# CIS 509 – Mining Assignment 3 – Pseudo Code

**Input:** coPurchaseGraph, asin, amazonBooks

**Output:** Recommendations for asin

HighNumber is **max**(salesRank of all the nodes)/**min** (**min**(clustCoeff of all the nodes), **min**(weight of all the edges))

collectibles = get **Collectibles** products

similar = get **Similar** products

popular = get **Popular** products

Recommendation = merge the lists of collectibles, similar, and popular adhering to the Rank

**get\_cliques**: A function to get cliques in a Graph with a given node

**def** *Collectibles* ()

**inputs**: coPurchaseGraph, asin , amazonBooks, HighNumber

**output:** list of collectibles

cliques = **get\_cliques** (coPurchaseGraph, asin)

Initialize Rank[node *forall* nodes] = HighNumber

Calculate meanSalesRank[c *forall* cliques] = Sum of Sales Rank in the clique/size of Clique

Calculate Rank of each node by minimizing value as

Rank[node *forall* nodes in each clique] = **min**(meanSalesRank[c where node in c]/CliqueFrequency[node]) across each clique

maxClique= **Sort**(Rank)

**return** maxClique

**get\_ego\_network**: A function to get ego network of a node with a certain radius

**def** *Similarity* ():

**inputs**: coPurchaseGraph, asin , amazonBooks, HighNumber

**output:** similar items

islandGraph: A graph, initialized to null

egoNetwork: **get\_ego\_network**(coPurchaseGraph, asin, radius = 1)

threshold = median(weights of edges in egoNetwork)

create islandGraph with the threshold

Rank the nodes with the Sales Rank as follows

Rank[node] = salesRank/weight

simItems = **Sort**(Rank)

**return** simItems

**def** *Popularity*():

**inputs**: coPurchaseGraph, asin , amazonBooks, HighNumber

**output:** popular items

newGraph is computed by removing asin from coPurchaseGraph

calculate clustering coefficient of nodes in newGraph

Rank the nodes with the Sales Rank and clustCoeff as follows

Rank[node] = salesRank of the node/clustCoeff

popularitems = **Sort**(Rank)

**return** popularitems