import numpy

import random

inp = 'G2'

vert = int(numpy.loadtxt(open(inp, "rb"), dtype=(str), delimiter=" ", max\_rows=1)[0])

edges = int(numpy.loadtxt(open(inp, "rb"), dtype=(str), delimiter=" ", max\_rows=1)[1])

result = numpy.loadtxt(open(inp, "rb"), dtype=(str), delimiter=" ", skiprows=1)

nodes = []

distances = {}

for i in range(vert):

nodes.append(str(i+1))

#print(nodes)

for source, dest, weight in result:

if source not in distances:

distances[source]= {}

if dest not in distances[source]:

# distances[source].update({dest:random.randint(1, 15)})

distances[source].update({dest:float(weight)})

for dest, source, weight in result:

if source not in distances:

distances[source]= {}

if dest not in distances[source]:

distances[source].update({dest:random.randint(1, 15)})

#distances[source].update({dest:int(weight)})

unvisited = {node: None for node in nodes} #using None as +inf

visited = {}

current = '1'

currentDistance = 0

unvisited[current] = currentDistance

path=[]

path.append(current)

while True:

for neighbour, distance in distances[current].items():

if neighbour not in unvisited: continue

newDistance = currentDistance + distance

if unvisited[neighbour] is None or unvisited[neighbour] > newDistance:

unvisited[neighbour] = newDistance

visited[current] = currentDistance

del unvisited[current]

if not unvisited: break

candidates = [node for node in unvisited.items() if node[1] ]

print(candidates)

current, currentDistance = sorted(candidates, key = lambda x: x[1])[0]

path.append(current)

print(visited)

print(path)