In [31]:
```
%matplotlib inline
import matplotlib
import numpy as np
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans
```

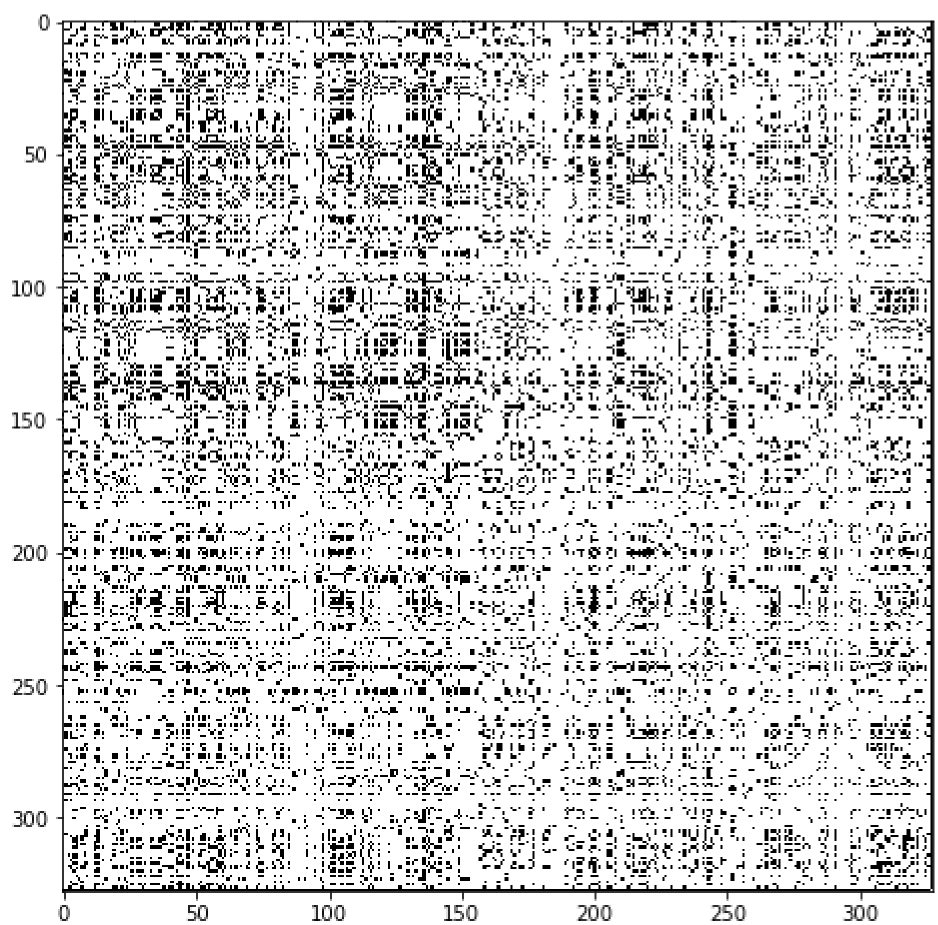
In [32]:
```
# Reads the adjacency matrix from file
A = np.loadtxt('adjacency.txt')
print(f'There are {A.shape[0]} nodes in the graph.')
matrix =A
```

```
There are 328 nodes in the graph.
```

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 [33]:
```
plt.figure(figsize=(8,8))
plt.imshow(A,aspect='equal',cmap='Greys',  interpolation='none')
```

Out[33]: `<matplotlib.image.AxesImage at 0x1b5ae0aad60>`



(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 [58]:

```
matrix = np.matrix(matrix)

d = np.zeros(len(matrix))

column_sum = matrix.sum(axis=0)

#Iterate through and change column sum
for j in range(0, len(matrix)):
    d[j] = column_sum[0,j]

#Initialize diagonal matrix
D = np.diag(d)
d = [1/(x**.5) for x in d]

#Get square roots
D_sqrt = np.diag(d)

#Calculate the Laplacian
L = D - A

#Get the L normalized
Lnorm = D_sqrt@L@D_sqrt
```

In [59]:

```
type(Lnorm)
```

Out[59]: numpy.ndarray

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

In [48]:

```
# Get eigenvectors and eigenvalues
eig_val, eig_vec = np.linalg.eigh(Lnorm)
```

In [49]:
```
Lnorm
```

Out[49]:
```
array([[ 1.        ,  0.        ,  0.        , ...,  0.        ,
         0.        , -0.01064251],
       [ 0.        ,  1.        ,  0.        , ...,  0.        ,
         0.        , -0.00606998],
       [ 0.        ,  0.        ,  1.        , ...,  0.        ,
        -0.01628656, -0.00685914],
       ...,
       [ 0.        ,  0.        ,  0.        , ...,  1.        ,
         0.        , -0.00680698],
       [ 0.        ,  0.        , -0.01628656, ...,  0.        ,
         1.        , -0.00726126],
       [-0.01064251, -0.00606998, -0.00685914, ..., -0.00680698,
        -0.00726126,  1.        ]])
```

(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 $\mathbb{R}^2$, exactly as explained in last lecture (also in 'Algorithm 1' of the notes) where you replace L by L_{norm}. Plot these points using the 'scatter' function of matplotlib.

In [50]:
```
# Reduce dimentionality to R2
E = np.zeros((328,2))
```

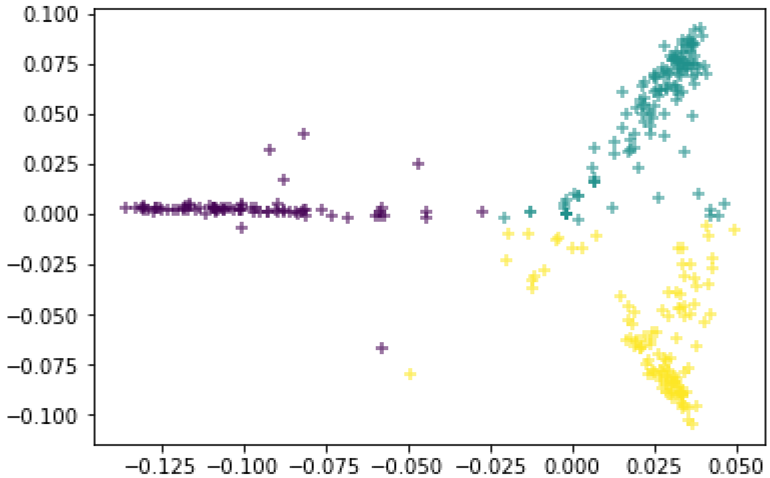
In [51]:
```
#Loop through and assign coordinate values to E
for i in range(len(E)):
    E[i,0] = eig_vec[:,1][i]
    E[i,1] = eig_vec[:,2][i]

E = np.array(E)
```

(d) Using the K-means algorithm (use the built-in function from scikit-learn), cluster the embeddings in $\mathbb{R}^2$ of the nodes in 3 groups.

In [60]:
```
# fit k means
kmeans = KMeans(n_clusters=3, random_state=0).fit(E)
labels=kmeans.labels_
```

In [61]:
```
#Plot scatter plot
plt.scatter(E[:,0], E[:,1], alpha=0.7, marker='+', c = labels)
```

Out[61]: <matplotlib.collections.PathCollection at 0x1b5af89b5e0>

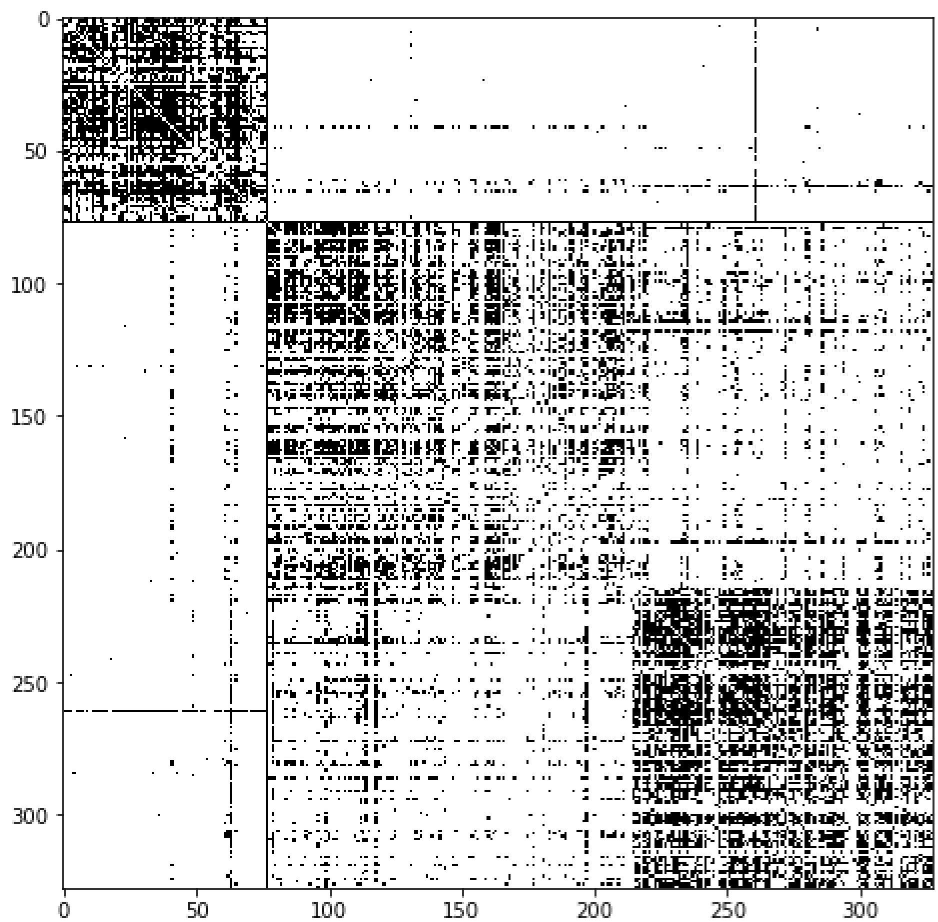


(e) Re-order the adjacency matrix according to the clusters computed in the previous question. That is, reorder the columns and rows of A to obtain a new adjacency matrix (that represents of course the same graph) such that the n_1 nodes of the first cluster correspond to the first n_1 rows/columns, 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 [62]:
```
## Get the index of each point in each cluster
zero_cluster = np.where(labels==0)
one_cluster = np.where(labels==1)
two_cluster = np.where(labels==2)

#Combine the seperated clusters into one groups
new_list = zero_cluster[0].tolist() + one_cluster[0].tolist() + two_cluster[0

#New matrix A
new_matrix = np.zeros((328,328))


#Take original matrix indices and assign to the regrouped values from new lis
for i in range(0,328):
    for j in range(0,328):
        new_matrix[i,j] = A[new_list[i],new_list[j]]
```

In [63]:
```
#Plot the figure
plt.figure(figsize=(8,8))
plt.imshow(new_matrix,aspect='equal',cmap='Greys',  interpolation='none')
```

Out[63]: <matplotlib.image.AxesImage at 0x1b5adfff5b0>



In [ ]:

In [ ]:

```

```