Problem 3

Implementing MDS for n_components = 1. Comparing the distances between the built in PCA and the implemented MDS, they both match.

In [185]:
```
import math
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
x = [[0,0],
    [3, 0],
    [3,4]]
distances = [
    [0, 3, 5],
    [3, 0, 4],
    [5, 4, 0]
]
pca = PCA(n_components=1)
norm_x = x - np.mean(x, 0)
pca_components = pca.fit(norm_x)
vec = pca.components_
print(vec)
dist = np.dot(norm_x, vec.T).transpose()
# Compare distances to implemented MDS below
print(dist)
```

```
[[0.49806073 0.86714215]]
[[-2.15231099 -0.65812881  2.8104398 ]]
```

In [158]:
```
import numpy as np

def mds(D):
    n = len(D)
    # centers matrix
    H = np.eye(n) - np.ones((n, n))/n
    # YY^T, squared distances
    B = -H.dot(np.square(D)).dot(H)/2
    # Diagonalize
    evals, evecs = np.linalg.eig(B)
    # find largest eigenvalue (abs val)
    m = -1000000
    ind = 0
    for i in range(len(evals)):
        if abs(evals[i]) > 0.001:
            if abs(evals[i]) > m:
                m = abs(evals[i])
                ind = i
    # obtain the largest eigenvalue & corresponding eigenvector
    # note: this is for n_components = 1
    evals = m
    evecs = evecs[:,ind]

    return evecs, evals

evecs, evals = mds(np.matrix(distances))
mds_res = evecs.transpose() * np.sqrt(evals)
# same distances as built in PCA above.
print(mds_res)
```

```
[[-2.15231099 -0.65812881  2.8104398 ]]
```