**Graph Analysis and Finding the Most Influential**

**Person in The Company**

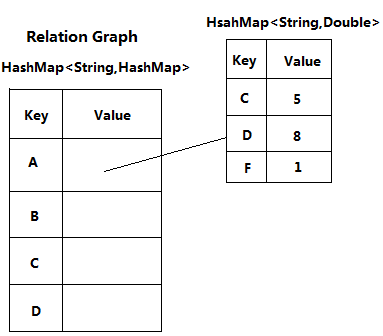
Qiaoyi He

Northeastern University

Author Note

This Report Was Prepared For Program Structure & Algorithms, Taught by Professor Kal Bugrara

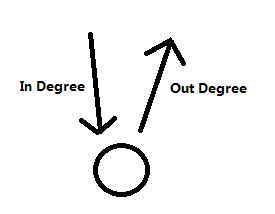
1. **Creating the Social Graph**

****

Description: After parsing Froms and Tos(and CCs), we store these information into a big HashMap named Relation Graph. This represents a directed and weighted graph. Each node is a person in company, and each line stands for who sent emails to whom, and the weight indicates the number of emails. The keys of this big HashMap are emails of people who sent emails to other people. The value of one key is another small HashMap. The keys of this small HashMap are emails of people who received email from this person. And the value of the small HashMap is the number of emails between this two people. In this relation graph, there are 5790 nodes and 21126 lines.

My final project is the website which is an online shopping store. It includes index page for users to start shopping, one home page for navigation, one products page show users all products and allows users to search a product category, one contact page for users sending message to customer service, one account page for login or register, one register page for create a new user.

1. **Calculating the Degree of Each Person**

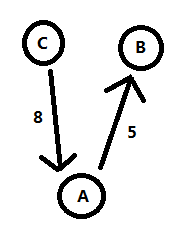


Description: For one person, we name the number of people who send emails to this person as In Degree Parameter. On the other hand, we define the number of the people who receive email from this person as the Out Degree Parameter. We add the in degree and the out degree together, and this result is the degree centrality of this person. In a practice words, this result is the all contacts of this person. Thus, in this report, we use contact to descript degree concept.

**For Each Person:**

**Degree Proportion=Degree/Nodes Number\*100%**

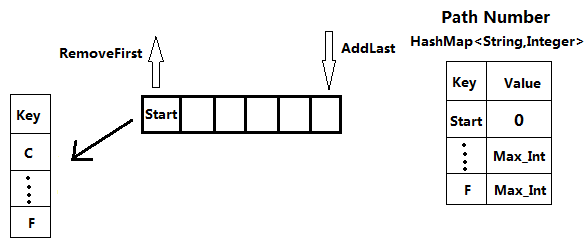
1. **Calculating Sent Emails and Received Emails of Each Person**



Description: When we analysis the Closeness Centrality and the betweenness Centrality of each person. We think the graph is a un-directed and un-weighted graph. However, we still need think about the number of emails between people, because email number is a crucial parameter when we analyze one person’s centrality. The total email number will reflect that whether this person is active at this company.

For one person, we add the number of all emails this person received and the number of all emails this person sent. The more the total number of emails, the more this person is active.

1. **Calculating the Closeness and the Farness of Each Person**
2. **Calculating the Geodesic( Shortest Path)**



Description: At the first, we create a new HashMap to store path number from this start node to all other nodes. We set the start node’s path number as 0, and other nodes’ path number as Max\_Integer. Then, we add start nodes into a new LinkedList. Until the LinkedList is not empty, we continually remove the first elements and get the small HashMap of this element from the big HashMap. Traversing this small HashMap, and change each node’s path number to the prior node’s value adding one once this node’s path number is Max\_Integer.

1. **The Closeness and the Farness Centrality**

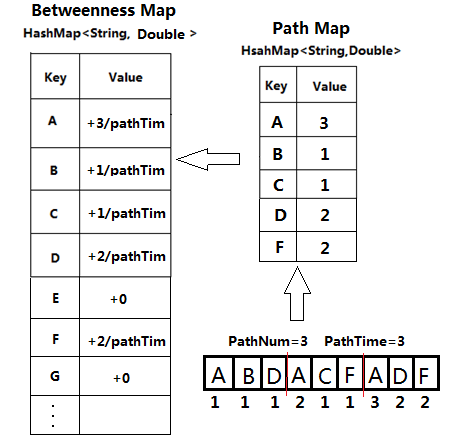
For each start node, we add the all value in the path number HashMap. This result is the Closeness Centrality of this node. And the reciprocal of Closeness is the Farness Centrality of this person.

**Closeness Centrality=**

**Farness Centrality= 1/ Closeness Centrality**

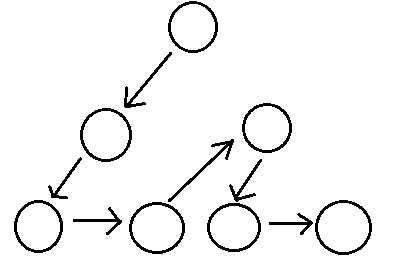
* Where n is the number of nodes in graph

1. **Calculating the Betweenness Centrality of Each Person**

****

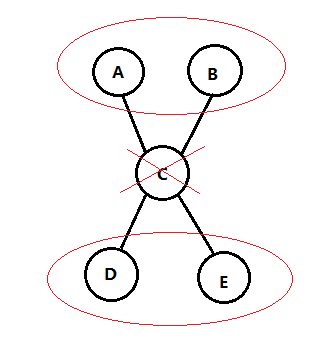
Description: When we calculate the betweenness centrality, we need find all shortest paths between all nodes. We use the DFS method to find all shortest paths. However, when we get the first element of stack, we do not pop that element until we check all children of this node. We use PathNum to record the minimum path number, and use a Path Map to store the nodes and times which shortest path passes. When we finish finding one shortest path between one pair of nodes, we put the betweenness of each node into Betweenness Map. Thus, Betweenness Map has the betweenness value of each node.

1. **Calculate the Transitivity of Each Node**

****

Description: Firstly, we get the start node. Then Using DFS method to traversal it’s all friends, and use recursion to traversal friends’ friends until there is no more nodes connect with them. We get the maximum subgraph which this node in. The number of nodes in this subgraph is the transitivity of this node. In this subgraph, this person can reach every node through other nodes.

1. **Finding the All CutPoints**

****

Description: If a node’s removal will increase the number of subgraph, this node is a cutpoint of this graph. We use Tarjan algorithm to find all cutpoints. Using DFS to traversal each node, and giving each node an order number. If its child cannot connect to its father, this node is a cutpoint. We use a HashMap to store this point and the number of subgraph when we remove this node.

1. **Finding the Most Powerful Person**

In the analysis part, there are five parameters to calculate the general centrality. Each part represents 20% in final result.

However, each part has different order of magnitudes. Thus, we need make these data have same order of magnitudes (1 to 10) by timing or dividing by 10^n.

**General Centrality= (Degree/100)\*20% +**

**(Sent Email/100)\*12% +**

**(Received Email/10)\*8%+**

**(Farness\*10)\*20%+**

**(Betweenness/100)\*20%+**

**((Transitivity-10000)/1000)\*20%**

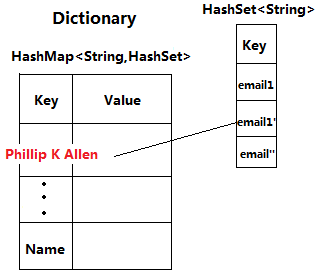
For all people, the person who has the most general centrality is the most powerful person at Enron.

1. **Solving Multiple Email Problem**
2. **Transforming Names**

****

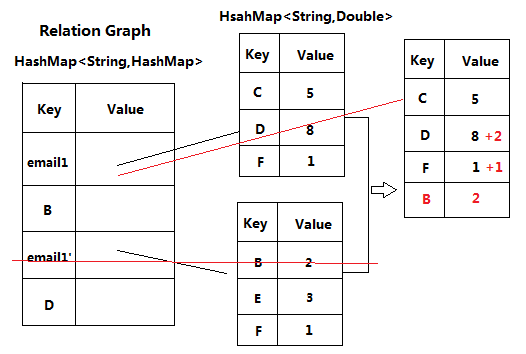
Description: The reason why one person has multiple emails is that they have multiple forms of names as shown in the picture above. We get all the names, and exchange before and after once there is a coma in the name. After that, we remove the period and space. //At last, we transform all letters to lowercase

1. **Setting Up the Dictionary**

****

**Description: We store transformed names in a new Dictionary HashMap as its keys. And the HashSet of each key is use for storing all of emails of this person.**

1. **Combining Data**

****

**Description: After building the whole Relation Graph, we use Dictionary HashMap to combine multiple emails’ data. When all emails data are combined to one HashMap for one email, we remove other emails’ information in Relation Graph.**