***Artificial Intelligence***

***CSL 411***

***Lab Journal 5***

****

**Student Name:Rida Sajjad**

**Enrolment No.01-134152-063**

**Class and Section:BSCS-7B**

**Department of Computer Science**

**BAHRIA UNIVERSITY, ISLAMABAD**

**Lab # 5: Graphs in Python**

**Objectives:**

To implement the concepts of graphs in python.

**Tools Used:**

Spyder IDLE

**Submission Date:**

**Evaluation: Signatures of Lab Engineer:**

**Task # 1:**

Change the function find path to return shortest path.

**Program:**

def find\_shortest\_path(self, start, end, path=[]):

path = path + [start]

if start == end:

return path

if start not in self.adj:

return None

shortest = None

for node in self.adj[start]:

if node not in path:

newpath = self.find\_shortest\_path( node, end, path)

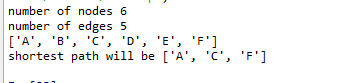
if newpath:

if not shortest or len(newpath) < len(shortest):

shortest = newpath

return shortest

**Result/Output:**



**Analysis/Conclusion:**

Here we learn about creating graph through python. Ad find shortest path of the graph of task2.

**Task # 2:**

**Program:making graph code missing because it was provide us from instructor for use**

class DWGraph(WGraph):

def add\_edge(self, u, v, w):

self.adj[u] = self.adj.get(u, []) + [v]

self.weight[(u,v)] = w

def find\_path(self, start, end, path=[]):

path = path + [start]

if start == end:

return path

if start not in self.adj:

return None

for node in self.adj[start]:

if node not in path:

newpath = self.find\_path( node, end, path)

if newpath: return newpath

return None

def find\_cost(self, start, end, path=[],cost=0):

path = path + [start]

if start == end:

print('cost is ',cost)

return path

if start not in self.adj:

return None

for node in self.adj[start]:

if node not in path:

newpath = self.find\_path(node, end,path, cost+self.get\_weight(start,node))

if newpath: return newpath

return None

d=DWGraph()

d.add\_node('A')

d.add\_node('B')

d.add\_node('C')

d.add\_node('D')

d.add\_node('E')

d.add\_node('F')

n=d.number\_of\_nodes()

print('number of nodes',n)

d.add\_edge('A','B',2)

d.add\_edge('A','C',1)

d.add\_edge('B','C',2)

d.add\_edge('B','D',5)

d.add\_edge('C','D',1)

d.add\_edge('C','F',3)

d.add\_edge('D','C',1)

d.add\_edge('D','E',4)

d.add\_edge('E','F',3)

d.add\_edge('F','C',1)

d.add\_edge('F','E',2)

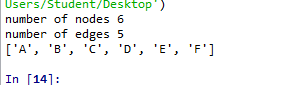
e=d.number\_of\_edges()

print('number of edges',e)

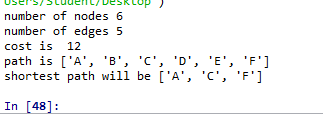
p=d.find\_path('A','F')

print(p)

**Result/Output:**



**With cost:**



**Analysis/Conclusion:**

Here we learn about creating graph through python. And also find the cost of graph.