密集合与稀疏集合 (1)

- ◆ **稠密集合**: 衍生集合, 包括初始集合的所有可能的配对。如,

sets:

wh / w1 .. w6 /: ai;

vd / v1 .. v8 /: dj;

links(wh, vd) : c, x;

endsets

- links 为衍生集合, 其成员包括 48 个, 为稠密集合。

- ◆ **稀疏集合**: 衍生集合, 其成员只是稠密集合中的一部分 (子集) 。

定义方法:

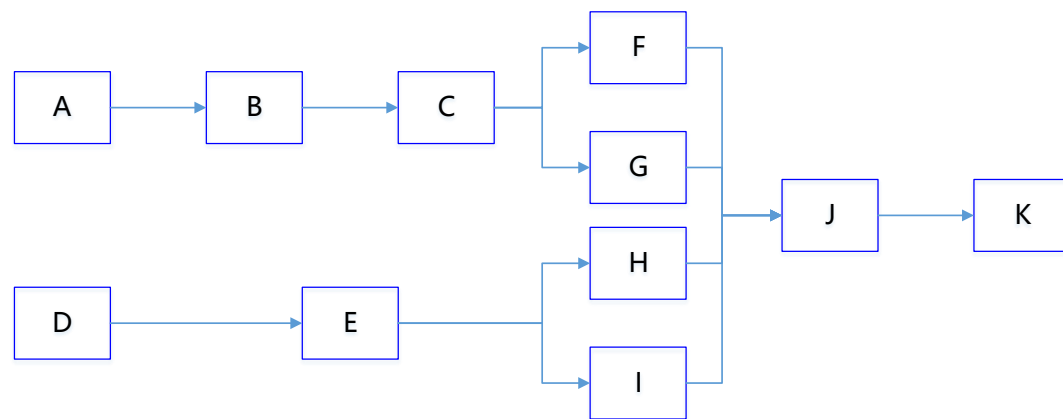
- (1) 直接列表法;
- (2) 元素过滤法

稠密集合与稀疏集合—直接列举法 (2-1)

例3.5 计划排序模型. 研究生产工序的计划和安排, 通过合理安排工序在某些设备上加工的先后次序, 使得完成任务的总时间最省。完成某产品的装配过程需要做 11 项任务 (用 A ~ K 表示), 每项任务所花费的时间如下表。

各项任务之间的先后顺序如图, 由某装配流水线来完成上述产品的装配, 该流水线上按顺序有四个工作站 (即班组1-4), 对于有先后次序的任务, 只能在流水线上向后传 (如任务 B 和 C 的次序是先 B 后 C, 如果 B 安排给第 3 站做, 则 C 要么由 B 自己完成, 要么传给第 4 站做, 不能倒回去给第 2 站, 或第 1 站做), 每项任务只能且必须分配至一个工作站来做, 不适当的分配会产生瓶颈——有较少任务的班组将被迫等待其前面分配了较多任务的班组。请在保证满足任务间的所有优先关系的条件下合理分配, (在流水作业处于稳定状态后) 使装配周期最短。

任务	A	B	C	D	E	F	G	H	I	J	K
时间	45	11	9	50	15	12	12	12	12	8	9



稠密集合与稀疏集合—直接列举法 (2-2)

解： x_{ik} : 表示任务 i ($i = A, B, \dots, K$) 分配给工作站 k ($k = 1, 2, 3, 4$) 的情况；

✓ $x_{ik} = 1$: 表示分配； $x_{ik} = 0$: 表示不分配；

➤ t_i : 表示完成各项任务所需时间；

➤ 目标函数： $\min T = \max_{1 \leq k \leq 4} \sum_{i=1}^{11} t_i \cdot x_{ik}$ ；

➤ 约束条件 (1)：每项任务只能，且必须分配至一个工作站来做，可表示为：

✓ $\sum_{k=1}^4 x_{ik} = 1, i = 1, 2, \dots, 11$ ；

➤ 约束条件 (2)：各项任务间如果有优先关系，则排在前面的任务 i 对应的工作站 (序号) 应小于 (或等于) 排在后面的任务 j 所对应的工作站，即对所有有顺序的任务，

✓ $\sum_{k=1}^4 (k \cdot x_{jk} - k \cdot x_{ik}) \geq 0, \text{ if } i < j$ ；

➤ 约束条件 (3)： $x_{ij} = 0$ 或 1；

➤ 这是一个非线性规划问题，但可以转化为线性规划问题，增加一个变量 Z ，再增加一个

约束条件： $\sum_{i=1}^{11} t_i \cdot x_{ik} \leq Z$ ；

➤ 目标函数变为： $\min Z$ ；

稠密集合与稀疏集合—直接列举法 (2-3)

```
model:
  sets:
    task / A, B, C, D, E, F, G, H, I, J, K /: t;
    pred( task, task ) / A,B, B,C, C,F, C,G, F,J, G,J,
                          J,K, D,E, E,H, E,I, H,J, I,J /;

    station / 1 .. 4 /;
    txs( task, station ) : x;
  endsets
  data:
    t = 45, 11, 9, 50, 15, 12, 12, 12, 12, 8, 9;
  enddata
  @for( task( i ) : @sum( station( k ) : x( i, k ) ) = 1 );
  @for( pred( i, j ) | ( i #t# j ): @sum( station( k ) :
    k * x( j, k ) - k * x( i, k ) ) >= 0 );
  @for( station( k ) : @sum( task( i ) : t( i ) * x( i, k ) )
    <= Z );
  min = Z;
  @for( txs( i, j ) : @bin( x( i, j ) ) );
end
```

注：lingo 18会执行错误，因为超出变量范围；
只能用 lingo 11 执行。

```
Global optimal solution found.
Objective value:                    50.00000
Objective bound:                    50.00000
Infeasibilities:                    0.000000
Extended solver steps:              0
Total solver iterations:            207
```

		Variable	Value	Re
		Z	50.00000	
		T(A)	45.00000	
		T(B)	11.00000	
		T(C)	9.000000	
		T(D)	50.00000	
		T(E)	15.00000	
		T(F)	12.00000	
		T(G)	12.00000	
		T(H)	12.00000	
		T(I)	12.00000	
		T(J)	8.000000	
		T(K)	9.000000	
		X(A, 1)	1.000000	
		X(A, 2)	0.000000	
		X(A, 3)	0.000000	
		X(A, 4)	0.000000	
		X(B, 1)	0.000000	
		X(B, 2)	0.000000	
		X(B, 3)	1.000000	
		X(B, 4)	0.000000	
		X(C, 1)	0.000000	
		X(C, 2)	0.000000	
		X(C, 3)	0.000000	
		X(C, 4)	1.000000	
		X(D, 1)	0.000000	
		X(D, 2)	1.000000	
		X(D, 3)	0.000000	
		X(D, 4)	0.000000	

X(E, 1)	0.000000
X(E, 2)	0.000000
X(E, 3)	1.000000
X(E, 4)	0.000000
X(F, 1)	0.000000
X(F, 2)	0.000000
X(F, 3)	0.000000
X(F, 4)	1.000000
X(G, 1)	0.000000
X(G, 2)	0.000000
X(G, 3)	0.000000
X(G, 4)	1.000000
X(H, 1)	0.000000
X(H, 2)	0.000000
X(H, 3)	1.000000
X(H, 4)	0.000000
X(I, 1)	0.000000
X(I, 2)	0.000000
X(I, 3)	1.000000
X(I, 4)	0.000000
X(J, 1)	0.000000
X(J, 2)	0.000000
X(J, 3)	0.000000
X(J, 4)	1.000000
X(K, 1)	0.000000
X(K, 2)	0.000000
X(K, 3)	0.000000
X(K, 4)	1.000000

稠密集合与稀疏集合—元素过滤法 (3-1)

例3.6 配对模型. 某公司准备将 8 个职员安排到 4 个办公室, 每室 2 人。根据已往观察, 已知某些职员在一起时合作好, 有些则不然, 下表列出了两两之间的相容程度, 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?

➤ 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。

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
s_1	—	9	3	4	2	1	5	6
s_2	—	—	1	7	3	5	2	1
s_3	—	—	—	4	4	2	9	2
s_4	—	—	—	—	1	5	5	2
s_5	—	—	—	—	—	8	7	6
s_6	—	—	—	—	—	—	2	3
s_7	—	—	—	—	—	—	—	4
s_8	—	—	—	—	—	—	—	—

稠密集合与稀疏集合—元素过滤法 (3-2)

解：

➤ X ：决策矩阵，其元素 X_{ij} 的取值为 0，或 1，

✓ $X_{ij} = 1$ ：表示 i 与 j 组合；

✓ $X_{ij} = 0$ ：表示 i 不与 j 组合；

➤ C ：表示上表中的相容程度；

➤ 目标函数：组合的总相容程度数字最小最好；

➤ 约束条件：

✓ 每人组合一次，即对于职员 i ，必有

$$\sum_{j=i \text{ 或 } k=i} X_{jk} = 1;$$

0-1 规划模型如下：

$$\min \sum_{i < j} C_{ij} \cdot X_{ij}$$

$$s.t. \begin{cases} \sum_{j=i \text{ 或 } k=i} X_{jk} = 1, i = 1, 2, \dots, 8, \\ X_{ij} = 0, \text{ 或 } 1, i < j. \end{cases}$$

稠密集合与稀疏集合—元素过滤法 (3-3)

```

model:
  sets:
    ren / 1 .. 8 /;
    links( ren, ren ) | &2 #gt# &1 : 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;

  enddata

  min = @sum( links( i, j ) | ( i #lt# j ) : C( i, j ) * X( i, j ) );
  @for( ren( i ) : @sum( links( j, k ) |
    ( ( j #eq# i ) #or# ( k #eq# i ) ) : X( j, k ) ) = 1 );
  @for( links( i, j ) | ( i #lt# j ) : @bin( X( i, j ) );

end
  
```

& 为占位符

```

Global optimal solution found.
Objective value:                6.000000
Objective bound:                6.000000
Infeasibilities:                0.000000
Extended solver steps:          0
Total solver iterations:        0
Elapsed runtime seconds:        0.53

Model Class:                    PILP
  
```

Total variables: 28

X(1, 2)	0.000000	9.000000	0
X(1, 3)	0.000000	3.000000	28
X(1, 4)	0.000000	4.000000	
X(1, 5)	0.000000	2.000000	9
X(1, 6)	1.000000	1.000000	0
X(1, 7)	0.000000	5.000000	
X(1, 8)	0.000000	6.000000	84
X(2, 3)	0.000000	1.000000	0
X(2, 4)	0.000000	7.000000	
X(2, 5)	0.000000	3.000000	
X(2, 6)	0.000000	5.000000	
X(2, 7)	1.000000	2.000000	
X(2, 8)	0.000000	1.000000	
X(3, 4)	0.000000	4.000000	
X(3, 5)	0.000000	4.000000	
X(3, 6)	0.000000	2.000000	
X(3, 7)	0.000000	9.000000	
X(3, 8)	1.000000	2.000000	
X(4, 5)	1.000000	1.000000	
X(4, 6)	0.000000	5.000000	
X(4, 7)	0.000000	5.000000	
X(4, 8)	0.000000	2.000000	
X(5, 6)	0.000000	8.000000	
X(5, 7)	0.000000	7.000000	
X(5, 8)	0.000000	6.000000	
X(6, 7)	0.000000	2.000000	
X(6, 8)	0.000000	3.000000	
X(7, 8)	0.000000	4.000000	

数据部分赋值

- ◆ 如果想给某个属性的部分成员赋值，而让另一些成员为变量，可以用如下方式：

```
sets:  
    wh / w1 .. w6 /: ai;  
endsets  
  
data:  
    ai = 60, 55, , , ;  
enddata
```

- ◆ 注：以上程序，属性 ai 的前两个成员分别为 60 和 55，后面的四个逗号表示省略四个成员的值，即他们可以自由取值。

初始化段 (1-1)

◆ 数据段赋值:

- 该变量在整个程序运行阶段都是常量, 其值保持不变;
- 该变量不是决策变量;

◆ 决策变量赋值:

- 该变量是决策变量;
- 该初始赋值作为寻找最优解的起始值;
- 该变量本身不是常量;

init:

$x = 0.99;$

$y = 0.01;$

endinit

$y \leq \text{@log}(x);$

$x^2 + y^2 \leq 1;$

- **注:** 初始化段只对非线性模型起作用, 在线性模型中不起任何作用。

初始化段 (1-2)

不赋初始值

```
Feasible solution found.
Infeasibilities:                  0.2900625E-05
Extended solver steps:           5
Best multistart solution found at step: 1
Total solver iterations:         76
Elapsed runtime seconds:         0.25

Model Class:                     NLP

Total variables:                 2
Nonlinear variables:             2
Integer variables:              0

Total constraints:              2
Nonlinear constraints:          2

Total nonzeros:                 4
Nonlinear nonzeros:            3
```

Variable	Value
Y	0.1620993E-05
X	1.000001

Row	Slack or Surplus
1	-0.1706843E-06
2	-0.2900625E-05

赋初始值

```
Feasible solution found.
Infeasibilities:                  0.1517370E-07
Extended solver steps:           5
Best multistart solution found at step: 1
Total solver iterations:         51
Elapsed runtime seconds:         0.31

Model Class:                     NLP

Total variables:                 2
Nonlinear variables:             2
Integer variables:              0

Total constraints:              2
Nonlinear constraints:          2

Total nonzeros:                 4
Nonlinear nonzeros:            3
```

Variable	Value
X	1.000000
Y	0.000000

Row	Slack or Surplus
1	0.7586851E-08
2	-0.1517370E-07

模型的标题

◆ title:

- 定义一个模型的标题;
- 从 title 到分号之间的文本就是模型的标题;
- 文本可以是英文, 也可以中文;

```
model:
  title 赋初始值;
  init:
    x = 0.99;
    y = 0.01;
  endinit

  y <= @log( x );
  x ^ 2 + y ^ 2 <= 1;
end
```

```
Feasible solution found.
Infeasibilities:                                0.1517370E-07
Extended solver steps:                          5
Best multistart solution found at step:         1
Total solver iterations:                        51
Elapsed runtime seconds:                       0.25

Model Class:                                    NLP

Total variables:                               2
Nonlinear variables:                           2
Integer variables:                             0

Total constraints:                             2
Nonlinear constraints:                         2

Total nonzeros:                                4
Nonlinear nonzeros:                            3

Model Title: 赋初始值

Variable      Value
      X      1.000000
      Y      0.000000

Row  Slack or Surplus
  1      0.7586851E-08
  2     -0.1517370E-07
```

PART
FIVE

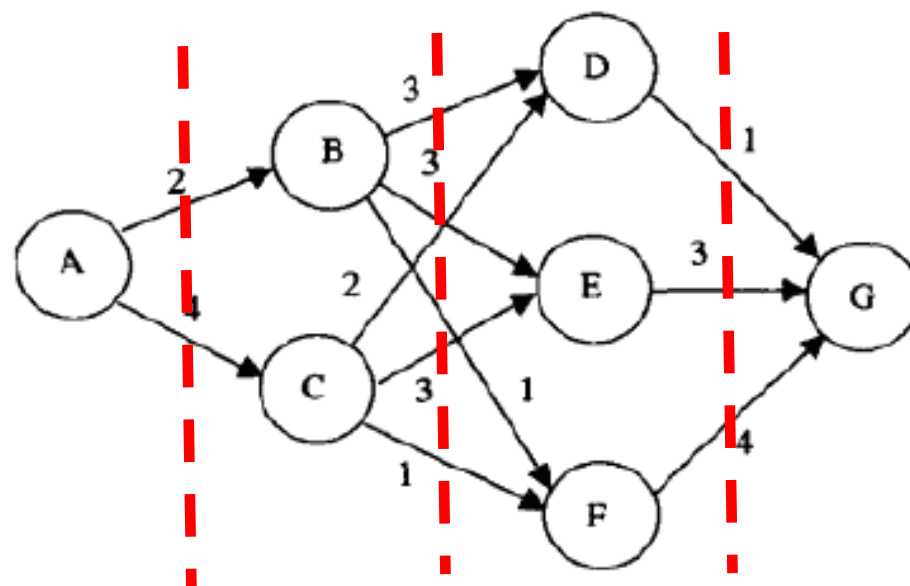
典型 问题

最短路问题 (1-1)

➤ 动态规划法：

给定 n 个点 p_i ($i=1, 2, \dots, n$) 组成集合 $\{p_i\}$ ，集合中任一点 p_i 到另一点 p_j 的距离用 w_{ij} 表示。如果 p_i 到 p_j 没有弧联结(无通路)，则规定 $w_{ij} = \pm\infty$ ，又规定 $w_{ii} = 0$ ，指定一个终点 p_N ，要求从 p_i 点出发到 p_N 的最短路线。

◆ 例5.1：右图中 A, B, ..., G 表示 7 个城市，连线表示城市之间有路相通，连线旁的数字表示路的长度 w_{ij} ，要从城市 A 到 G 找出一条最短路线。



最短路问题 (1-2)

- 该问题有 3 个阶段，第一阶段：从 A 到 B 或 C；第二阶段：到 D, E, 或 F；第三阶段：到终点 G，而我们从终点向前倒过来找最短路径。
- 第三阶段：从 D, E, F 到 G 的最短路分别为 1, 3, 4，记为 $f(D) = 1, f(E) = 3, f(F) = 4$ ；
- 第二阶段：与 D, E, F 有连线的出发点为 B 和 C，从 B 出发分别经过 D, E, F，到终点 G 的里程分别为：

$$w_{CD} + f(D) = 2 + 1 = 3, w_{CE} + f(E) = 3 + 3 = 6, w_{CF} + f(F) = 1 + 4 = 5.$$

故 B 到 G 的最短路是上述三者的最小值，可以写成 $f(B) = \min \{ w_{Bj} + f(j) \}$,

同理，C 到 G 的最短路，也可以写成 $f(C) = \min \{ w_{Cj} + f(j) \}$.

- 第一阶段：与第二阶段相同。

最短路问题 (1-3)

上述算法可以简写成：

$$\begin{cases} f(i) = \min_j \{w_{ij} + f(j)\}, i = n-1, \dots, 2, 1, \\ f(n) = 0. \end{cases}$$

式中 n 是终点, 1 是起点, 终点的 $f(n) = 0$, 逐步向起点推算, j 是与 i 相邻, 上一步考察过, 且与终点相通, $f(j)$ 为已知的点。

以上基本做法是把问题分成为多个阶段, 一个一个阶段地考虑问题, 将一个复杂问题简单化, 这就是解动态规划的基本思想。

最短路问题 (1-4)

```
model:
  sets:
    cities / A, B, C, D, E, F G /: f;
    roads( cities, cities ) / A,B A,C B,D B,E B,F C,D
    C,E C,F D,G E,G F,G /: W, P;
  endsets

  data:
    w = 2, 4, 3, 3, 2, 2, 3, 1, 1, 3, 4;
  enddata

  N = @size( cities );
  f( N ) = 0;

  @for( cities( i ) | i #lt# N :
    f( i ) = @min( roads( i, j ) : w( i, j ) + f( j ) ) );
  @for( roads( i, j ) : p( i, j ) =
    @if( f( i ) #eq# w( i, j ) + f( j ), 1, 0 ) );
end
```

Variable	Value
N	7.000000
F(A)	6.000000
F(B)	4.000000
F(C)	3.000000
F(D)	1.000000
F(E)	3.000000
F(F)	4.000000
F(G)	0.000000
W(A, B)	2.000000
W(A, C)	4.000000
W(B, D)	3.000000
W(B, E)	3.000000
W(B, F)	2.000000
W(C, D)	2.000000
W(C, E)	3.000000
W(C, F)	1.000000
W(D, G)	1.000000
W(E, G)	3.000000
W(F, G)	4.000000
P(A, B)	1.000000
P(A, C)	0.000000
P(B, D)	1.000000
P(B, E)	0.000000
P(B, F)	0.000000
P(C, D)	1.000000
P(C, E)	0.000000
P(C, F)	0.000000
P(D, G)	1.000000
P(E, G)	1.000000
P(F, G)	1.000000

旅行售货商 (TSP) 问题 (2-1)

◆ **旅行售货商问题** (又称货郎担问题, TSP 模型) 是运筹学的一个著名命题。

其模型: 有一个旅行推销商, 从某个城市出发, 要遍访若干城市各一次, 且仅一次, 最后返回原出发城市 (成能到每个城市一次, 且仅一次的路线为一个巡回)。已知从城市 i 到 j 的旅费为 C_{ij} , 问如何安排旅行路线使总旅费最少?

- TSP 模型是 NP-完全问题;
- 最优解很难得到, 就算得到近似解也不容易;
- 目前算法: 构造性算法、改进型算法、二边逐次修正法的改进型算法、神经网络、模拟退化法、弹性网法, 这些算法的解法都有一定难度;
- TSP 模型转化为混合整数规划, 优点: 程序简洁、计算速度快、适用范围广;

◆ TSP 模型的数学描述

➤ x_{ij} : 表示路线从 i 到 j 是否可行,

取值为 0, 1 ;

✓ $x_{ij} = 1$: 表示从 i 到 j ;

✓ $x_{ij} = 0$: 表示不走 i 到 j ;

TSP 模型可表示为 :

$$\begin{aligned} \min & \sum_{i=1}^n \sum_{j=1}^n c_{ij} \cdot x_{ij}, \\ \text{s.t.} & \begin{cases} \sum_{j=1}^n x_{ij} = 1, i = 1, 2, \dots, n, j \neq i, \\ \sum_{i=1}^n x_{ij} = 1, j = 1, 2, \dots, n, i \neq j, \\ x_{ij} = 0, 1, i, j = 1, 2, \dots, n, \\ \text{不含子巡回.} \end{cases} \end{aligned}$$

若仅考虑前三个约束条件,则是类似于指派问题的模型。对于 TSP 模型只是必要条件,并不是充分条件。

“不含子巡回”的约束条件,如何用数学表达式实现该条件?

◆ 方法:增加变量 $u_i (i = 1, 2, \dots, n)$, 并附加约束条件:

$$u_i - u_j + nx_{ij} \leq n-1, u_i, u_j \geq 0,$$

$$i = 1, 2, \dots, n, j = 2, 3, \dots, n, \text{ 且 } i \neq j.$$

- ✓ 任何子巡回的路线都必然不满足该约束条件, 不论 u_i 如何取值;
- ✓ 全部不含子巡回的整体巡回路线都可以满足该约束条件, 只要 u_i 取适当的值。

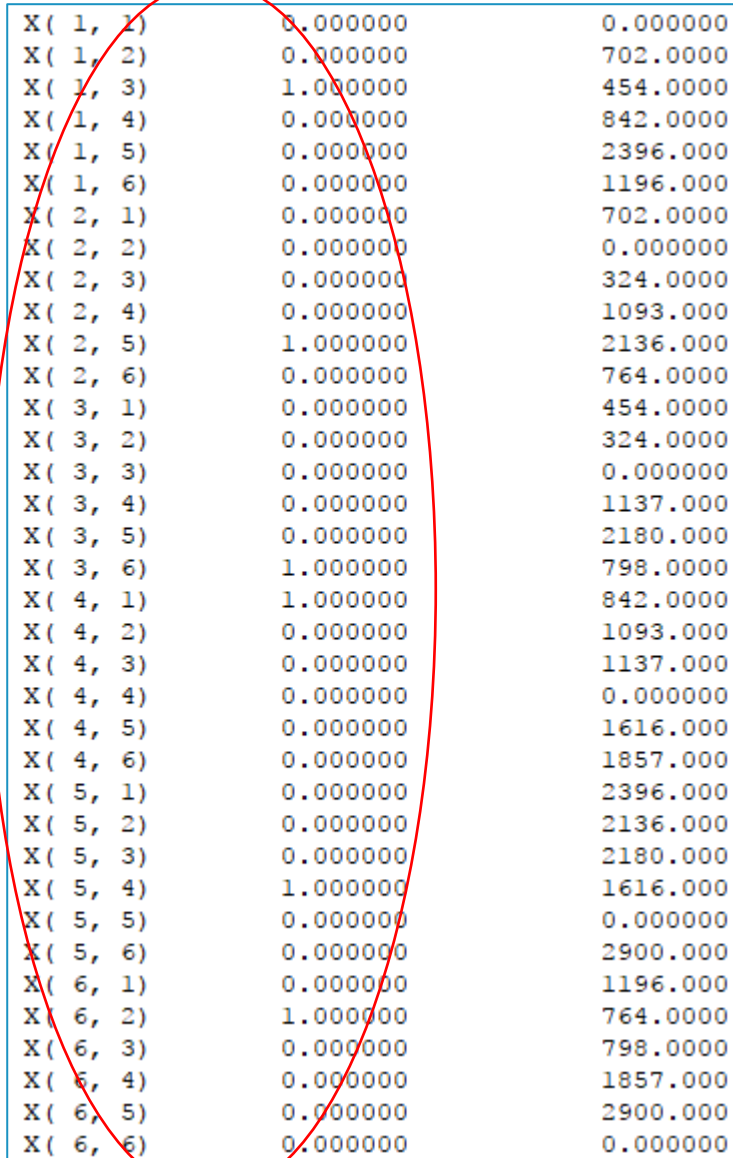
旅行售货商 (TSP) 问题 (2-3)

例 5.2 已知 6 个城市之间的距离矩阵, 求从 1 出发回到 1 的 TSP 路线。

	城市1	城市2	城市3	城市4	城市5	城市6
城市1	0	702	454	842	2396	1196
城市2	702	0	324	1093	2136	764
城市3	454	324	0	1137	2180	798
城市4	842	1093	1137	0	1616	1857
城市5	2396	2136	2180	1616	0	2900
城市6	1196	764	798	1857	2900	0

旅行售货商 (TSP) 问题 (2-4)

```
model:
  sets:
    city / 1 .. 6 /: u;
    link( city, city ) : c, x;
  endsets
  data:
    c = 0, 702, 454, 842, 2396, 1196,
        702, 0, 324, 1093, 2136, 764,
        454, 324, 0, 1137, 2180, 798,
        842, 1093, 1137, 0, 1616, 1857,
        2396, 2136, 2180, 1616, 0, 2900,
        1196, 764, 798, 1857, 2900, 0;
  enddata
  n = @size( city );
  min = @sum( link( i, j ) : c( i, j ) * x( i, j ) );
  @for( city( i ) : @sum( city( j ) | j #ne# i : x( i, j ) ) = 1 );
  @for( city( j ) : @sum( city( i ) | i #ne# j : x( i, j ) ) = 1 );
  @for( link( i, j ) : @bin( x( i, j ) ) );
  @for( city( i ) : @for( city( j ) | j #gt# 1 #and# i #ne# j :
      u( i ) - u( j ) + n * x( i, j ) <= n - 1 );
end
```



X(1, 1)	0.000000	0.000000
X(1, 2)	0.000000	702.0000
X(1, 3)	1.000000	454.0000
X(1, 4)	0.000000	842.0000
X(1, 5)	0.000000	2396.0000
X(1, 6)	0.000000	1196.0000
X(2, 1)	0.000000	702.0000
X(2, 2)	0.000000	0.000000
X(2, 3)	0.000000	324.0000
X(2, 4)	0.000000	1093.0000
X(2, 5)	1.000000	2136.0000
X(2, 6)	0.000000	764.0000
X(3, 1)	0.000000	454.0000
X(3, 2)	0.000000	324.0000
X(3, 3)	0.000000	0.000000
X(3, 4)	0.000000	1137.0000
X(3, 5)	0.000000	2180.0000
X(3, 6)	1.000000	798.0000
X(4, 1)	1.000000	842.0000
X(4, 2)	0.000000	1093.0000
X(4, 3)	0.000000	1137.0000
X(4, 4)	0.000000	0.000000
X(4, 5)	0.000000	1616.0000
X(4, 6)	0.000000	1857.0000
X(5, 1)	0.000000	2396.0000
X(5, 2)	0.000000	2136.0000
X(5, 3)	0.000000	2180.0000
X(5, 4)	1.000000	1616.0000
X(5, 5)	0.000000	0.000000
X(5, 6)	0.000000	2900.0000
X(6, 1)	0.000000	1196.0000
X(6, 2)	1.000000	764.0000
X(6, 3)	0.000000	798.0000
X(6, 4)	0.000000	1857.0000
X(6, 5)	0.000000	2900.0000
X(6, 6)	0.000000	0.000000

最优连线问题 (3-1)

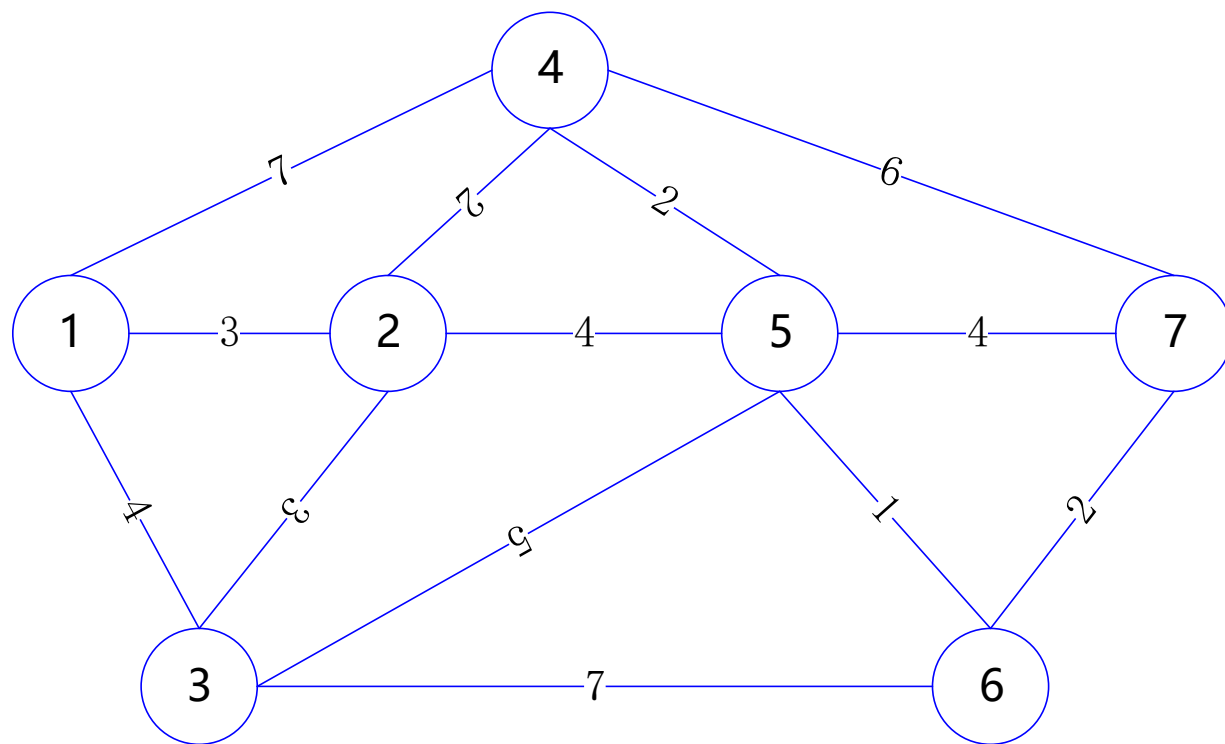
◆ 最优连线问题，最小生成树问题（MST 问题）。

树是图论中的一种简单而重要的图，连通并且无圈的无向图称为树，具有 n 个顶点的无向连通图是树的充要条件是它有 $n - 1$ 条边。如果图的边有权（对应于边的实数），则权的总和达到最小的生成树称为最小生成树，最小生成树不一定唯一。

- 最小生成树是网络优化中的一个重要问题；
- 如：修筑城镇公路；架设通讯网络；修筑水渠等；
- 方法：破圈法、避圈法（Kruskal 算法）、Dijkstra 算法等，仅限于手工操作，程序设计较复杂；
- 此处将 MST 问题转化为整数规划，优点为：程序简洁、计算速度快、适用范围广。

最优连线问题 (3-2)

例5.2 假设某电力公司计划在 7 个村庄之间架设电线，各村庄之间的距离如下图。试求出使电线总长度最小的架线方案。



解：节点 1 表示树根；

- c_{ij} ：表示点 i 到点 j 的距离；
 - ✓ 当两个节点之间没有线路相通时，两点之间的距离用 M （很大的实数）来表示；
- x_{ij} ：取值为 0, 1；
 - ✓ $x_{ij} = 1$ ：表示从 i 到 j 的边在树中， $i \neq j$ ；
 - ✓ $x_{ij} = 0$ ：表示从 i 到 j 的边不在树中；

◆ **目标函数：**

$$\min z = \sum_{i=1}^n \sum_{j=1}^n c_{ij} \cdot x_{ij},$$

◆ **约束条件：**

- (1) 除树根外的每个接点都有线路通到，且只需要一次，写为，

$$\sum_{i=1}^n x_{ij} = 1, j = 2, 3, \dots, n, i \neq j,$$

- (2) 至少有一条线路从节点 1 出来，

$$\sum_{j=2}^n x_{1j} \geq 1,$$

以上约束条件为必要条件，但不充分，需增加一个变量 u ，再附加约束条件：

- (3) 限制 u_j 的取值范围：

$$u_1 = 0, 1 \leq u_j \leq n-1, j = 2, 3, \dots, n,$$

用以下不等式能达到目的：

$$u_j \geq 1, \text{ 且 } u_j \leq n-1-(n-2) \cdot x_{1j}, j > 1,$$

- (4) 如果线路从 j 到 k ，则

$$u_k = u_j + 1, \text{ 为了避免形成圈以及重}$$

复路线，约束条件为：

$$u_j \geq u_k + x_{kj} - (n-2) \cdot (1-x_{kj}) + (n-3) \cdot x_{jk}, \\ 1 \leq k \leq n, 2 \leq j \leq n, j \neq k.$$

最优连线问题 (3-4)

model:

title = 最小生成树;

sets:

city / 1 .. 7 /: u;

link(city, city) : c, x;

endsets

data:

c = 0, 3, 4, 7, 100, 100, 100,
3, 0, 3, 2, 4, 100, 100,
4, 3, 0, 100, 5, 7, 100,
7, 2, 100, 0, 2, 100, 6,
100, 4, 5, 2, 0, 1, 4,
100, 100, 7, 100, 1, 0, 2,
100, 100, 100, 6, 4, 2, 0;

enddata

n = @size(city);

min = @sum(link : c * x);

@for(city(j) | j #gt# 1 :

@sum(city(i) | i #ne# j : x(i, j)) = 1);

@sum(city(j) | j #gt# 1 : x(1, j)) >= 1;

@for(city(j) | j #gt# 1 :

u(j) >= 1; u(j) <= n - 1 - (n - 2) * x(1, j));

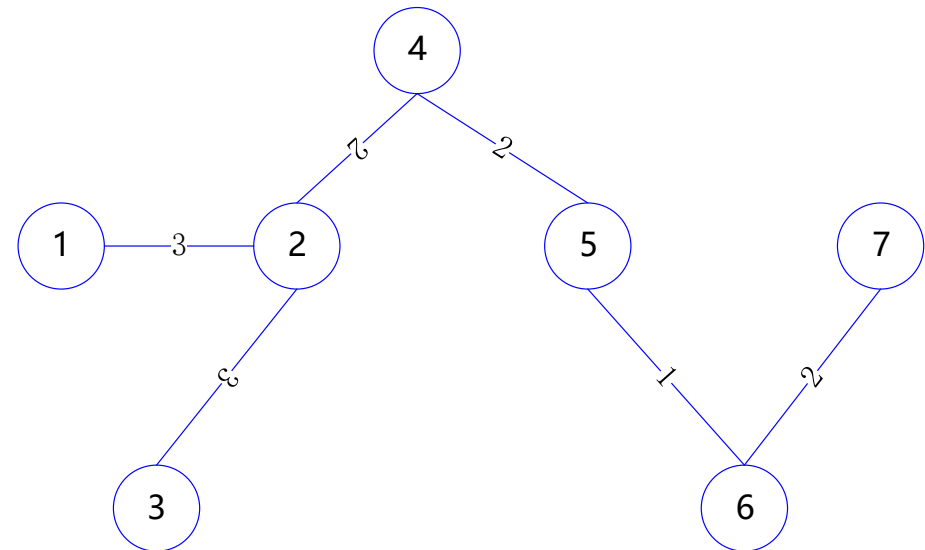
@for(city(k) :

@for(city(j) | j #gt# 1 #and# j #ne# k :

u(j) >= u(k) + x(k, j) - (n - 2)
* (1 - x(k, j)) + (n - 3) * x(j, k));

@for(link : @bin(x));

end



THANKS