Create Fuzzy relation by Cartesian product of any two Fuzzy sets and perform Max-Min composition of any two Fuzzy relations.

122021601009 - ALLENA VENKATA SAI ABHISHEK

```
pip install colorama
```

```
Collecting colorama
  Downloading https://files.pythonhosted.org/packages/44/98/5b86278fbbf250d239ae0ecb7
Installing collected packages: colorama
Successfully installed colorama-0.4.4
```

For different conditions you have to change the conditions in the Cartesian product and Crisp relation functions

```
import numpy as np
from colorama import Fore
### This function takes the user input with same number of elements in set 1 and set2####
def UserInput():
 listX =[]
 listY=[]
 print("Enter number of elements X: ")
 nX=int(input())
 print("Enter number of elements Y: ")
 nY = int(input())
 print(Fore.RED + "Enter values for set X: ")
 for i in range(nX):
    x= int(input())
    listX.append(x)
 print("Enter values for set Y: ")
 for j in range(nY):
   y= int(input())
   listY.append(y)
 print("X = {",listX,"}")
 print("Y = {",listY,"}")
 CartesianProduct(listX,listY) ## passing the listX and listY to Cartesianproduct function
 crisprelationRS(listX,listY)## passing the listX and listY to crisp relationRS function
###This function creates cartesian product of two sets
def CartesianProduct(listX,listY):
 print()
 print(Fore.BLUE+"cartesian product of X and Y")
 print("{",end="")
 for k in listX:
  for m in listY:
    print((k,m),end=",")
 print("}")
 print()
 print("R relation:")
 print("{",end="")
 for k in listX:
  for m in listY:
    if (k+2==m):
      print((k,m),end=",")
 print("}")
 print()
 print("S relation: ")
 print("{",end="")
 for k in listX:
  for m in listY:
    if (k<m):
      print((k,m),end=",")
 print("}")
#### This function creates a crisp relation for R and S ,generates matrices for R and S
def crisprelationRS(listX,listY):
 print()
 print(Fore.GREEN+"R Matrix: ")
 new_R=[]
 new_S=[]
 for x in listX:
  for y in listY:
    if (y==x+2):
      new_R.append(1)
    else:
      new_R.append(0)
    if (x<y):
      new_S.append(1)
    else:
      new_S.append(0)
 R_matrix = np.array(new_R).reshape(len(listX),len(listY))
 S_matrix = np.array(new_S).reshape(len(listX),len(listY))
 print(R_matrix)
 print()
 print("S Matrix")
 print(S_matrix)
 MinOperation(R_matrix,S_matrix)## passing the R_matrix and S_matrix to MinOperation funct
###This function is used to find the minimum operation in composition operation
def MinOperation(R_matrix,S_matrix):
 min_list=[]
 # iterate through rows of R_matrix
 for i in range(len(R_matrix)):
  for j in range(len(S_matrix[0])): #column length to compare the elements number of times
 # iterate through rows of S_matrix
    for k in range(len(S_matrix)):
      min_list.append(min(R_matrix[i][k] , S_matrix[k][j]))
 A=np.array(min_list).reshape(len(R_matrix)*len(R_matrix),len(S_matrix ))
 print(Fore.LIGHTRED_EX+"Minimum operation (Ros): ")
 print(A)
 MaxOperation(A,R_matrix,S_matrix) ## passing the A list,R_matrix and S_matrix to MaxOpera
###This function is used to find the maximum operation which is the next step in compositi
```

```
def MaxOperation(A,R_matrix,S_matrix):
 print()
 print("composition operation RoS is:")
 com_list=[]
 for i in range(len(A)):
  max_A=max(A[i])
  com_list.append(max_A)
 B=np.array(com_list).reshape(len(R_matrix),len(S_matrix))
 print(B)
UserInput() ### calling the userInput function
```

```
Enter number of elements X:
3
Enter number of elements Y:
3
Enter values for set X:
1
3
5
Enter values for set Y:
1
3
5
X = { [1, 3, 5] }
Y = { [1, 3, 5] }

cartesian product of X and Y
{(1, 1),(1, 3),(1, 5),(3, 1),(3, 3),(3, 5),(5, 1),(5, 3),(5, 5),}

R relation:
{(1, 3),(3, 5),}

S relation:
{(1, 3),(1, 5),(3, 5),}

R Matrix:
[[0 1 0]
 [0 0 1]
 [0 0 0]]

S Matrix
[[0 1 1]
 [0 0 1]
 [0 0 0]]
Minimum operation (Ros):
[[0 0 0]
 [0 0 0]
 [0 1 0]
 [0 0 0]
 [0 0 0]
 [0 0 0]
 [0 0 0]
 [0 0 0]
 [0 0 0]]

composition operation RoS is:
[[0 0 1]
 [0 0 0]
 [0 0 0]]
```

✓ 38s completed at 2:34 PM ● ✕