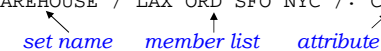


Sets in LINGO

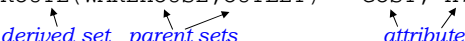
Dennis Bricker
 Dept of Mechanical & Industrial Engineering
 The University of Iowa

LINGO sets can be

- *primitive*, by specifying
 - name of the set
 - member list
 - attributes, if any

example: WAREHOUSE / LAX ORD SFO NYC / : CAPACITY


- *derived* (formed from one or more other sets) using
 - selection of a subset
 - Cartesian product (cross product)

example: WAREHOUSE / LAX ORD SFO NYC / : CAPACITY;
 OUTLET / 1..6 / : DEMAND;
 ROUTE (WAREHOUSE, OUTLET) : COST, X;


LINGO, as do other modeling languages (e.g. MPL, AMPL, GAMS), allows you to group similar objects together in *sets*.

A single statement in LINGO can then apply to all members of the set.

Each member of a set may have one or more *attributes*, which can be known quantities or values to be determined by the optimization.

DATA section

- assigns values to some of the attributes
- isolates data from rest of the model

example:

```
DATA :
CAPACITY = 300 450 400 625;
DEMAND = 225 310 290;
COST = 2.10 1.85 2.75 2.05 1.70 1.95
       1.45 1.75 2.30 2.05 1.55 1.60
       1.20 1.90 1.65 2.10 1.35 1.80;
ENDDATA
```

SET LOOPING functions

<i>Function</i>	<i>purpose</i>
@FOR	generates constraints over members of set
@SUM	sum of expression over members of set
@MIN	minimum of expression over members of set
@MAX	maximum of expression over members of set

example:

```
@FOR(OUTLET(J):  
    @SUM(WAREHOUSE(I): X(I,J) ) >= DEMAND(J) );
```

```
! The objective;  
    MIN = @SUM( LINKS( I, J): COST( I, J) * VOLUME( I, J));  
! The demand constraints;  
    @FOR( VENDORS( J):  
        @SUM( WAREHOUSES( I): VOLUME( I, J)) = DEMAND( J));  
! The capacity constraints;  
    @FOR( WAREHOUSES( I):  
        @SUM( VENDORS( J): VOLUME( I, J)) <= CAPACITY( I));  
END
```

Example: A 6-Warehouse & 8-Vendor Transportation Problem

SETS:

```
WAREHOUSES / WH1 WH2 WH3 WH4 WH5 WH6/: CAPACITY;  
VENDORS / V1 V2 V3 V4 V5 V6 V7 V8/ : DEMAND;  
LINKS( WAREHOUSES, VENDORS): COST, VOLUME;  
ENDSETS
```

DATA:

```
CAPACITY = 60 55 51 43 41 52;  
DEMAND = 35 37 22 32 41 32 43 38;  
COST = 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
```

Example: An integer LP problem

SETS:

```
PLANES/ ROCKET, METEOR, STREAK, COMET, JET, BIPLANE /:  
    PROFIT, SETUP, QUANTITY, BUILD;  
RESOURCES /STEEL, COPPER, PLASTIC, RUBBER, GLASS, PAINT/:  
    AVAILABLE;  
RXP( RESOURCES, PLANES): USAGE; ! derived set  
ENDSETS
```

DATA:

```
PROFIT = 30 45 24 26 24 30;  
SETUP = 35 20 60 70 75 30;  
AVAILABLE = 800 1160 1780 1050 1360 1240;  
USAGE = 1 4 0 4 2 0  
        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
```

```
! Maximize profits minus setup costs
MAX = @SUM( PLANES: PROFIT * QUANTITY - SETUP * BUILD);

@FOR( RESOURCES( I):
    @SUM( PLANES( J): USAGE( I, J) * QUANTITY( J)) <=
        AVAILABLE( I)
    );
@FOR( PLANES: QUANTITY <= 400 * BUILD;
@BIN( BUILD)      ! BINARY (YES/NO) DECISION TO BUILD
    );
@FOR( PLANES:
    @GIN( QUANTITY) ! INTEGER RESTRICTIONS ON # PLANES BUILT
    );

END
```