LAB ASSIGNMENT 3

Group 1
Pranjal Mathur, Prerna, Saketh Vallakatla

- Store the graph using appropriate data structure

```
graph = { "O" : ["Z","S"],
          "Z" : ["A", "O"],
          "A" : ["Z", "S", "T"],
          "S" : ["O","A","F"],
          "T" : ["A", "L"],
          "L" : ["M","T"],
          "M":["L","D"],
          "D":["M","C"],
          "C":["P","R","D"],
          "P":["R","C","B"],
          "R":["S","P","C"],
          "F":["S","B"],
          "B":["P","G","F","U"],
          "G":["B"],
          "U":["B","H"],
          "H":["V","E"],
          "E":["H"],
          "V":["H","I"],
          "I":["V","N"],
          "N":["I"]
        }
```

- List the degree of each node

```
def printDegree(graph):
    for eachKey in graph:
        print('Degree of ',eachKey,' is ',len(graph[eachKey]))

printDegree(graph)
```

- Input any 2 nodes and find if there exists a path between them.

```
def find_all_paths(graph, start, end, path=[]):
    path = path + [start]
    if start == end:
        return [path]
    if not graph.has_key(start):
        return []
    paths = []
    for node in graph[start]:
        if node not in path:
            newpaths = find_all_paths(graph, node, end, path)
            for newpath in newpaths:
                paths.append(newpath)
```

```
    return paths

path_result=find_all_paths(graph,'A','O')
s='List of paths='+repr(path_result)
print s
```

- Closeness for any node x is defined as C(x) = $N / \sum_{y} d(y,x)$, where N is the number of nodes and d(y,x) is the distance between y and x.

```
def shortestPath(result):
    min=10000
    minPath=[]
    for eachPath in result:
        if(len(eachPath)<min):
            min=len(eachPath)
            minPath=eachPath
    return minPath

def allPairsShortestPath(graph,theNode):
    theSum=0
    for eachNode in graph:
        if(eachNode!=theNode):
            result=find_all_paths(graph,theNode,eachNode)
            shortest=shortestPath(result)
            if(len(shortest)>0):
                theSum+=(len(shortest)-1)
    return theSum

def calcCloseness(graph,x):
    sum_n=allPairsShortestPath(graph,x)
    if(sum_n==0):
        s='Closeness of node '+x+' is infinity'
    else:
        close=((float)(len(graph))/(float)(sum_n))
        s='Closeness of node '+x+' is '+repr(close)
    print s

def allCloseness(graph):
    for eachNode in graph:
        calcCloseness(graph,eachNode)

calcCloseness(graph,'A')
calcCloseness(graph,'B')
```

- Can you plot the graph and display it on screen? (2 marks extra)
  Yes.

```
xPointsDict = {'O': 4, 'Z': 0, 'A': 1, 'T': 0, 'L': 3, 'M': 4, 'D': 4, 'C': 6,
               'R': 5, 'S': 5, 'F': 8, 'P': 7, 'G': 10, 'B':11, 'U':12, 'E': 14, 'H': 14,
               'V': 14, 'I': 13, 'N': 12}
yPointsDict = {'O': 12, 'Z': 11, 'A': 10, 'T': 8, 'L': 7, 'M': 5, 'D': 4, 'C': 3,
               'R': 8, 'S': 9, 'F': 10, 'P': 7, 'G': 3, 'B':5, 'U':7, 'E': 4,
```

```
'H': 7,'V': 10, 'I': 11, 'N': 12}


xPoints = list(xPointsDict.values())
yPoints = list(yPointsDict.values())


plt.plot(xPoints, yPoints, 'ro')
for n in xPointsDict.keys():
    plt.annotate(n, (xPointsDict[n], yPointsDict[n]))
for mNode in graph.keys():
    for cNode in graph[mNode]:
        plt.plot((xPointsDict[mNode], xPointsDict[cNode]), (yPointsDict[mNode],
yPointsDict[cNode]))

plt.show()
```

Screenshots

```
('Degree of ', 'A', ' is ', 3)
('Degree of ', 'C', ' is ', 3)
('Degree of ', 'B', ' is ', 4)
('Degree of ', 'E', ' is ', 1)
('Degree of ', 'D', ' is ', 2)
('Degree of ', 'G', ' is ', 1)
('Degree of ', 'F', ' is ', 2)
('Degree of ', 'I', ' is ', 2)
('Degree of ', 'H', ' is ', 2)
('Degree of ', 'M', ' is ', 2)
('Degree of ', 'L', ' is ', 2)
('Degree of ', 'O', ' is ', 2)
('Degree of ', 'N', ' is ', 1)
('Degree of ', 'P', ' is ', 3)
('Degree of ', 'S', ' is ', 3)
('Degree of ', 'R', ' is ', 3)
('Degree of ', 'U', ' is ', 2)
('Degree of ', 'T', ' is ', 2)
('Degree of ', 'V', ' is ', 2)
('Degree of ', 'Z', ' is ', 2)
```

```
List of paths=[['A', 'Z', 'O'], ['A', 'S', 'O'], ['A', 'T', 'L', 'M', 'D', 'C', 'P', 'R', 'S', 'O'], ['A', 'T
', 'L', 'M', 'D', 'C', 'P', 'B', 'F', 'S', 'O'], ['A', 'T', 'L', 'M', 'D', 'C', 'R', 'S', 'O'], ['A', 'T', 'L
', 'M', 'D', 'C', 'R', 'P', 'B', 'F', 'S', 'O']]
```

```
Closeness of node A is 0.273972602739726
Closeness of node B is 0.37735849056603776
```

```
    for cNode in graph[mNode]:
        plt.plot((xPointsDict[mNode], xPointsDict[cNode]), (yPointsDict[mNode], yPointsDict[cNode]))
```

```
In [85]: plt.show()
```