```
# COMPLEMENTO DO PROBLEMA DE CUSTOS - ENTREGAR ATÉ 24/01
# Resolver o problema do menor custo sem usar nenhuma sequência de comandos
# Não usar delimitadores na matriz, mas produzir uma nova matriz que será
# enviada na chamada recursiva. Por exemplo, para a matriz
#
# [ [ 1, 7, 2 ],
#   [ 1, 1, 1 ],
#   [ 4, 6, 1] ]
#
# Produzir
#
# [    [ 1, 1, 1 ],                 -> tentativa para baixo
#      [ 4, 6, 1] ]
#
# e...
#
#    [ [ 7, 2 ],                    -> tentativa para a direita
#      [ 1, 1 ],
#      [ 6, 1] ]
#  (usar funções recursivas para isso
#  Decomponha bastante o problema para reduzir a complexidade
#  Recurso importante: criar uma função que forneça a cauda de uma lista,
#                      isto é, todos os elementos menos o primeiro
#  Observe que essa função serve tanto para listas de inteiros, como  [2,5,3,1],
#   listas de caracteres, como ["a","v","t","r"], ou listas de listas que é como
#   criamos uma matriz (exemplo acima)


m = [ [ 4, 5, 7, 0, 1, 1,13],
      [ 3, 5, 2, 1, 6, 1, 4],
      [ 1, 3, 9,11, 9, 0, 3],
      [17, 3, 8, 4, 1, 1,14],
      [ 2, 0, 9, 5,11, 0, 4],
      [ 6, 4, 0, 2, 1, 1, 3],
      [ 7,13, 2, 0, 7, 1, 2] ]

m2 = [ [ 1, 7, 2 ],
       [ 1, 1, 1 ],
       [ 4, 6, 1] ]


def minCusto(m):
   if ...
      return ...
   elif ...
      return ...
   elif ...
       ...




print ("MínimoCusto: ", minCusto(m))
```