```
import numpy as np

def check(allocation):
    allocatedCells = 0
    m = len(allocation[0])
    n = len(allocation)

    for i in range(0, n):
        for j in range(0, m):
            if allocation[i][j] > 0:
                allocatedCells += 1

    if (m + n - 1) == allocatedCells:
        return True

    return False

def findZero(allocation):
    m = len(allocation[0])
    n = len(allocation)
    allocatedCells = 0
    maxAllocated = 0
    maxi = -1
    maxj = -1
    val = -1

    for i in range(0, n):
        for j in range(0, m):
            if allocation[i][j] > 0:
                allocatedCells += 1
        if allocatedCells > maxAllocated:
            maxAllocated = allocation[i][j]
            maxi = i
            maxj = j
            val = "row"

    for j in range(0, m):
        for i in range(0, n):
            if allocation[i][j] > 0:
                allocatedCells += 1
        if allocatedCells > maxAllocated:
            maxAllocated = allocation[i][j]
            maxi = i
            maxj = j
            val = "col"

    if val == "col":
        return maxj, val

    else:
        return maxi, val


def initVU(v, u, matrix, allocation, val, maxindex):
    n = len(allocation)
    m = len(allocation[0])
    flag = 0

    for i in range(0, n):
        if u[i] == 9999:
            flag += 1
            break
    for i in range(0, m):
        if v[i] == 9999:
            flag += 1
            break

    if flag == 0:
        return u, v
```

```
    if val == "col":
        for i in range(0, n):
            if allocation[i][maxindex] > 0 and u[i] == 9999:
                u[i] = matrix[i][maxindex] - v[maxindex]
                u, v = initVU(v, u, matrix, allocation, "row", i)

    else:
        for j in range(0, m):
            if allocation[maxindex][j] > 0 and v[j] == 9999:
                v[j] = matrix[maxindex][j] - u[maxindex]
                u, v = initVU(v, u, matrix, allocation, "col", j)

    return u, v


def allocationChecker(allocationCheck) :
    m = len(allocationCheck[0])
    n = len(allocationCheck)

    for i in range(0, n):
        for j in range(0, m):
            if allocationCheck[i][j] < 0:
                return True

    return False


def newAllocation(allocationCheck, matrix, u, v):
    m = len(allocationCheck[0])
    n = len(allocationCheck)

    for i in range(0, n):
        for j in range(0, m):
            if allocationCheck[i][j] != 9999:
                allocationCheck[i][j] = matrix[i][j] - u[i] - v[j]

    return allocationCheck


def findNegIndex(allocationCheck):
    m = len(allocationCheck[0])
    n = len(allocationCheck)
    indI =- 1
    indJ =- 1
    allocated = 0
    for i in range(0, n):
        for j in range(0, m):
            if allocationCheck[i][j] < allocated:
                allocated = allocationCheck[i][j]
                indI = i
                indJ = j

    return indI, indJ


def traverseMatrix(allocation, indI, indJ, allocationCheck) :
    doneAllocation = np.array([
        [0,0,0,0],
        [0,0,0,0],
        [0,0,0,0]
    ])
    doneAllocation[indI][indJ] = 9999
    flag = 1
    f = 0

    allocation, flag, doneAllocation = moveRight(allocation, indI, indJ+1, allocationCheck, doneAllocation, f)
    if flag == 0:
        doneAllocation[indI][indJ] = 1
```

```
        return allocation, doneAllocation
    allocation, flag, doneAllocation = moveLeft(allocation, indI, indJ-1, allocationCheck, doneAllocation, f)
    if flag == 0:
        doneAllocation[indI][indJ] = 1
        return allocation, doneAllocation
    allocation, flag, doneAllocation = moveDown(allocation, indI+1, indJ, allocationCheck, doneAllocation, f)
    if flag == 0:
        doneAllocation[indI][indJ] = 1
        return allocation, doneAllocation
    allocation, flag, doneAllocation = moveUp(allocation, indI-1, indJ, allocationCheck, doneAllocation, f)
    if flag == 0:
        doneAllocation[indI][indJ] = 1
        return allocation, doneAllocation

def moveRight(allocation, indI, indJ, allocationCheck, doneAllocation, f):
    if indJ >= len(allocation[0]):
        return allocation, 1, doneAllocation
    elif doneAllocation[indI][indJ] == 1:
        return allocation, 1, doneAllocation
    elif doneAllocation[indI][indJ] == 9999:
        return allocation, 0, doneAllocation
    elif allocationCheck[indI][indJ] == 9999:
        if f == 1:
            doneAllocation[indI][indJ] =- 1
            f = 0
        else:
            doneAllocation[indI][indJ] = 1
            f = 1
        allocation, flag, doneAllocation = moveRight(allocation, indI, indJ+1, allocationCheck, doneAllocation, f)
        if flag == 0:
            doneAllocation[indI][indJ] = 0
            return allocation, 0, doneAllocation
        allocation, flag, doneAllocation = moveDown(allocation, indI+1, indJ, allocationCheck, doneAllocation, f)
        if flag == 0:
            return allocation, 0, doneAllocation
        allocation, flag, doneAllocation = moveUp(allocation, indI-1, indJ, allocationCheck, doneAllocation, f)
        if flag == 0:
            return allocation, 0, doneAllocation
    else:
        allocation, flag, doneAllocation = moveRight(allocation, indI, indJ+1, allocationCheck, doneAllocation, f)
        return allocation, flag, doneAllocation


def moveLeft(allocation, indI, indJ, allocationCheck, doneAllocation, f):
    if indJ < 0:
        return allocation, 1, doneAllocation
    elif doneAllocation[indI][indJ] == 1:
        return allocation, 1, doneAllocation
    elif doneAllocation[indI][indJ] == 9999:
        return allocation, 0, doneAllocation
    elif allocationCheck[indI][indJ] == 9999:
        if f == 1:
            doneAllocation[indI][indJ] =- 1
            f = 0
        else:
            doneAllocation[indI][indJ] = 1
            f = 1
        allocation, flag, doneAllocation = moveLeft(allocation, indI, indJ-1, allocationCheck, doneAllocation, f)
        if flag == 0:
            doneAllocation[indI][indJ] = 0
            return allocation, 0, doneAllocation
        allocation, flag, doneAllocation = moveDown(allocation, indI+1, indJ, allocationCheck, doneAllocation, f)
        if flag == 0:
            return allocation, 0, doneAllocation
        allocation, flag, doneAllocation = moveUp(allocation, indI-1, indJ, allocationCheck, doneAllocation, f)
        if flag == 0:
            return allocation, 0, doneAllocation
    else:
        allocation, flag, doneAllocation = moveLeft(allocation, indI, indJ-1, allocationCheck, doneAllocation, f)
```

```
        return allocation, flag, doneAllocation

    def moveDown(allocation, indI, indJ, allocationCheck, doneAllocation, f):
        if indI >= len(allocation):
            return allocation, 1, doneAllocation
        elif doneAllocation[indI][indJ] == 1:
            return allocation, 1, doneAllocation
        elif doneAllocation[indI][indJ] == 9999:
            return allocation, 0, doneAllocation
        elif allocationCheck[indI][indJ] == 9999:
            if f == 1:
                doneAllocation[indI][indJ] =- 1
                f = 0
            else:
                doneAllocation[indI][indJ] = 1
                f = 1
            allocation, flag, doneAllocation = moveLeft(allocation, indI, indJ-1, allocationCheck, doneAllocation, f)
            if flag == 0:
                return allocation, 0, doneAllocation
            allocation, flag, doneAllocation = moveRight(allocation, indI, indJ+1, allocationCheck, doneAllocation, f)
            if flag == 0:
                return allocation, 0, doneAllocation
            allocation, flag, doneAllocation = moveDown(allocation, indI+1, indJ, allocationCheck, doneAllocation, f)
            if flag == 0:
                doneAllocation[indI][indJ] = 0
                return allocation, 0, doneAllocation
        else:
            allocation, flag, doneAllocation = moveLeft(allocation, indI, indJ-1, allocationCheck, doneAllocation, f)
            return allocation, flag

    def moveUp(allocation, indI, indJ, allocationCheck, doneAllocation, f):
        if indI < 0:
            return allocation, 1, doneAllocation
        elif doneAllocation[indI][indJ] == 1:
            return allocation, 1, doneAllocation
        elif doneAllocation[indI][indJ] == 9999:
            return allocation, 0, doneAllocation
        elif allocationCheck[indI][indJ] == 9999:
            if f == 1:
                doneAllocation[indI][indJ] =- 1
                f = 0
            else:
                doneAllocation[indI][indJ] = 1
                f = 1
            allocation, flag, doneAllocation = moveLeft(allocation, indI, indJ-1, allocationCheck, doneAllocation, f)
            if flag == 0:
                return allocation, 0, doneAllocation
            allocation, flag, doneAllocation = moveRight(allocation, indI, indJ+1, allocationCheck, doneAllocation, f)
            if flag == 0:
                return allocation, 0, doneAllocation
            allocation, flag, doneAllocation = moveUp(allocation, indI-1, indJ, allocationCheck, doneAllocation, f)
            if flag == 0:
                doneAllocation[indI][indJ] = 0
                return allocation, 0, doneAllocation
        else:
            allocation, flag, doneAllocation = moveLeft(allocation, indI, indJ-1, allocationCheck, doneAllocation, f)
            return allocation, flag, doneAllocation


    matrix = np.array([
        [19, 30, 50, 10],
        [70, 30, 40, 60],
        [40, 8, 70, 20]
    ])
    demand = np.array([5,8,7,14])
    supply = np.array([7,9,18])

    val = ""
    maxindex = 0
```

```
u = np.array([9999, 9999, 9999])
v = np.array([9999, 9999, 9999, 9999])

allocation = np.array([
    [5, 0, 0, 2],
    [0, 0, 7, 2],
    [0, 8, 0, 10]
])
allocationCheck = np.array([
    [9999, -1, -1, 9999],
    [-1, -1, 9999, 9999],
    [-1, 9999, -1, 9999]
])
doneAllocation = np.array([
    [0, 0, 0, 0],
    [0, 0, 0, 0],
    [0, 0, 0, 0]
])


while True :
    min = 9999
    cost = 0
    checker = check(allocation)

    if checker == True:

        maxindex, val = findZero(allocation)
        if val == "col":
            v[maxindex] = 0
            u, v = initVU(v, u, matrix, allocation, val, maxindex)

        else :
            u[maxindex] = 0
            u, v = initVU(v, u, matrix, allocation, val, maxindex)

        allocationCheck = newAllocation(allocationCheck, matrix, u, v)

        finding = allocationChecker(allocationCheck)
        if finding == False:
            break

        indI, indJ = findNegIndex(allocationCheck)

        allocation, doneAllocation = traverseMatrix(allocation, indI, indJ, allocationCheck)

        for i in range(0, len(allocation)):
            for j in range(0, len(allocation[0])):
                if doneAllocation[i][j] != 0 and allocation[i][j] != 0:
                    if min > allocation[i][j]:
                        min = allocation[i][j]

        for i in range(0, len(allocation)):
            for j in range(0, len(allocation[0])):
                if doneAllocation[i][j] != 0:
                    allocation[i][j] = allocation[i][j] + doneAllocation[i][j] * min

        for i in range(0, len(allocation)):
            for j in range(0, len(allocation[0])):
                cost = cost + allocation[i][j] * matrix[i][j]
        # print(f"Matrix :\n{matrix}")
        # print(f"Old Allocation : \n{allocation}")
        print(f"Old Cost: {779}")
        # print(f"New Allocation : \n{allocation}")
        print(f"Optimized Cost: {cost}")

        for i in range(0, len(allocation)):
            for j in range(0, len(allocation[0])):
```

```
            if allocation[i][j] == 0:
                allocationCheck[i][j] =- 1
            else :
                allocationCheck[i][j] = 9999

    else :
        print("Degeneracy occurs!")
        break
```