Donovan Guelde
PS1
CSCI 5352

1.a. where $A_{i,j} = 1$ implies that there is an edge from node j to node i

	1	2	3	4	5
1	0	0	1	1	0
2	1	0	0	0	0
3	0	1	0	0	1
4	0	0	0	0	1
5	1	0	0	1	0

1.b.

1→{2,5}
2→{3}
3→{1}
4→{1,5}
5→{3,4}

1.c. *top, dark nodes*

	1	2	3	4	5
1	0	1	1	1	1
2	1	0	1	1	1
3	1	0	0	0	0
4	1	1	0	0	0
5	1	1	0	0	0

lower, light nodes

	1	2	3	4	5
1	0	1	1	1	1
2	1	0	1	1	1
3	1	0	0	0	0
4	1	1	0	0	0
5	1	1	0	0	0

2.a. k = A1
2.b. m = k^T[1]
2.c. N = [A^2]
2.d. #triangles = trace([A^3])

3. For one partition of a bipartite network, mean degree c = m/n. We eliminate the 2 from the familiar formula c=2m/n because only one end of an edge connects with one node in each partition. This leads to:

$c_1 = m/n_1$

$c_2 = m/n_2$

where m is the number of edges in the bipartite graph. m carries the same value in each expression. So:

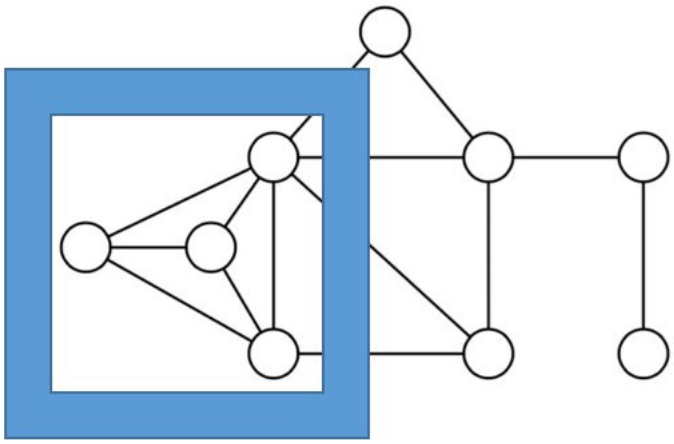
$m = c_1 n_1$

$m = c_2 n_2$

$c_1 n_1 = c_2 n_2$

$c_2 = (n_1/n_2)c_1$

4.a.



Boxed nodes represent a 3-core

4.b.

$$reciprocity = (1/m)\sum_{ij} A_{ij}A_{ji}$$

$$= (1/8)(1+1+1) = 3/8$$

4.c. cosine similarity of vertices A and B in network (C) =

$$\frac{2}{\sqrt{4*5}} = \frac{1}{\sqrt{5}}$$

5. Proof:

Prove by induction that

$$(1)\ n = k(k-1)^{d-1}$$

vertexes can be reached in d steps, where d is a natural number.

Base Case:

When d=1, k nodes are reachable. This follows from the structure of a Cayley tree. From the root node, exactly k nodes are reachable in 1 step. So, when d=1,

$$n = k(k-1)^{d-1} = k(k-1)^0 = k$$

For d=2, $n = k(k-1)^1$, since each node connects to exactly k nodes, inclusive of parent and offspring. In other words, every additional step from root node results in n=n(k-1) reachable nodes.

Induction Step:

Let s be any real number greater than 1, and suppose (1) above holds for d = s. Then:

$$n = k(k-1)^s$$

for s=s+1:

$$n(k-1) = k(k-1)^{(s+1)}$$

$$n = \frac{k(k-1)^{s+1}}{k-1}$$

$$n = k(k-1)^s$$

Thus (1) holds for s = d+1

Conclusion:

By the principal of induction, (1) is true for all d st d is a natural number greater than or equal to one.

Diameter of a Cayley tree is a path from one leaf through the root to any other leaf with no repeating nodes. Diameter = 2*radius+1 = (2(n-1)/k) +1 where n = total number of nodes in the tree, as opposed to reachable nodes as used in proof above.

Cayley trees do exhibit the "small-world" effect. As stated above, the diameter = (2(n-1)/k)+1, and the number of new nodes as the diameter increases = $k(k-1)^{d-1}$
This means that the number of nodes n must increase by $k(k-1)^{d-1}$ in order to increase diameter by 1.
So: $\Delta n = k(k-1)^{d-1}$

$\log \Delta n = (d-1)log(k(k-1))$

$(d-1) = \frac{log\Delta n}{log(k^2-k)}$

$d = \frac{log\Delta n}{log(k^2-k)} + 1$

As n grows larger, diameter d grows as a factor of log(n)

6.a. $MND = (1/2m)\sum_{uv}^{n} k_v A_{uv}$

$= (1/nk_{average})\sum_{uv}^{n} k_v A_{uv}$

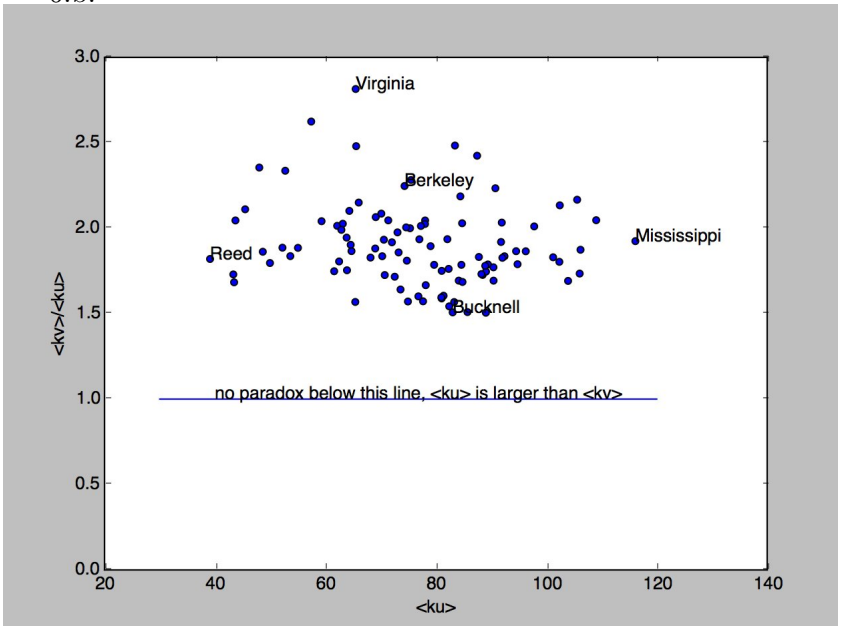
where $\sum_{uv}^{n} k_v A_{uv}$ is the equivalent of matrix multiplication AA, resulting in:

$MND = (1/nk_{average})\sum uvA^2$ where $\frac{\sum A^2}{n} = k^2_{average}$

$= \frac{nk^2_{average}}{nk_{average}}$

$= \frac{k^2_{average}}{k_{average}}$

6.b.



Considering the similarity of the networks in question, as well as their sizes, it is not too surprising that all of the networks included in the dataset exhibit the 'Friendship Paradox.' The ¡kv¿/¡ku¿ value appears to be centered around 1.5 to 2, regardless of other network factors, so there does not appear to be a dependency between MND and mean degree.

6.c. The majority illusion is closely related to the friendship paradox. Considering that, on average, our friends have more friends than we do, many of our friends may be extremely popular hubs in the network, rather than members with an average number of connections. Since the same idea applies to all members of a given network, a random distribution of network members exhibiting property x may cause the majority illusion if, by chance, some of these property-x exhibiting members are also popular hub nodes. While q (the fraction of vertices exhibiting property x) may be low, the probability of a given vertex being connected to a vertex exhibiting x may be high.

```
1  # Author: Donovan Guelde
2  # CSCI 5352 PS1
3  # references: online documentation for numpy
4  # Collaborators: None
5
6  import numpy as np
7  import os
8
9
10
11 class Network:
12  def __init__(self,fileName):
13   self.n = 0
14   self.kAverage = 0
15   self.degreeVector=[]
16   self.associationMatrix = self.readFile(fileName)
17   self.m = np.sum(self.associationMatrix)/2
18   self.mnd = self.getMND()
19  def readFile(self,fileName):
20   with open("./CSCI5352_Data/facebook100txt/"+fileName+"_attr.txt",'r') as f:
21   #uses attr file to initialize a 2d matrix of appropriate size
22    counter=0
23    for line in f:
24     counter+=1
25    associationMatrix = np.empty((counter,counter))
26    self.n = counter-1 #-1 for label row
27    associationMatrix.fill(0)
28   f.close()
29   with open("./CSCI5352_Data/facebook100txt/"+fileName+".txt",'r') as f:
30   #uses data file to build association matrix (assumes simple graph)
31    lines = f.readlines()
32    for line in lines:
33     line = line.split()
34     associationMatrix[int(line[0])][int(line[1])] = 1
35     #make it undirected...
36     associationMatrix[int(line[1])][int(line[0])] = 1
37   f.close()
38   self.degreeVector = np.sum(associationMatrix,axis=0)
39   kAverageSum=0
```

```
40    for index in range (1,self.n+1):
41
42     kAverageSum += self.degreeVector[index]
43    self.kAverage = kAverageSum/self.n
44
45    return associationMatrix
46   def getMND(self):
47    associationMatrixSquared = np.linalg.matrix_power(self.associationMatrix,2)
48    self.mnd = (np.sum(associationMatrixSquared)/np.sum(self.associationMatrix))
49    return self.mnd
50
51  def main():
52   plotArray=np.empty((100,2)) #an array of points to plot
53   nameArray=[""]*100
54   #array to hold names of schools where [index] corresponds to plotArray[index]
55   nextUniversity = [2]
56   lastUniversity = [2]
57   counter=0
58   for file in os.listdir("./CSCI5352_Data/facebook100txt/"):
59    if (file != ".DS_Store"):
60
61     nextFile, fileExtension = os.path.splitext(file)
62     nextUniversity = nextFile.split('_')
63     if (str(nextUniversity[0]) != str(lastUniversity[0])):
64      nextGraph = Network(nextUniversity[0])
65      plotArray[counter][0] = float(nextGraph.mnd)
66      plotArray[counter][1] = float(nextGraph.kAverage)
67
68      nameArray[counter] = str(nextUniversity[0])
69      print nameArray
70      lastUniversity=nextUniversity
71      counter+=1
72   np.savetxt("plotResults.txt",plotArray)
73   with open("nameResults.txt","w") as f:
74    np.savetxt(f,nameArray,fmt='%s')
75
76  main()
```