Network Effect in Fixed Income Market

Hackathon Challenge:

Fixed Income Market (also known as the bond market) is where large corporations raise money to fund its operations. There are three main types of institutions in the bond market:

1. **Issuers** - Corporations that borrow money
2. **Investors** - Investors that lend money
3. **Dealers** - Brokers who act as a middleman for Issuers and Investors

The relationships amongst the three parties are complex. The objective of the problem is to understand the strength of the network in two distinct worlds:

World 1 - Simplified version of the current world, where Dealers broker all communications. This implies that if an Issuer needs to communicate with Investor(s) or vice-versa, all communication has to be routed via a Dealer. In other words, no direct line of communication exists between any Issuer or Investor

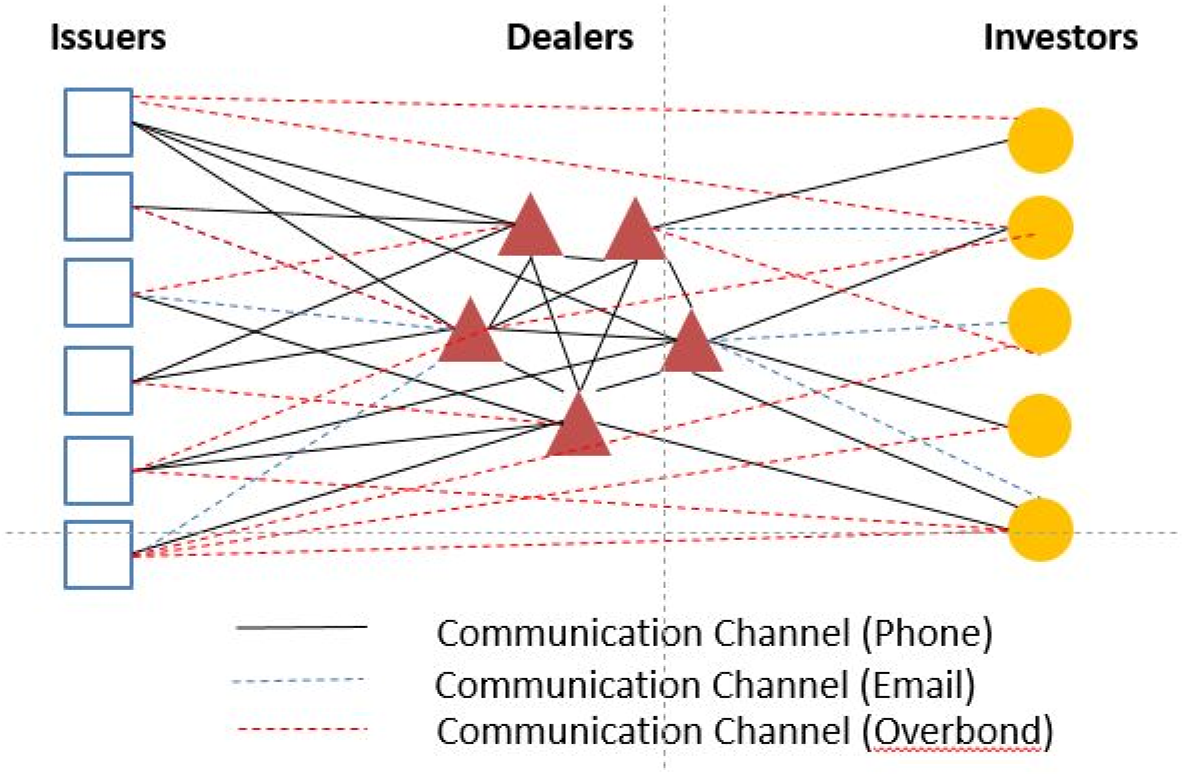
World 2A - Advanced world, where there exist efficient, digital channel of communication enabled by a platform such as Overbond. In this world, in addition to existing communication channels between Issuers/Investors -> Dealers, there are direct channels between Issuer and Investors as well. In this world, **~80%** Issuer/Investor have direct channels via Overbond.

World 2B - Same as World 2A, but in this World **~60%** Issuer/Investor have direct channels via Overbond.

You will find two files for each World. These files represent the state of a World as an undirected graph, where each node in the graph is an institution (Dealer, Investor or Issuer) and each edge between any 2 nodes is a channel of communication of type Phone / Email / Overbond

1. *nodes_world.txt* - Mapping of nodes to institution type
2. *edges_world.clq* - A graph representing the existing communication channels between all entities (in DIMACS format)

Example:


Issuers
Dealers
Investors
Communication Channel (Phone)
Communication Channel (Email)
Communication Channel (Overbond)

Problem Statement

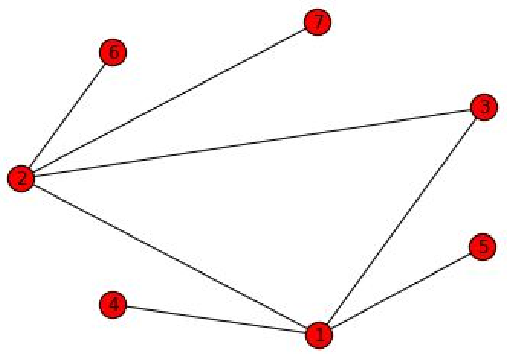
The task is, for each World, to find the largest subset of entities such that every entity in this set has at least one communication channel with every other entity in the set.

Sample Problem

Input:

1 Dealer 2 Dealer 3 Investor 4 Investor 5 Investor 6 Issuer 7 Issuer	e 1 2 Phone e 1 2 Email e 1 3 Email e 1 4 Phone e 1 5 Email e 2 6 Phone e 2 7 Email e 2 3 Phone
File 1: nodes_world.txt	File 2: edges_world.clq

Visualization:



Output: result.txt

```
1 Dealer/n
2 Dealer/n
3 Investor/n
```

Subset in result.txt should be sorted by node numbers (from low to high)

Please note the format of result.txt must match the above sample EXACTLY. (the new line character is shown explicitly for clarification purposes)

Presentations:

At the end of the hackathon, the winning teams will be required to briefly present their solutions. As part of this presentation, each team should present the total average communication cost for the subset that was calculated in result.txt for World 2A **and** World 2B

Costs of Communication Channels

In the real world, different communication channels incur different costs for the business

Cost Base = \$1.00

Phone - 100% Cost Base
Email - 80% Cost Base
Overbond - 50% Cose Base

Cost for Example Solution

```
Total_Sum_Cost   = Sum_Cost( Sum_Cost(1, 2) + Sum_Cost(1, 3) + Sum_Cost(2, 3) )
                 = Sum_Cost( Sum_Cost(Phone + Email) + Sum_Cost(Email) + Sum_Cost(Phone) )
                 = Sum_Cost( Sum_Cost($1.00 + $0.80) + Sum_Cost($0.80) + Sum_Cost($1.00) )
                 = Sum_Cost( $1.80 + $0.80 + $.100 )
                 = $3.60
```

Rules

Teams:

Teams of **maximum two** are permitted, but candidates can choose to work individually.

Programming Language:

You are allowed to submit your solution in either Python(2.x), C++ or Ruby(>=2.1)

Please note you are not allowed to use any libraries (besides the standard library).

For C++, you may not use std::list (if you need to use a linked list in your solution, you'll need to implement it yourself!)

Submission:

Please email your final submission to uoft-hackathon-submission@overbond.com in a .zip file with the following files included:

1. "**readme.txt**" with a brief overview of the design of your solution (We will read this file if your solution is competing for the top 5/10 spots)
2. For **1 person team** name .zip file: **FirstName_LastName_StudentNumber.zip**
3. For **2-person team** name .zip file: **FirstName1_LastName1_StudentNumber1_FirstName2_LastName2_StudetNumber2.zip**
4. Please submit your submissions by Sunday June 19, 12pm

Solution Files

Our testing framework will expect the following structure in the .zip file:

Python/Ruby:

```
$ python solution.py nodes_world.txt edges_world.clq
```

```
$ ruby solution.rb nodes_world.txt edges_world.clq
```

Once solution.py/solution.rb is finished executing there should be a file called result.txt in the same directory

C++:

```
$ make
```

This should create an executable called 'solution'. You are allowed to use the optimization flags (o1/o2/o3)

```
./solution nodes_world.txt edges_world.clq
```

Once solution is finished executing there should be a file called result.txt in the same directory

Please note, the .zip should not contain any nodes_world.txt/edges_world.clq files. Our testing framework will execute your solution with different instances of World 2 states

Criteria:

1. The submitted product must be functionally correct, adhering to all the conditions set out in the problem set.
2. Submissions will be evaluated based on the runtime of the program (e.g. how long does it take for your program to complete execution)

Note: We expect most contestants to create a functionally correct solution fairly early in the weekend. The real challenge is to achieve the most optimal solution in terms of runtime / space. We will be ranking the winners based on the fastest execution times.

Challenge: Our execution time to beat is 9.1736 seconds for World 2A