```
param n, integer, >=3; # número de nós
set cidades := {1..n};
set arcos within cidades cross cidades;
param c{(i,j) in arcos};
var X{(i,j) in arcos}, binary;
var Y{(i,j) in arcos}, >=0;

# Função Objetivo

minimize: total sum{(i,j) in arcos} c[i,j] * X[i,j];
```

```
param n, integer, >= 3;  /* numero de nós */
set V := 1..n;           /* conjunto de nós  */
set A, within V cross V; /* conjunto de arcos */
param c{(i,j) in A};     /* distância do nó i ao nó j */
var x{(i,j) in A}, binary;/* x[i,j] = 1 se o caixeiro viajante for do nó i para  j */
var y{(i,j) in A}, >= 0;  /* y[i,j] é o número de itens que vão do nó i par o nó j

minimize total: sum{(i,j) in A} c[i,j] * x[i,j];
/* o objetivo é minimizar a distância total percorrida */

saida{i in V}: sum{(i,j) in A} x[i,j] = 1;
/* o viajante deixa cada nó uma única vez */

chegada{j in V}: sum{(i,j) in A} x[i,j] = 1;
/* o viajante chega em cada nó uma única vez  */

nos{i in V: i!=1}: sum{(j,i) in A} y[j,i]-sum{(i, j) in A} y[i, j]= 1;
/* o fluxo que sai - fluxo que chega no nó é =  1 , ou seja, o viajante deixa um item no nó*/

nos1: sum{(1,j) in A} y[1, j] - sum{(j, 1) in A} y[j, 1] = n - 1;
/*  o fluxo que sai da origem - fluxo que chega na origem = n-1, o viajante leva n itens */

cap{(i,j) in A}: y[i,j] <= (n-1) * x[i,j];
/* se (i,j) não pertence ao ciclo, sua capacidade deve ser 0; c/c. será <=  n-1 objetos */
```