import time

from lab1 import pathfinder

class MaxFlowAlgorithm:

    def \_\_init\_\_(self,graph):

        self.graph = graph       #defines the graph of the problem

        self.N = len(graph)      #the number of the nodes

    def B\_DFS(self,s,t,prt):

        marked=[False]\*self.N    #creating a vector to keep track of marked nodes

        q=[]                     #creating a queue of the nodes needed to be marked

        q.append(s)              #assigning the first member of the queue which is the source node of the graph

        marked[s]=True           #obviously we need to mark the source node to begin, so we mark it as true (index=0)

        prt[s]=-1                #it doesn't matter what value is determined to be the parent of the source (conventionally -1)

        while q:                 #so long as the queue of the nodes that need to be marked is not empty, the algorithm goes on

            i=q.pop()

            for j in range(self.N):   #for each node (i) we try other nodes to see if

                                      # 1:that node is the child node of i with a positive capacity

                                      # 2:the child node is not marked

                                      # if so:

                                      # 1:the child will be appended to the queue

                                      # 2:the child will be marked as traversed

                                      # 3: the parent node of the child will be determined

                if marked[j]==False and self.graph[i][j]>0:

                    q.append(j)

                    marked[j]=True

                    prt[j]=i

        if marked[t]:                 # if the sink node (t) of the graph is traveresed we are done

            return True

        else:

            return False

    def FordFulkerson(self,s,t):

        residualgraph=self.graph

        prt=[0]\*self.N

        maximum\_flow=0

        while self.B\_DFS(s,t,prt):

            pathflow=float('inf')     # a large number is needed for the first iteration of the line 57

            j=t                       # begin from the sink node and backtrack until you reach the source

            while not j == s:

                i=prt[j]

                pathflow=min(pathflow, residualgraph[i][j])     #we need to calculate the min flow of the path

                                                                #to attaint the residual graph in each iteration

                j=prt[j]

            j=t

            while not j == s:

                i=prt[j]

                residualgraph[i][j] -= pathflow                   #calculating the new capacity of each edge

                residualgraph[j][i] += pathflow

                j=prt[j]

            maximum\_flow += pathflow

        return maximum\_flow

graph= [[0,12,14,0,11,0,7],

        [0,0,17,17,0,20,0],

        [0,0,0,10,12,0, 16],

        [0,0,0,0,9,0,11],

        [0,0,0,0,0,12,0],

        [0,0,15,0,0,0,9],

        [0,0,0,0,0,0,0]]

graphshape = {}

keys = range(len(graph))

for k in keys:

    graphshape[k]=[]

for u in keys:

    for v in keys:

        if graph[u][v]>0:

            graphshape[u].append([v])

pathfinder(graph, 0)

N=len(graph)

MFA=MaxFlowAlgorithm(graph)

print("Максимальний потік цієї мережі дорівнює {}".format(MFA.FordFulkerson(0,6)))