In [2]:

```
%matplotlib inline
import matplotlib
import numpy as np
import matplotlib.pyplot as plot
from sklearn.cluster import KMeans
```

In [3]:

```
# Reads the adjacency matrix from file
A=np.loadtxt('adjacency.txt')
print("There are",A.shape[0],"nodes.")
```

```
There are 328 nodes.
```

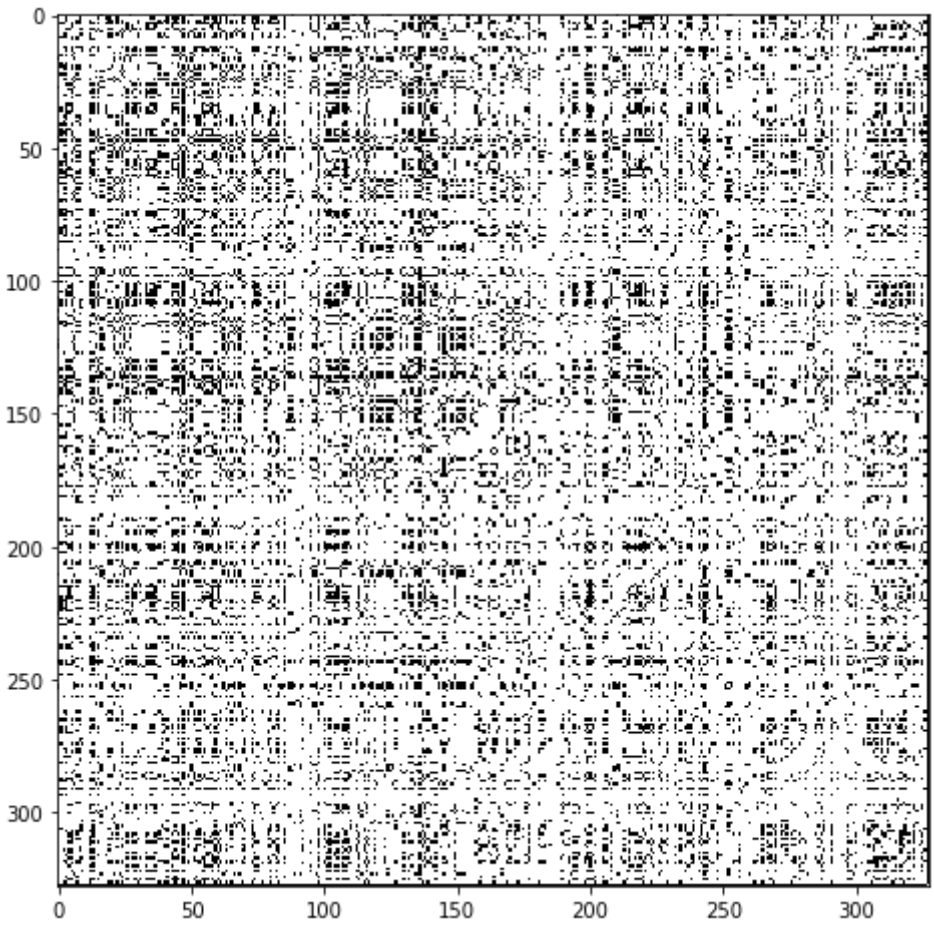
As you can see above, the adjacency matrix is relatively large (328x328): there are 328 persons in the graph. In order to visualize this adjacency matrix, it is convenient to use the 'imshow' function. This plots the 328x328 image where the pixel (i,j) is black if and only if A[i,j]=1.

In [4]:

```
plot.figure(figsize=(8,8))
plot.imshow(A,aspect='equal',cmap='Greys',  interpolation='none')
```

Out[4]:

```
<matplotlib.image.AxesImage at 0x7fe2c5be6f10>
```



a) Construct in the cell below the degree matrix:

$$D_{i,i} = \deg(i) \qquad \text{and} \qquad D_{i,j} = 0 \ \text{ if } i \neq j,$$

the Laplacian matrix:

$$L = D - A$$

and the normalized Laplacian matrix:

$$L_{\text{norm}} = D^{-1/2} L D^{-1/2}.$$

In [5]:

```
d = np.sum(A,axis=0)
D=np.diag(d)
D2=np.diag(1/np.sqrt(d))
L= D2 @ A @ D2
```

b) Using the command 'linalg.eigh' from numpy, compute the eigenvalues and the eigenvectors of L_{norm}.

In [6]:

```
v,w = np.linalg.eigh(L)
```

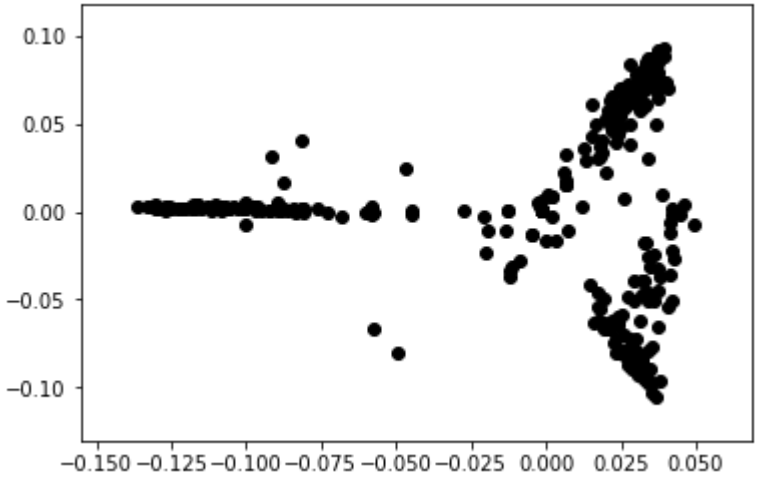
c) We would like to cluster the nodes (i.e. the users) in 3 groups. Using the eigenvectors of L_{norm}, assign to each node a point in R^2, exactly as in 'Algorithm 1' of the notes where you replace L by L_{norm}. Plot these points using the 'scatter' function of matplotlib.

In [9]:

```
X=w[:,-3:-1]
plot.scatter(X[:,1],X[:,0], color='black')
```

Out[9]:

```
<matplotlib.collections.PathCollection at 0x7fe2c5e7ebd0>
```



d) Using the K-means algorithm (use the built-in function from scikit-learn)

In [10]:

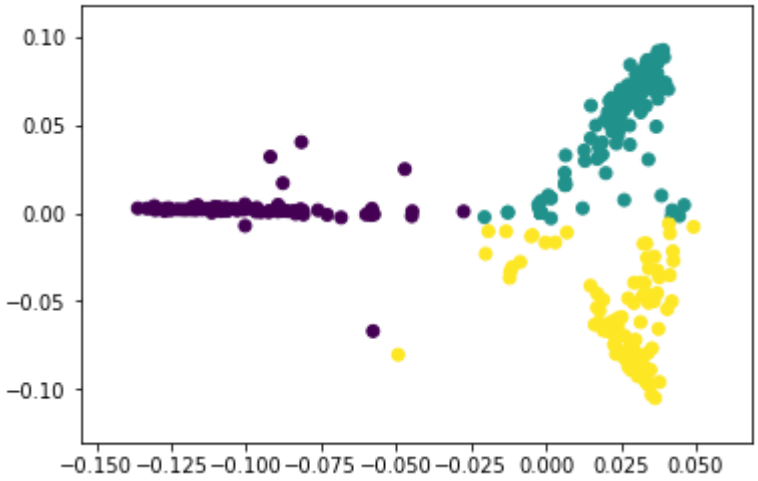
```
# Replace ??? by the matrix of the points computed in (c)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
labels=kmeans.labels_
# labels contains the membership of each node
```

In [12]:

```
# Optional: color the points accorind to their labels, to check that kmeans work
ed well
plot.scatter(X[:,1],X[:,0],c=labels)
```

Out[12]:

```
<matplotlib.collections.PathCollection at 0x7fe2c5e28790>
```



e) Re-order the adjacency matrix according to the clusters computed in the previous question. That is reorder the columns and rows of A to obtain a new adjacency matrix (that represent of course the same graph) such that the n_1 nodes of the first cluster correspond to the first n_1 rows/columns, the n_2 nodes of the second cluster correspond to the next n_2 rows/columns, and the n_3 nodes of the third cluster correspond to the last n_3 rows/columns. Plot the reordered adjacency matrix using 'imshow'.

In [36]:

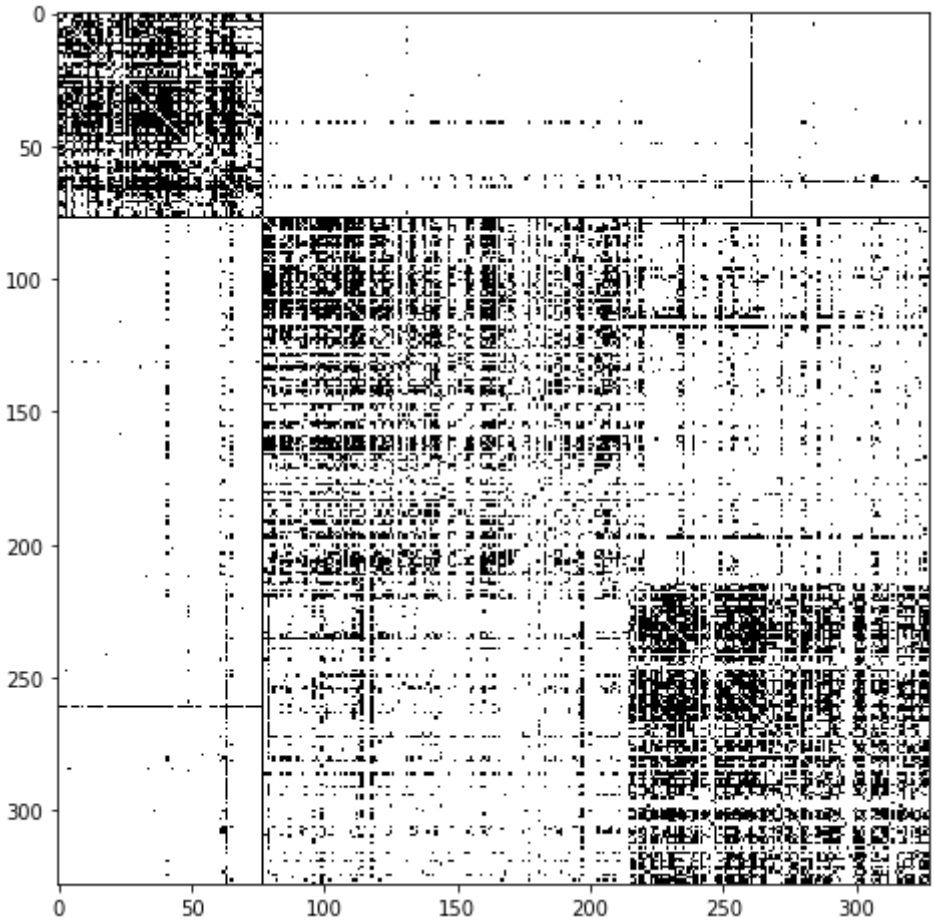
```
N=A.shape[0]
nodes = np.arange(N)
cluster1 = nodes[labels==0]
cluster2 = nodes[labels==1]
cluster3 = nodes[labels==2]
permutation = np.concatenate((cluster1, cluster2, cluster3))
rA=np.zeros((N,N))
for i in range(N):
    for j in range(N):
        rA[i,j]= A[permutation[i],permutation[j]]
```

In [37]:

```
plot.figure(figsize=(8,8))
plot.imshow(rA,aspect='equal',cmap='Greys',  interpolation='none')
```

Out[37]:

```
<matplotlib.image.AxesImage at 0x7fe2c0bf2650>
```



In [ ]: