Tên: `Nguyễn Trường Phát`

MSSV: `17520880`

Lớp: `KHTN2017`

```
import numpy as np
class Node:
    def __init__(self,Name):
        self.Name = Name
        self.Edges = {} #{Edge.Name:[object,weight,pheromone]}
        self.From = None
    def AddEdge(self,Edge,Weight):
        # Cập nhật dict cạnh bằng tuple gồm địa chỉ Edge và weight và pheromone khởi tạo =
0
        self.Edges[Edge.Name]=[Edge,Weight,0.0001]
    def updateWeight(self,nodeB,Weight):
        self.Edges[nodeB.Name][1] = Weight
        nodeB.Edges[self.Name][1] = Weight
    def updatePhero(self,nodeB,Phero,stage):
        self.Edges[nodeB.Name][2] += Phero
        nodeB.Edges[self.Name][2] += Phero
class Ant:
    def __init__(self,startNode):
        self.startNode = startNode
        self.Location = startNode
        self.Path = []
        self.Distance = 0
    def findNextWay(self,homeRun = False):
        # Compute probability for each way
        Edges = self.Location.Edges.copy()
        nameEdges = [name[0] for name in Edges.keys()]
        #Xóa những tên đã có trong path
        for nodeName in self.Path:
            try:
                del Edges[nodeName]
                nameEdges.remove(nodeName)
            except:
                pass
        # Nếu chưa tìm được đỉnh cuối thì chưa được quay về (homeRun) -> Xóa đỉnh đầu ra
khỏi đỉnh mục tiêu (nếu có)
        if (homeRun == False):
            if (self.startNode.Name in nameEdges):
                del Edges[self.startNode.Name]
                nameEdges.remove(self.startNode.Name)
        # Tổng số pheromones
        sumPheromones = sum([a[2] for a in Edges.values()])
        # Tính xác suất cho từng cạnh tiếp theo
        probEdges = [ ((data[2])/sumPheromones) for data in Edges.values()]
        # Random xác suất
        probWay = np.random.random()
```

```
        # Tính xác suất cumulative sum
        csProb = np.cumsum([0] + probEdges)
        # Chọn đỉnh
        chooseEdgeName = None
        for i in reversed(range(len(csProb))):
            if (probWay > csProb[i]):
                chooseEdgeName = nameEdges[i]
                idx = nameEdges.index(chooseEdgeName)
                break
        return chooseEdgeName
    def goNextWay(self,homeRun = False):
        # Tìm tên đường ngắn nhất
        chooseEdgeName = self.findNextWay(homeRun)
        # Thêm vào history các điểm đã đi qua
        self.Path.append(chooseEdgeName)
        # Cập nhật quãng đường đã đi
        self.Distance += self.Location.Edges[chooseEdgeName][1]
        # Cập nhật location qua điểm mới
        self.Location = self.Location.Edges[chooseEdgeName][0]
    def resetAnt(self):
        self.Path = []
        self.Distance = 0
    def goGraph(self,targetName):
        self.goNextWay()
        while (self.Location.Name != targetName):
            self.goNextWay()
        while (self.Location.Name != self.startNode.Name):
            self.goNextWay(homeRun=True)
    def applyPheromones(self,stage):
        Pheromones = 100/(self.Distance**5) #Pheromones = 10000/khoảng cách đã đi mũ 4
        # Thêm pheromones vào từng cạnh
        for i in range(len(self.Path)):
            self.startNode.updatePhero(self.startNode.Edges[self.Path[i]]
[0],Pheromones,stage)
            self.startNode = self.startNode.Edges[self.Path[i]][0]
        self.resetAnt()

class AntColony:
    def __init__(self,weight_mat,startName,endName):
        self.Graph={}
        self.weight_mat = weight_mat
        self.startName = startName
        self.endName = endName
    def initGraph(self):
        names = ['A','B','C','D','E']
        # Khởi tạo graph
        for i in range(len(self.weight_mat)):
            self.Graph[names[i]] = Node(Name=names[i])
        #Cập nhật cạnh và trọng số
        for i in range(len(self.weight_mat)):
            for j in range(len(self.weight_mat)):
                # Nếu weight = 0 thì bỏ qua
                if (self.weight_mat[i][j]==0): continue
```

```
                self.Graph[names[i]].AddEdge(Edge=self.Graph[names[j]],Weight=weight_mat[i]
[j])

    def findPath(self,numAnts):
        for i in range(100):
            myAnts = []
            for i in range(numAnts):
                myAnts.append(Ant(self.Graph[self.startName]))

            for ant in myAnts:
                ant.goGraph(self.endName)

            for ant in myAnts:
                ant.applyPheromones(stage=i)
        myPath = []
        homeRun = False
        Location = self.Graph[self.startName]
        while(self.startName not in myPath):
            pheroNextList = [[edge,phero[2]] for (edge,phero) in
zip(Location.Edges.keys(),Location.Edges.values())
                             if edge not in myPath]
            pheroNextList = sorted(pheroNextList,key=lambda x:x[1],reverse=True)
            if homeRun == False:
                try:
                    pheroNextList.remove([self.startName,Location.Edges[self.startName]
[2]])
                except:
                    pass
            myPath += [pheroNextList[0][0]]
            if self.endName in myPath:
                homeRun = True
            Location = Location.Edges[pheroNextList[0][0]][0]

        print('Đường ngắn nhất là:',[self.startName]+myPath)
```

```
weight_mat = np.array([[0,17,10,9,12],
                       [17,0,8,14,5],
                       [10,8,0,7,11],
                       [9,14,7,0,11],
                       [12,5,11,11,0]])
myColony = AntColony(weight_mat,'B','D') # ma trận khoảng cách, điểm đầu và điểm cuối
myColony.initGraph() # Khởi tạo graph
myColony.findPath(numAnts=2) # Tìm đường đi ngắn nhất và in ra, với 2 con kiến
```

```
Đường ngắn nhất là: ['B', 'D', 'B']
```