```
# Assignment Problem (Hungarian Method)
import numpy as np
# The Input Matrix
inp_mat = np.array([
  [80, 40, 50, 46],
  [40, 70, 20, 25],
  [30, 10, 20, 30],
  [35, 20, 25, 30]
])
cost = 0
# Duplicate Matrices
dup_mat = np.copy(inp_mat)
inp_mat_copy = np.copy(inp_mat)
# Row Reduction (Subtracting min element from other elements of same row)
for row in dup_mat:
  min_ele = min(row)
  i = 0
  while (i < len(row)):
    row[i] -= min_ele
    i += 1
# Column Reduction (Subtracting min element from other elements of same
Column)
for col in dup_mat.T:
  min_ele = min(col)
  i = 0
  while (i < len(col)):
    col[i] -= min_ele
    i += 1
# Num of Lines Deleted
num_delete = 0
# Copy of Matrix After Row and Col Reductions
dup_mat_copy = np.copy(dup_mat)
i = 0
# Indices of Columns Deleted
cols = []
# Deleting Columns with Zero Row Wise
while (i < len(dup_mat)):
  # Getting Index of Zero in the row
  zeroes = np.where(dup_mat[i] == 0)[0]
  if len(zeroes) == 1:
    # Adding it's Cost
    cost += inp_mat_copy[i][zeroes[0]]
    # Deleting it
    dup_mat = np.delete(dup_mat, zeroes[0], axis=1)
    inp_mat_copy = np.delete(inp_mat_copy, zeroes[0], axis=1)
    num_delete += 1
    cols.append(zeroes[0])
  i += 1
i = 0
dup_mat = dup_mat.T
inp_mat_copy = inp_mat_copy.T
rows = []
# Deleting Rows with Zero Col Wise
while (i < len(dup_mat)):
  zeroes = np.where(dup_mat[i] == 0)[0]
  if len(zeroes) == 1:
    cost += inp_mat_copy[i][zeroes[0]]
    dup_mat = np.delete(dup_mat, zeroes[0], axis=1)
    inp_mat_copy = np.delete(inp_mat_copy, zeroes[0], axis=1)
```

```
    num_delete += 1
    rows.append(zeroes[0])
  i += 1
dup_mat = dup_mat.T
inp_mat_copy = inp_mat_copy.T
if num_delete == len(inp_mat):
  print (cost)
else:
  # Get the Min Element from the Current Matrix
  min_element = np.amin(dup_mat)
  i = 0
  j = 0
  # Subtract Minimum from Uncovered Rows
  while i < len(dup_mat_copy):
    if i not in rows:
      while j < len(dup_mat_copy):
        dup_mat_copy[i][j] -= min_element
        j += 1
      j = 0
    i +=1
  i = 0
  j = 0
  # Add Minimum to Covered Cols
  while i < len(dup_mat_copy):
    while j < len(dup_mat_copy):
      if j in cols:
        dup_mat_copy[i][j] += min_element
      j += 1
    j = 0
    i +=1
  # Finding the Cost Again
  cost = 0
  inp_mat_copy = np.copy(inp_mat)
  num_zeroes = np.where(dup_mat_copy == 0)[0]
  # Repeat Until All Zeroes are Over
  while (len(num_zeroes) != 0):
    i = 0
    while (i < len(dup_mat_copy)):
      zeroes = np.where(dup_mat_copy[i] == 0)[0]
      if len(zeroes) == 1:
        cost += inp_mat_copy[i][zeroes[0]]
        dup_mat_copy = np.delete(dup_mat_copy, zeroes[0], axis=1)
        inp_mat_copy = np.delete(inp_mat_copy, zeroes[0], axis=1)
      i += 1
    i = 0
    dup_mat_copy = dup_mat_copy.T
    inp_mat_copy = inp_mat_copy.T
    while (i < len(dup_mat_copy)):
      zeroes = np.where(dup_mat_copy[i] == 0)[0]
      if len(zeroes) == 1:
        cost += inp_mat_copy[i][zeroes[0]]
        dup_mat_copy = np.delete(dup_mat_copy, zeroes[0], axis=1)
        inp_mat_copy = np.delete(inp_mat_copy, zeroes[0], axis=1)
      i += 1
    dup_mat_copy = dup_mat_copy.T
    inp_mat_copy = inp_mat_copy.T
    num_zeroes = np.where(dup_mat_copy == 0)[0]
  print (cost)
```