```
# tsp.run
model tsp.mod;
data tsp.dat;
# distance matrix must be symmetric
for {i in V, j in V : i > j} {
  let d[i,j] := d[j,i];
}

option solver cplex;
# don’t print CPLEX messages
option solver_msg 0;
# data structures required for the algorithm
let subtours := 0;
param successorvertex{V} >= 0, integer;
param currentvertex >= 0, integer;
param termination binary;
let termination := 0;
# iterative part
repeat while (termination = 0) {
  # solve the problem
  solve;
  let subtours := subtours + 1;
  # find the successors of each vertex
  for {i in V} {
    let successorvertex[i] := sum{j in V : j != i} j * x[i,j];
  }
  # find a subtour
  let currentvertex := 1;
  let S[subtours] := {};
  repeat {
    let S[subtours] := S[subtours] union {currentvertex};
    let currentvertex := successorvertex[currentvertex];
  } until (currentvertex = 1);
  # print the subtour we wish to eliminate
  printf "(sub)tour: (";
  for {i in S[subtours]} {
    printf "%d -> ", i ;
  }
  printf "1)\n";
  # termination condition
  if (card(S[subtours]) >= n) then {
    # Hamiltonian tour, terminate
    let termination := 1;
  }
}
printf "travelled distance in the optimal tour: %f\n", distance;
```