# Circular Trading Detection

P. Ram Anudeep - EE16BTECH11027
V. Revanth Babu - ES16BTECH11027
C.Jeevan Chandra - CS16BTECH11042

## Dataset :

Given dataset has 32,449 entries of seller-buyer transactions. It has 8573 sellers and 30,520 buyers. So it is clear that not many people are both buyers and sellers in the dataset. In a circular trading, a person needs to be both buyer and seller to be considered as a valid fraudulent candidate. So, we removed all the traders which are not both buyer and seller. It reduced our traders to almost 6000. And there are many transactions where the buyer and seller are the same person, so we did not consider these edges. Dataset is of the following form :

| S.No | Seller Id | Buyer Id | Amount |
|---|---|---|---|

## Generating Graphs :

We have generated two different graphs using **NetworkX** :

1. Undirected Graph
2. Directed Graph

In all these three graphs, we considered the nodes as the trader who is both buyer and seller.

1. **Undirected Graph** :
   Let's say if there are some edges from u->v and the sum of transactions be sum1. And if there are some edges from v->u and the sum of transactions be sum2. We have removed all the above mentioned edges and created an edge with edge weight (sum1+sum2).

2. **Directed Graph** :
   In directed graph, we have removed all the edges same as above. And we have kept two edges u->v with edge weight sum1 and v->u with edge weight sum2.

## Algorithm:

- We have used 2 popular clustering algorithms to find the clusters.
- First we have generated all the clusters using shared nearest neighbours algorithm.
- Next we have generated a directed graph containing only the edges of all the nodes found in the clusters generated previously.
- Then we have used mutual nearest neighbour algorithm to find the clusters again. This time we can say with high confidence that the resulting clusters have collusion sets.

## Shared nearest neighbor Algorithm :

**Parameters : ( k = 6, Θ = 2 )**

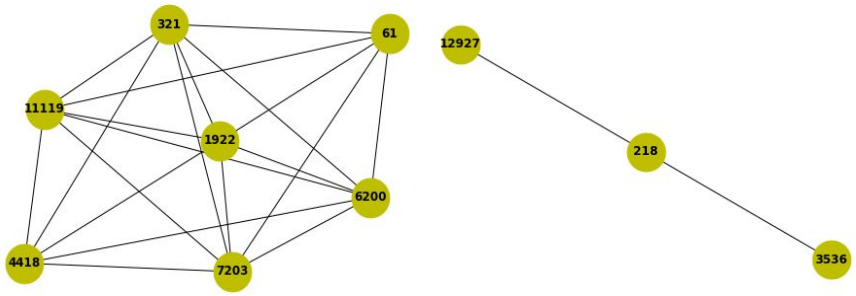
**Steps** : Parameters ( k, Θ)

1. Cluster Initializations
   Find KNN(v) for all the vertices in the graph.
   Remove every edge from v to any vertex not
   in kNN(v).
2. Cluster grouping
   Find all the vertices which have >=Θ neighbors in common &
   also neighbors of each other and combine them into a cluster.

Finally this algorithm classifies the dealers into clusters with high resemblance.

**Time Complexity** : O(k*N*N)
**Running Time :** 1minute 6seconds
**Number of Clusters generated :** 44 clusters

**Results :** There are many clusters without cycles in the above algorithm. So we have decided to run the Mutual Nearest Neighbor algorithm on top of it. Few of the clusters generated are shown in the graph below.



## Virtual Graph :

From the original directed graph we have considered only edges which has at least one of the nodes in the clusters formed in the Shared Nearest neighbour algorithm. This graph has almost 3000 nodes. We have considered this graph and ran the Mutual Nearest Neighbor Algorithm.

## Mutual Nearest Neighbour Algorithm :

**Parameters : ( k = 6, M = 30, MAX_MNV = 150 )**

**Steps** : Parameters (k, M, MAX_MNV)

k : k nearest neighbours
M : Number of desired clusters
MAX_MNV: Threshold value of MNV

1. Cluster Initializations
   Find KNN(v) for all the vertices in the graph.
   Remove every edge from v to any vertex not in kNN(v).
2. Cluster grouping
   Until N is reached :
   Find two clusters such that mutual neighbourhood value is minimum among all pairs of clusters. Combine them into one cluster and continue.

Repeat the above process for different values of k.

**Time Complexity** : O(k*N*N)
**Running Time :** 9minute 24seconds
**Number of Clusters generated :** 27 clusters

**Results :**

We have got a good number of clusters with size >=3. Glimpses are as below :