### UE20CS323

## Graph Theory and Its Applications

## Assignment 234 – Problem 2

Name : Renita Kurian

SRN : PES1UG20CS331

Roll No. : 13

**Problem Statement:**

A highways department must inspect its roads for fallen trees. The adjacency matrix stores the lengths of the roads, in miles, that must be inspected in one district. List the different ways in which the odd vertices can be paired. Find the shortest distance that must be travelled in inspecting all the roads in the district, starting, and finishing at the same point. Find the number of ways of pairing these odd vertices.

**Solution:**

The above problem can be solved using Chinese postman algorithm. The graph is first checked for odd vertices. If there are odd vertices present then one or more edges are added using shortest path algorithm. The sum of edges or road lengths will be the shortest distance that must be travelled to inspect all roads. Dijkstra’s algorithm is used to calculate the shortest distance between two nodes.

Dijkstra’s algorithm is a single source shortest path algorithm that finds the shortest distance from a source vertex to all other vertices in a graph. It is a greedy algorithm and is only applicable when there are no negative weights.

The solution has been implemented in Python. Code attached below.

**Code:**

def sum\_edges(graph):

sum = 0

l = len(graph)

for i in range(l):

for j in range(i,l): #Sum of edges = sum of elements in top triangle

sum += graph[i][j]

return sum

# Dijkstra's algorithm to find shortest distance between two nodes

def dijktra(graph, source, dest):

shortest = [0 for i in range(len(graph))]

selected = [source]

l = len(graph)

inf = 10000000

min\_sel = inf

for i in range(l):

if(i==source):

shortest[source] = 0

else:

if(graph[source][i]==0):

shortest[i] = inf

else:

shortest[i] = graph[source][i]

if(shortest[i] < min\_sel):

min\_sel = shortest[i]

ind = i

if(source==dest):

return 0

selected.append(ind)

while(ind!=dest):

for i in range(l):

if i not in selected:

if(graph[ind][i]!=0):

if((graph[ind][i] + min\_sel) < shortest[i]):

shortest[i] = graph[ind][i] + min\_sel

temp\_min = 1000000

for j in range(l):

if j not in selected:

if(shortest[j] < temp\_min):

temp\_min = shortest[j]

ind = j

min\_sel = temp\_min

selected.append(ind)

return shortest[dest]

#Finding odd degree vertices in graph

def odd\_vertices(graph):

degrees = [0 for i in range(len(graph))] #degrees for each node set to 0 initially

for i in range(len(graph)): #loop to find degree for each node

for j in range(len(graph)):

if(graph[i][j]!=0):

degrees[i]+=1

v = [i for i in range(len(degrees)) if degrees[i]%2!=0]

print('Odd Vertices are:',v)

return v

#Function to generate unique pairs for list of given nodes

def generate\_pairs(v):

pairs = []

print("Generated Pairs: ")

for i in range(len(v)-1):

pairs.append([])

for j in range(i+1,len(v)):

pairs[i].append([v[i],v[j]])

print(v[i], "-", v[j])

print()

return pairs

def get\_pairs(pairs, done, final, l, pairings\_sum):

if(pairs[0][0][0] not in done):

done.append(pairs[0][0][0])

for i in pairs[0]:

f = final[:]

val = done[:]

if(i[1] not in val):

f.append(i)

else:

continue

if(len(f)==l):

pairings\_sum.append(f)

return

else:

val.append(i[1])

get\_pairs(pairs[1:],val, f, l, pairings\_sum)

else:

get\_pairs(pairs[1:], done, final, l, pairings\_sum)

#Chinese Postman Algorithm

def Chinese\_Postman(graph):

odds = odd\_vertices(graph)

# If eulerian circuit then return sum of road lengths

if(len(odds)==0):

return sum\_edges(graph)

# If not eulerian circuit then,

pairs = generate\_pairs(odds)

pairings\_sum = []

l = (len(pairs)+1)//2

get\_pairs(pairs,[],[], l, pairings\_sum)

min\_sums = []

#print(pairings\_sum)

for i in pairings\_sum:

s = 0

for j in range(len(i)):

s += dijktra(graph, i[j][0], i[j][1])

min\_sums.append(s)

#print(min\_sums)

added\_distance = min(min\_sums)

total = added\_distance + sum\_edges(graph)

return total

# Graph as Adjacency Matrix

graph = [[0, 3, 1, 0, 5, 0],

[3, 0, 0, 1, 0, 6],

[1, 0, 0, 0, 2, 0],

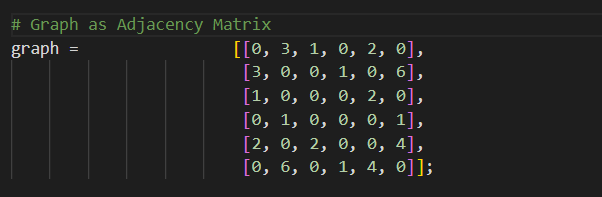
[0, 1, 0, 0, 0, 1],

[5, 0, 2, 0, 0, 4],

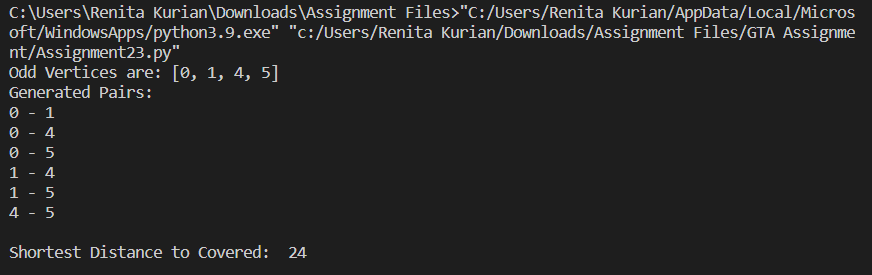
[0, 6, 0, 1, 4, 0]];

print("Shortest Distance to Covered: ",Chinese\_Postman(graph))

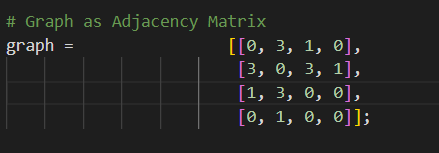
**Sample Input 1:**



**Output 1:**



**Sample Input 2:**



**Sample Output 2:**

