Programming Assignment 2

Spectral Clustering on Star War

機械四 b07502057 李婷穎

1. eigenvectors

先將給定的 csv 檔轉換成 adjacent matrix，利用 adjacent matrix 各個點的連結數計算出 diagonal degree matrix，此為一對角矩陣，再由後者減去前者計算出 ratio-cut Laplacian L。接著利用 numpy 中 linear algebra 的函式算出前三小的 eigenvalues 以及 eigenvectors，並將後者輸出成 eigen.csv 檔。

2. kmeans

因為最小 eigenvalue 的 eigenvector 為常數，因此省略此項並利用剩下兩個倒數最小的 eigenvectors 各點的作為特徵值來進行 k-means 的計算。
計算 k-means 時有幾步驟：（ k = 3 ）

一：先隨機取三個點當作中心點

二：將所有的點分成三群（與其距離最短的中心點）

三：重新計算各個族群的中心（重心）

接著重複步驟二、三直到中心點不再變動為止。

根據步驟二定義一 cluster 函數，此函數計算出各點與各個中心點的距離，三個中心點代表三個族群，並將其分配給其最小距離的中心點，依此分成三個族群，回傳值為一二維陣列，每一項內的數值代表每個族群內各個的 node 編號。

```
def cluster (k, X, C):
    community = []
    for i in range(k):
        community.append([])
    for i in range(69):
        mindis = 100000000
        for j in range(k):
            dis = ((X[i][0] - C[j][0]) ** 2 + (X[i][1] - C[j][1]) ** 2 )
            if dis < mindis:
                mindis = dis
                t = j
        community[t].append(i)
    return community
```

根據步驟三定義一 centroid 函數，此函數計算出各個族群的中心點，也就是重

心，並將其作為新的中心點來進行下一步驟地重新分群，回傳三個新的中心點值。

```
def centroid (k, X, community):
    center = []
    for i in range(k):
        center.append([])
    for i in range(k):
        l = len(community[i])
        if l == 0:
            l = 1
        center[i] = sum(X[community[i]]) / l
    return center
```

接著為 kmeans 的函數，先經過 cluster 計算出重新分配好的群集，再放入 centroid 計算各個群集的新中心點，若此新的中心點與輸入的中心點相同，則代表已經收斂，若否，則繼續進行下一次的 kmeans 計算。而因為要輸出的 result.csv 檔內要記錄的為各個 node 所屬的陣營編號，因此需再將原先的紀錄各個陣營中所有點編號的 community 陣列轉換成紀錄各點所屬陣營的 m 陣列，並回傳 m 陣列。

```
def kmeans(k, X, C):
    community = cluster(k, X, C)
    center = centroid(k, X, community)
    new = np.array(center)
    m = [0]*69
    for i in range(3):
        for item in community[i]:
            m[item] = i
    if (np.array_equal(C, new) == False ):
        kmeans(k, X, new)
    return m
```

因為一開始要有初始的中心點值，因此在所有 69 個 node 中隨機取三個作為中心點並放入 kmeans 函數開始計算。最後將各點依照所屬陣營以不同顏色的形式繪製成圖形來觀看分群結果。