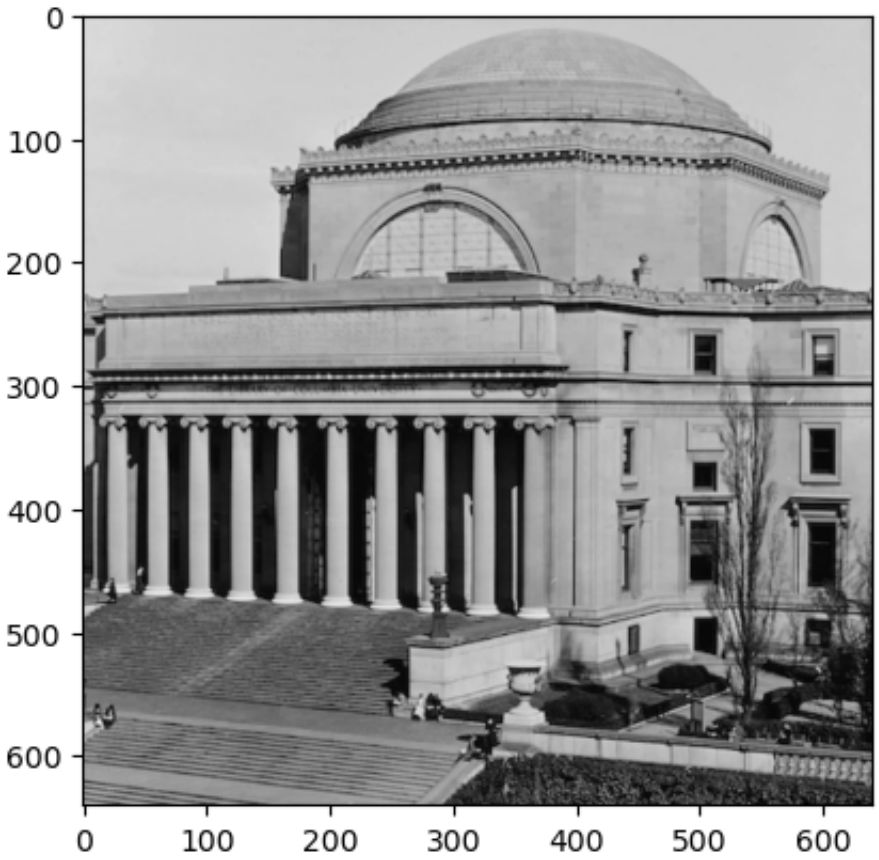
In [41]:
```
import numpy as np

from PIL import Image
from urllib.request import urlopen

image_data = Image.open(urlopen("https://raw.githubusercontent.com/changyaoc
X = np.array(image_data)
```

In [42]:
```
import matplotlib.pyplot as plt

plt.figure()
plt.imshow(image_data, cmap="gray")
# plt.imshow(image_data, cmap="gray")
# can also use - plt.show()
```

Out[42]:
```
<matplotlib.image.AxesImage at 0x11edb3bd0>
```



In [43]:
```
# find the value of the element at index (128,128), that is X[128,128]

X = np.array(image_data)
value = X[128,128]
print(f"The value of the element at index (128, 128) is: {value}")
```

```
The value of the element at index (128, 128) is: 194
```

In [44]:
```
# perform standardization on matrix X to create a new matrix, Xscaled, what
# standardizatoion: for each column in matrix X, subtract its mean and divic

mean_values = np.mean(X, axis = 0)
standard_deviation_values = np.std(X, axis = 0)

# avoid division by 0 -> replace zero standard deviations with 1
standard_deviation_values[standard_deviation_values ==0] = 1

# standardize each column in matrix X
Xscaled = (X - mean_values) / standard_deviation_values

# acess the element at index (128,128) in the new, standardized matrix
scaled_value = Xscaled[128,128]

print(f"The value of the element Xscaled[128, 128] is: {scaled_value}")
```

```
The value of the element Xscaled[128, 128] is: 0.9937429746312681
```

In [45]:
```
# perform PCA (Principal Component Analysis) on this matrix Xscaled -> what

from sklearn.decomposition import PCA

# create a PCA object
pca = PCA()

# fit the PCA model to the standardized matrix Xscaled
pca.fit(Xscaled)

# access the first principle component
first_principle_component = pca.components_[0]

# obtain the value of the first element of the first principal component
first_element = first_principle_component[0]

print(f"The value of the first element of the first principal component is:
```

```
The value of the first element of the first principal component is: -0.04050
88355734232
```

In [47]:
```
# if we only use the first 50 principle components of Xscaled to reconstruct

components = 50

# reconstruct the matrix
X_reconstructed = np.dot(pca.components_[:components, :], Xscaled.T).T

# calculate reconstruction error
reconstruction_error = np.mean((Xscaled - X_reconstructed) ** 2)

print(f"The reconstruction error when only using the first 50 principle comp
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[47], line 9
      6 X_reconstructed = np.dot(pca.components_[:components, :], Xscaled.T)
.T
      8 # calculate reconstruction error
----> 9 reconstruction_error = np.mean((Xscaled - X_reconstructed) ** 2)
     11 print(f"The reconstruction error when only using the first 50 princi
ple components is: {reconstruction_error}")

ValueError: operands could not be broadcast together with shapes (640,640) (
640,50)
```

In [ ]:
```
# do not understand this error
```

In [38]:
```
# try using professor's code (PCA - week 7) - much quicker
```

In [29]:
```
X[128,128]
```

Out[29]:
```
194
```

In [28]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled[128,128]
```

Out[28]:
```
0.9937429746312681
```

In [32]:
```
# perform SVD
n = len(X_scaled)
U, S, Vh = np.linalg.svd(X_scaled.T @ X_scaled / n)
Vh[0,0]
```

Out[32]:
```
-0.04050883557342325
```

In [36]:
```
# calculate reconstruction error
U_reduced = U[:, :50]
Z = X_scaled @ U_reduced
X_approx = Z @ U_reduced.T
error = (
    np.sum(np.square(np.linalg.norm((X_scaled - X_approx), ord=2, axis=1)))
    / np.sum(np.square(np.linalg.norm(X_scaled, ord=2, axis=1)))
)
error
```

Out[36]:
```
0.04240254839520039
```