Part a

```
import numpy as np
import seaborn as sns
import matplotlib.pylab as plt
import pandas as pd
import networkx as nx
import matplotlib.image as mpimg
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.sparse import csr_matrix
from scipy.sparse import diags
from scipy import sparse as sp
import pickle
```

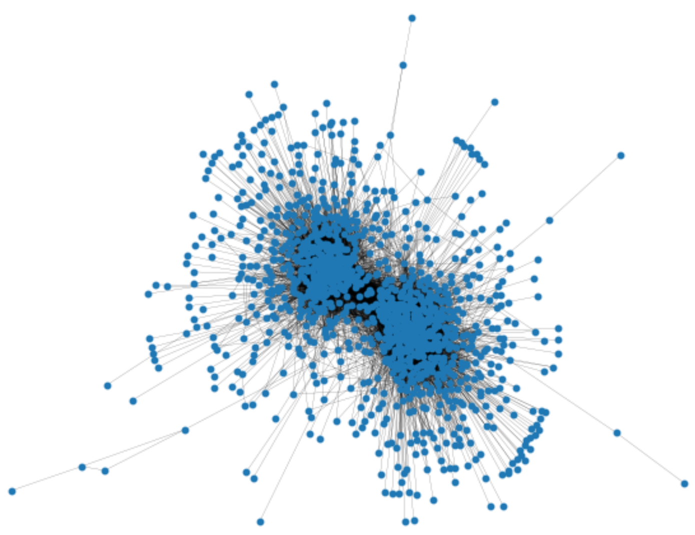
```
network=nx.read_edgelist("/content/asset-v1_UTAustinX+DataSci.312+1T2023+type@asset+block@PoliticalBlogsUndir.txt")
```

```
print(network)
```

```
    Graph with 1222 nodes and 16713 edges
```

```
f = plt.figure(1)

np.random.seed(1)
nx.draw(network,node_size=10,width=0.1)
plt.title( "Network of political blogs" )
plt.show()
```

Network of political blogs



Answer

Number of nodes : 1222
Number of edges : 16713

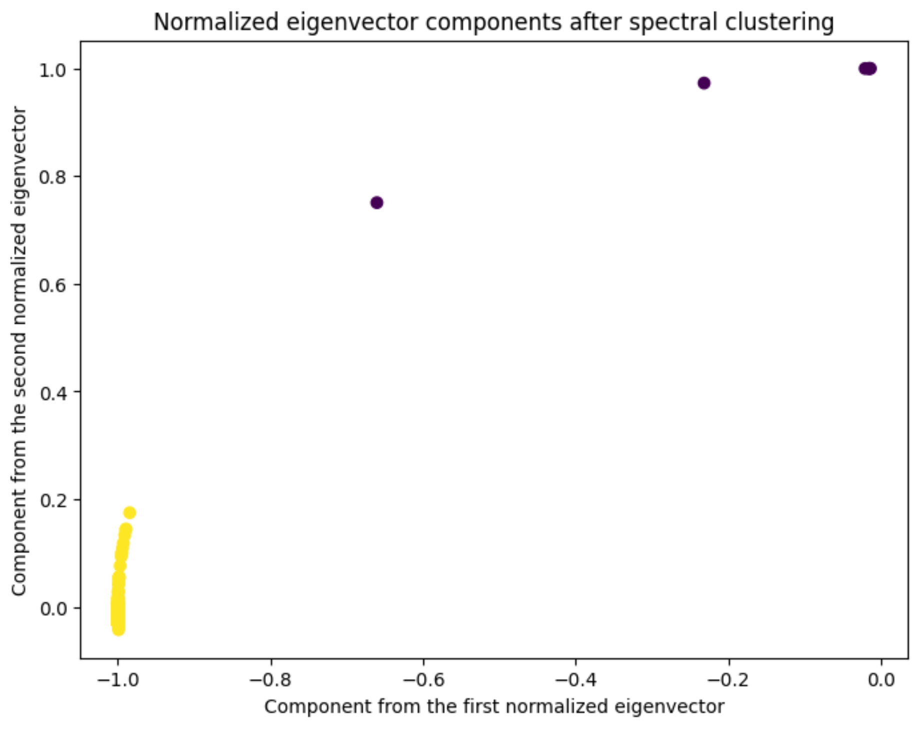
Part b

```
from scipy.spatial.distance import pdist, squareform
from sklearn import metrics
from numpy.linalg import eig
from sklearn.preprocessing import normalize
from scipy import sparse as sp
```

```
def spectral_with_plotting(n_eigen,K):
  for i in range(K.shape[0]):
        K[i,i]=0.0


  D1=np.diag(np.sum(K,axis=1)**(-0.5))

  #D1=sp.csr_matrix(D1)

  #print("Shape D1",D1.shape)
  #print("Shape K",K.shape)
  K1=D1.dot(K).dot(D1)
```

```
    w,v=eig(K1.astype(np.float))
    w=w.real
    v=v.real
    top2=[-20,-22]
    index=[-10,-10]
    for i in range(w.shape[0]):
        if(w[i]>=top2[0]):
          top2[0]=w[i]
          index[0]=i
        if(w[i]>=top2[1] and w[i]<top2[0]):
          top2[1]=w[i]
          index[1]=i
    top=np.array([v[:,index[0]],v[:,index[1]]]).transpose()
    #sum_of_rows = np.sqrt(np.sum(top**2,axis=1))
    normalized_array =normalize(top)
    kmeans = KMeans(n_clusters=n_eigen,init="random").fit(normalized_array)
    p_kmeans=kmeans.predict(normalized_array)
    count_1=0
    count_0=0
    for i in range(p_kmeans.shape[0]):
      if p_kmeans[i]==0:
        count_0=count_0+1
      if p_kmeans[i]==1:
        count_1=count_1+1
    print("In cluster 0 there are",count_0)
    print("In cluster 1 there are",count_1)
    colors=["blue","green","red","purple","orange","yellow","brown","pink"]
    fig1, ax1 = plt.subplots(figsize=(8,6))
    plt.scatter(normalized_array[:,0],normalized_array[:,1],c=p_kmeans)#,cmap=matplotlib.colors.ListedColormap(colors))
    ax1.set_xlabel('Component from the first normalized eigenvector')
    ax1.set_ylabel('Component from the second normalized eigenvector')
    ax1.set_title('Normalized eigenvector components after spectral clustering')

    return p_kmeans
  #ax1.set_title("Spectral clustering, using $\sigma=$"+str(s))


  adj_mat=sp.csr_matrix.toarray(nx.adjacency_matrix(network))


  p_kmeans=spectral_with_plotting(2,adj_mat)
```

```
<ipython-input-59-82212b78cb63>:21: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  w,v=eig(K1.astype(np.float))
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will cha
  warnings.warn(
In cluster 0 there are 6
In cluster 1 there are 1216
```



Answer

How many nodes are in each cluster?

In cluster 0 there are 1216 nodes

In cluster 1 there are 6 nodes

Part c

```
cluster_one=np.where(p_kmeans == 0)[0]+1
cluster_two=np.where(p_kmeans == 1)[0]+1

print(cluster_two)
"""
c_one=[]
```

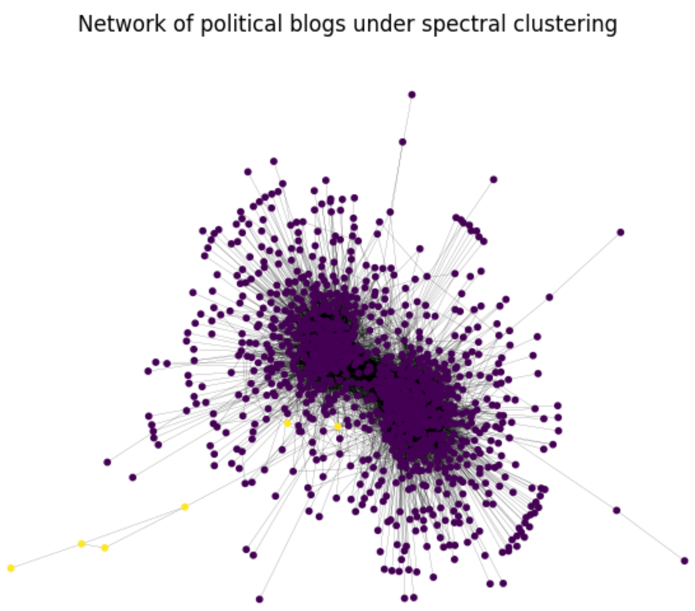
```
c_two=[]

for i in range(cluster_one.shape[0]):
  c_one.append(cluster_one[i])

for i in range(cluster_two.shape[1]):
  c_two.append(cluster_two[i])
"""
```

```
    [ 449  834  914 1212 1213 1221]
    '\nc_one=[]\nc_two=[]\n\nfor i in range(cluster_one.shape[0]):\n  c_one.ap
    pend(cluster_one[i])\n\nfor i in range(cluster_two.shape[1]):\n  c_two.app
    end(cluster two[i])\n'
```

```
print(type(cluster_one))
print(cluster_one)
```

```
    <class 'numpy.ndarray'>
    [   1    2    3 ... 1219 1220 1222]
```

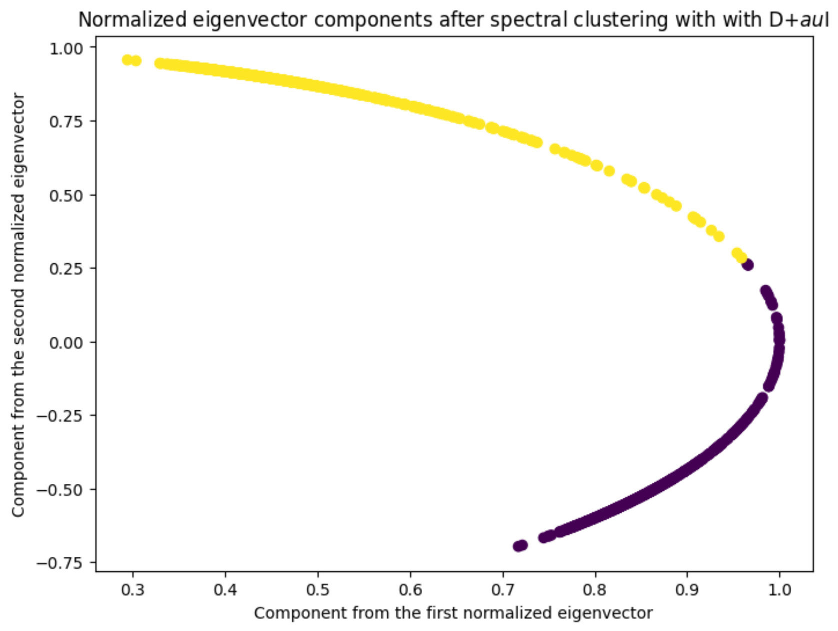
```
np.random.seed(1)
pos=nx.spring_layout(network)

#edges=network.edges()

#print(edges)

f1 = plt.figure(1)

np.random.seed(1)
nx.draw(network,node_color=p_kmeans,node_size=10,width=0.1)
plt.title( "Network of political blogs under spectral clustering" )
plt.show()
#nx.draw_networkx(network,nodelist=cluster_one,pos=pos,node_size=10,node_color="tab:red")
#nx.draw_networkx(network,nodelist=cluster_two,node_size=10,node_color="tab:blue")
```



Part d

```
def spectral_with_plotting_mod(n_eigen,K):
  for i in range(K.shape[0]):
        K[i,i]=0.0


  D1=np.diag(np.sum(K,axis=1)**(-0.5))

  #D1=sp.csr_matrix(D1)

  #print("Shape D1",D1.shape)
  #print("Shape K",K.shape)
  sum_=0
  for i in range(D1.shape[0]):
    sum_=sum_+D1[i,i]
  tau=sum_/D1.shape[0]
  factor=tau*np.identity(D1.shape[0])
  K1=(D1+factor).dot(K).dot(D1+factor)
  print("K1",K1)

  w,v=eig(K1.astype(np.float))
  w=w.real
  v=v.real
  top2=[-20,-22]
  index=[-10,-10]
  for i in range(w.shape[0]):
      if(w[i]>=top2[0]):
        top2[0]=w[i]
        index[0]=i
      if(w[i]>=top2[1] and w[i]<top2[0]):
        top2[1]=w[i]
        index[1]=i
  top=np.array([v[:,index[0]],v[:,index[1]]]).transpose()
  #sum_of_rows = np.sqrt(np.sum(top**2,axis=1))
  normalized_array =normalize(top)
  kmeans = KMeans(n_clusters=n_eigen,init="random").fit(normalized_array)
  p_kmeans=kmeans.predict(normalized_array)
  count_1=0
  count_0=0
  for i in range(p_kmeans.shape[0]):
    if p_kmeans[i]==0:
      count_0=count_0+1
    if p_kmeans[i]==1:
      count_1=count_1+1
  print("In cluster 0 there are",count_0)
  print("In cluster 1 there are",count_1)
  colors=["blue","green","red","purple","orange","yellow","brown","pink"]
  fig1, ax1 = plt.subplots(figsize=(8,6))
  plt.scatter(normalized_array[:,0],normalized_array[:,1],c=p_kmeans)#,cmap=matplotlib.colors.ListedColormap(colors))
  ax1.set_xlabel('Component from the first normalized eigenvector')
  ax1.set_ylabel('Component from the second normalized eigenvector')
  ax1.set_title('Normalized eigenvector components after spectral clustering with with D+${\tau}$I')
  return p_kmeans


p_kmeans_mod=spectral_with_plotting_mod(2,adj_mat)
```

```
K1 [[0.         0.32013334 0.29024668 ... 0.         0.         0.        ]
 [0.32013334 0.         0.         ... 0.         0.         0.        ]
 [0.29024668 0.         0.         ... 0.         0.         0.        ]
 ...
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]]
<ipython-input-55-1341d7cc7d06>:20: DeprecationWarning: `np.float` is a dep
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/
  w,v=eig(K1.astype(np.float))
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: Futu
  warnings.warn(
In cluster 0 there are 554
In cluster 1 there are 668
```



Answer

How many nodes are in each cluster?

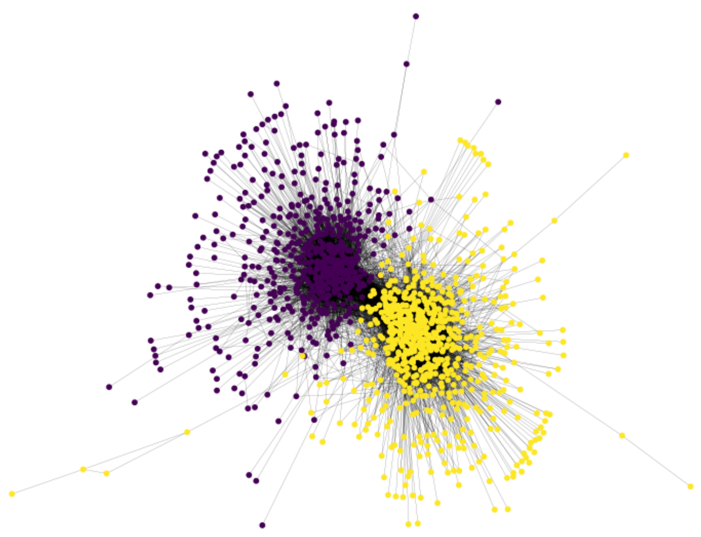
In cluster 0 there are 668

In cluster 1 there are 554

Part e

```
f1 = plt.figure(1,figsize=(8,6))

np.random.seed(1)
nx.draw(network,node_color=p_kmeans_mod,node_size=10,width=0.1)
plt.title(r"Network of political blogs under spectral clustering with D+${\tau}$I" )

plt.show()
```

Network of political blogs under spectral clustering with D+τI



Answer

How well does this method of spectral clustering separate the groups compared to the previous method? **It does a much better work separeting the two groups than the previous method**

✓ 4s completed at 2:24 PM