In [ ]:
```
import pandas as pd
import numpy as np
import networkx as nx
from numpy.linalg import norm
```

In [ ]:
```
# # from colab run this
# !git clone https://github.com/gru13/lncRNA_Disease_association.git
# df_lnc_die = pd.read_csv(r"./lncRNA_Disease_association/datasets/lncrnaDiesease3/website_al
# df_lnc_mir = pd.read_csv(r"./lncRNA_Disease_association/datasets/LncRNASNP/Homo_sapiens_lnc
# df_mir_die = pd.read_csv(r"./lncRNA_Disease_association/datasets/rnaDisease/RNADisease_RNA-
```

In [ ]:
```
# from local system

df_lnc_die = pd.read_csv(r"./../datasets/lncrnaDiesease3/website_alldata_p1.csv")
df_lnc_mir = pd.read_csv(r"./../datasets/LncRNASNP/Homo_sapiens_lncRNA_miRNA_interactions.csv
df_mir_die = pd.read_csv(r"./../datasets/rnaDisease/RNADisease_RNA-disease_miRNA_predict.csv"
```

In [ ]:
```
df_lnc_die.head(),df_lnc_mir.head(),df_mir_die.head()
```

Out[ ]:
```
(        lncRNA                    Disease  PubMed ID
 0  ARHGAP5-AS1  Carcinoma, Hepatocellular   36354136
 1       HOTTIP               Osteosarcoma   33475442
 2       HOTTIP                     Glioma   35402278
 3       HOTTIP             Retinoblastoma   33784880
 4       HOTTIP          stomach carcinoma   32633359,
              lncRNA           miRNA
 0  NONHSAT000002.2  hsa-miR-15a-5p
 1  NONHSAT000002.2  hsa-miR-15a-3p
 2  NONHSAT000002.2   hsa-miR-16-5p
 3  NONHSAT000002.2   hsa-miR-21-3p
 4  NONHSAT000002.2   hsa-miR-28-5p,
           miRNA                    Disease
 0  hsa-let-7a-1  Abdominal aortic aneurysm
 1  hsa-let-7a-2  Abdominal aortic aneurysm
 2  hsa-let-7a-3  Abdominal aortic aneurysm
 3    hsa-let-7c  Abdominal aortic aneurysm
 4    hsa-let-7d  Abdominal aortic aneurysm)
```

In [ ]:
```
dataset1 = pd.merge(df_lnc_mir,df_lnc_die,on=['lncRNA'],how='inner')
dataset2 = pd.merge(df_lnc_mir,df_mir_die,on=['miRNA'],how='inner')
```

In [ ]:
```
dataset1
```

Out[ ]:

	lncRNA	miRNA	Disease	PubMed ID
0	NONHSAT000612.2	hsa-miR-570-3p	Osteoarthritis	34780784
1	NONHSAT000612.2	hsa-miR-651-3p	Osteoarthritis	34780784
2	NONHSAT000612.2	hsa-miR-449b-3p	Osteoarthritis	34780784
3	NONHSAT000612.2	hsa-miR-1468-3p	Osteoarthritis	34780784
4	NONHSAT000612.2	hsa-miR-874-5p	Osteoarthritis	34780784
...	...	...	...	...
590	NONHSAT015292.2	hsa-miR-9902	Osteoarthritis	34780784
591	NONHSAT015292.2	hsa-miR-10522-5p	Osteoarthritis	34780784
592	NONHSAT015292.2	hsa-miR-10526-3p	Osteoarthritis	34780784
593	NONHSAT015292.2	hsa-miR-9851-3p	Osteoarthritis	34780784
594	NONHSAT015292.2	hsa-miR-12118	Osteoarthritis	34780784

595 rows × 4 columns

In [ ]:

```
dataset2
```

Out[ ]:

	lncRNA	miRNA	Disease
0	NONHSAT000002.2	hsa-miR-15a-5p	Abortion habitual
1	NONHSAT000002.2	hsa-miR-15a-5p	Acoustic neuroma
2	NONHSAT000002.2	hsa-miR-15a-5p	Acquired immunodeficiency syndrome
3	NONHSAT000002.2	hsa-miR-15a-5p	Acute coronary syndrome
4	NONHSAT000002.2	hsa-miR-15a-5p	Acute kidney failure
...	...	...	...
9376652	NONHSAT017163.2	hsa-miR-523-3p	Intellectual disability
9376653	NONHSAT017163.2	hsa-miR-523-3p	Leukemia
9376654	NONHSAT017163.2	hsa-miR-523-3p	Neurodegenerative diseases
9376655	NONHSAT017163.2	hsa-miR-523-3p	Skin disease
9376656	NONHSAT017163.2	hsa-miR-523-3p	Vascular diseases

9376657 rows × 3 columns

In [ ]:

```
df = dataset2.copy()
df
```

Out[ ]:

	lncRNA	miRNA	Disease
0	NONHSAT000002.2	hsa-miR-15a-5p	Abortion habitual
1	NONHSAT000002.2	hsa-miR-15a-5p	Acoustic neuroma
2	NONHSAT000002.2	hsa-miR-15a-5p	Acquired immunodeficiency syndrome
3	NONHSAT000002.2	hsa-miR-15a-5p	Acute coronary syndrome
4	NONHSAT000002.2	hsa-miR-15a-5p	Acute kidney failure
...	...	...	...
9376652	NONHSAT017163.2	hsa-miR-523-3p	Intellectual disability
9376653	NONHSAT017163.2	hsa-miR-523-3p	Leukemia
9376654	NONHSAT017163.2	hsa-miR-523-3p	Neurodegenerative diseases
9376655	NONHSAT017163.2	hsa-miR-523-3p	Skin disease
9376656	NONHSAT017163.2	hsa-miR-523-3p	Vascular diseases

9376657 rows × 3 columns

In [ ]:

```
df = dataset2.copy()
```

In [ ]:

```
# # only run this for first time
# disease = df['Disease'].unique()
# lncRNA = df['lncRNA'].unique()
# miRNA = df['miRNA'].unique()

# np.savetxt("./disease.txt",disease,delimiter=',',fmt='%s')
# np.savetxt("./lncRNA.txt",lncRNA,delimiter=',',fmt='%s')
# np.savetxt("./miRNA.txt",miRNA,delimiter=',',fmt='%s')
```

In [ ]:

```
disease = [a[:-1] for a in open("./disease.txt").readlines()]
lncRNA = [a[:-1] for a in open("./lncRNA.txt").readlines()]
miRNA = [a[:-1] for a in open("./miRNA.txt").readlines()]
```

In [ ]:

```
# len(df['lncRNA'].unique()),len(df['miRNA'].unique()),len(df['Disease'].unique()),
```

CReation of the graph that contains the totaly connected with miRNA, lncRNA, Disease

In [ ]:

```
LM_Graph = nx.Graph()
MD_Graph = nx.Graph()
LD_Graph = nx.Graph()

LM_Graph.add_nodes_from(lncRNA)
LM_Graph.add_nodes_from(miRNA)

LD_Graph.add_nodes_from(lncRNA)
LD_Graph.add_nodes_from(disease)

MD_Graph.add_nodes_from(miRNA)
MD_Graph.add_nodes_from(disease)
```

In [ ]:

```
for a in df.values:
    LM_Graph.add_edge(a[0],a[1])
    MD_Graph.add_edge(a[1],a[2])
    LD_Graph.add_edge(a[0],a[2])
```

CReation of the Lncrna-disease , Lncrna-miRNA, miRNA-Disease assoction matrix (LM,LD,MD)

In [ ]:

```
LM_index = {a:b for (a,b) in zip(LM_Graph.nodes(),range(len(LM_Graph.nodes())))}
MD_index = {a:b for (a,b) in zip(MD_Graph.nodes(),range(len(MD_Graph.nodes())))}
LD_index = {a:b for (a,b) in zip(LD_Graph.nodes(),range(len(LD_Graph.nodes())))}
```

In [ ]:

```
LM_M_A = nx.adjacency_matrix(LM_Graph).todense()
MD_M_A = nx.adjacency_matrix(MD_Graph).todense()
LD_M_A = nx.adjacency_matrix(LD_Graph).todense()
```

In [ ]:

```
LM_M = np.zeros(shape=(len(lncRNA),len(miRNA)),dtype=int)
MD_M = np.zeros(shape=(len(miRNA), len(disease)),dtype=int)
LD_M = np.zeros(shape=(len(lncRNA), len(disease)),dtype=int)
```

In [ ]:

```
for a in range(len(lncRNA)):
    for b in range(len(miRNA)):
        LM_M[a][b] = LM_M_A[LM_index[lncRNA[a]]][LM_index[miRNA[b]]]
LM_M
```

Out[ ]:

```
array([[1, 1, 1, ..., 0, 0, 0],
       [1, 1, 1, ..., 0, 0, 0],
       [1, 0, 1, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

In [ ]:

```
for a in range(len(miRNA)):
    for b in range(len(disease)):
        MD_M[a][b] = MD_M_A[MD_index[miRNA[a]]][MD_index[disease[b]]]
MD_M
```

Out[ ]:

```
array([[1, 1, 1, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [1, 1, 1, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

In [ ]:

```
LD = LM_M.dot(MD_M)
LD
```

Out[ ]:

```
array([[5, 6, 6, ..., 0, 0, 0],
       [3, 5, 5, ..., 0, 0, 0],
       [7, 9, 7, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

In [ ]:

```
# # run this if CD.csv is not found in the current folder
# CD = np.zeros(shape=(len(disease),len(disease)))
# for i in range(len(disease)):
#     for j in range(len(disease)):
#         CD[i][j] = (LD[:,i].dot(LD[:,j]))/(norm(LD[:,i])*norm(LD[:,j]))
# np.savetxt("CD.csv",CD,delimiter=',',fmt='%.5f')
```

In [ ]:

```
CD = np.loadtxt("CD.csv",delimiter=',',dtype=float)
```

In [ ]:

```
# # run this if CL.csv not found in the current folder
# CL = np.zeros(shape=(len(lncRNA),len(lncRNA)))
# for i in range(len(lncRNA)):
#     for j in range(len(lncRNA)):
```

```
#           CL[i][j] = (LD[i,:].dot(LD[j,:]))/(norm(LD[i,:])*norm(LD[j,:]))
# np.savetxt("CL.csv",CL,delimiter=',',fmt="%.5f")
```

In [ ]:
```
CL = np.loadtxt("./CL.csv",delimiter=',',dtype=float)
```

In [ ]:
```
JD = np.zeros(shape=CD.shape)
for i in range(MD_M.shape[-1]):
    for j in range(MD_M.shape[-1]):
        JD[i][j] =(np.bitwise_and(MD_M[:,i],MD_M[:,j]).sum())/(np.bitwise_or(MD_M[:,i],MD_M[:
```

In [ ]:
```
JL = np.zeros(shape=CL.shape)
for i in range(LM_M.shape[-1]):
    for j in range(LM_M.shape[-1]):
        JL[i][j] =(np.bitwise_and(LM_M[:,i],LM_M[:,j]).sum())/(np.bitwise_or(LM_M[:,i],LM_M[:
JL
```

Out[ ]:
```
array([[1.        , 0.28849315, 1.        , ..., 0.        , 0.        ,
         0.        ],
       [0.28849315, 1.        , 0.28849315, ..., 0.        , 0.        ,
         0.        ],
       [1.        , 0.28849315, 1.        , ..., 0.        , 0.        ,
         0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
         0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
         0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
         0.        ]])
```

In [ ]:
```
IDS = np.zeros(shape=CD.shape)
for i in range(IDS.shape[0]):
    for j in range(IDS.shape[-1]):
        if CD[i][j] == 0:
            IDS[i][j] = JD[i][j]
        else:
            IDS[i][j] = (CD[i][j]+JD[i][j])/2
IDS
```

Out[ ]:
```
array([[1.        , 0.81975833, 0.73970227, ..., 0.17259   , 0.17259   ,
         0.150645  ],
       [0.81975833, 1.        , 0.85354136, ..., 0.24611955, 0.24611955,
         0.14695   ],
       [0.73970227, 0.85354136, 1.        , ..., 0.17002   , 0.17002   ,
         0.14324   ],
       ...,
       [0.17259   , 0.24611955, 0.17002   , ..., 1.        , 1.        ,
         0.091245  ],
       [0.17259   , 0.24611955, 0.17002   , ..., 1.        , 1.        ,
         0.091245  ],
       [0.150645  , 0.14695   , 0.14324   , ..., 0.091245  , 0.091245  ,
         1.        ]])
```

In [ ]:
```
ILS = np.zeros(shape=CL.shape)
for i in range(ILS.shape[0]):
    for j in range(ILS.shape[-1]):
        if CL[i][j] == 0:
            ILS[i][j] = JL[i][j]
        else:
            ILS[i][j] = (CL[i][j]+JL[i][j])/2
ILS
```

Out[ ]:

```
array([[1.        , 0.63924158, 0.963005  , ..., 0.31482   , 0.4182    ,
        0.282695  ],
       [0.63924158, 1.        , 0.62178658, ..., 0.298965  , 0.401295  ,
        0.269725  ],
       [0.963005  , 0.62178658, 1.        , ..., 0.253995  , 0.330115  ,
        0.2033    ],
       ...,
       [0.31482   , 0.298965  , 0.253995  , ..., 0.5       , 0.27778   ,
        0.175035  ],
       [0.4182    , 0.401295  , 0.330115  , ..., 0.27778   , 0.5       ,
        0.303395  ],
       [0.282695  , 0.269725  , 0.2033    , ..., 0.175035  , 0.303395  ,
        0.5       ]])
```

Latent Factor model

In [ ]:

```
def latent_factor_model(ALM, AMD, k, alpha=2*10**-6, lmbda=4*10**-5, max_iter=1000):
    """
    Latent factor model for calculating lncRNA-disease associations.

    Args:
        ALM (numpy.ndarray): Adjacency matrix of lncRNA-miRNA associations (m x n).
        AMD (numpy.ndarray): Adjacency matrix of miRNA-disease associations (n x e).
        k (int): Number of latent factors.
        alpha (float): Learning rate for gradient descent.
        lmbda (float): Regularization parameter.
        max_iter (int): Maximum number of iterations for gradient descent.

    Returns:
        X (numpy.ndarray): lncRNA feature matrix (m x k).
        Y (numpy.ndarray): Disease feature matrix (e x k).
        psi (numpy.ndarray): lncRNA-disease association score matrix (m x e).
    """



    # Calculate the preliminary lncRNA-disease association matrix
    ALD = ALM @ AMD

    m = ALD.shape[0]
    e = ALD.shape[1]

    # Initialize feature matrices randomly
    X = np.random.rand(m, k)
    Y = np.random.rand(e, k)

    for _ in range(max_iter):
        # Calculate the lncRNA-disease association score matrix
        psi = X @ Y.T

        # Calculate the loss function and gradients
        loss = np.sum((psi - ALD) ** 2) + lmbda * (np.sum(norm(X)) + np.sum(norm(Y)))
        grad_X = 2 * (psi - ALD) @ Y + 2 * lmbda * X
        grad_Y = 2 * (psi - ALD).T @ X + 2 * lmbda * Y

        # Update feature matrices using gradient descent
        X -= alpha * grad_X
        Y -= alpha * grad_Y

    return X, Y, psi
X,Y,phi = latent_factor_model(LM_M,MD_M,213,max_iter=500)
X,Y,phi
```

```
Out[ ]: (array([[ 0.71844427,  0.64993427,  1.00394431, ...,  1.11543469,
          0.3881249 ,  0.6779231 ],
        [ 0.32987549,  0.45956764,  0.77244734, ...,  0.71035878,
          0.58338387,  0.64630339],
        [ 0.13258212,  0.75861352,  0.03481279, ...,  0.36921492,
          0.87155001,  0.31143265],
        ...,
        [ 0.16687526,  0.24092648,  0.28832576, ..., -0.157413  ,
          0.26234988,  0.43437924],
        [ 0.809285  ,  0.48125953, -0.01465503, ...,  0.14616748,
          0.70565643,  0.08476818],
        [ 0.33844405,  0.38182429,  0.22363135, ...,  0.75503067,
          0.10509675,  0.04646025]]),
 array([[ 0.15447941, -0.00348781,  0.1486679 , ...,  0.48057782,
          0.15906654, -0.14522516],
        [ 0.24836356,  0.03967689,  0.33132264, ...,  0.5323393 ,
          0.04829386,  0.04102722],
        [ 0.08263745, -0.02402689,  0.11290417, ...,  0.65893284,
          0.19192877, -0.10970896],
        ...,
        [ 0.04447496, -0.06654206, -0.05462629, ...,  0.09249866,
         -0.04465434,  0.06984064],
        [-0.02106483,  0.02739569,  0.06098578, ..., -0.09407618,
          0.10685792,  0.15673478],
        [-0.15061595,  0.05219417,  0.01925796, ...,  0.03899952,
          0.12989182, -0.09922008]]),
 array([[ 5.88851094e+00,  6.39493279e+00,  6.42567098e+00, ...,
          3.41379594e-02, -2.30975312e-01, -8.48058249e-02],
        [ 2.86533588e+00,  3.94315871e+00,  4.26806029e+00, ...,
          3.24974038e-01,  4.92719662e-01,  3.03938357e-01],
        [ 6.51840322e+00,  7.28141064e+00,  6.04911365e+00, ...,
          2.98212450e-01,  2.55496864e-01,  1.58512641e-01],
        ...,
        [-1.64811466e-02, -6.35298091e-02,  3.43044391e-01, ...,
         -2.61988906e-01, -3.12971614e-01, -9.79802487e-01],
        [-4.75589184e-01,  2.10913647e-01, -3.56288105e-01, ...,
         -2.93152011e-01, -5.15714485e-02,  3.06025901e-03],
        [-7.22446819e-01,  3.09843834e-02,  6.75764894e-01, ...,
          7.15025102e-01,  8.74327120e-02, -3.71423276e-03]]))
```

```
In [ ]:
```