We know that

$$w(\mathbf{r}') = \sum_{i=1}^{n} r'_i = \sum_{i=1}^{n} \sum_{j=1}^{n} M_{ij} r_j.$$

Then we can interchange the order of summation, we get

$$w(\mathbf{r}') = \sum_{i=1}^{n} \sum_{j=1}^{n} M_{ij} r_j = \sum_{j=1}^{n} \sum_{i=1}^{n} M_{ij} r_j.$$

Since there is no dead ends in the network, $\sum_{i=1}^{n} M_{ij} = 1$ for each j. Therefore,

$$w(\mathbf{r}') = \sum_{j=1}^{n} \sum_{i=1}^{n} M_{ij} r_j = \sum_{j=1}^{n} r_j = w(\mathbf{r}).$$

By the definition, we have

$$\mathbf{r}'_i = \beta(\sum_{j=1}^{n} M_{ij} r_j) + \frac{1-\beta}{n}.$$

Then we can calculate $w(\mathbf{r}')$ as:

$$\begin{aligned} w(\mathbf{r}') &= \sum_{i=1}^{n} \left(\beta(\sum_{j=1}^{n} M_{ij} r_j) + \frac{1-\beta}{n} \right) \\ &= \beta \sum_{i=1}^{n} \sum_{j=1}^{n} M_{ij} r_j + (1-\beta) \\ &= \beta w(\mathbf{r}) + (1-\beta) \end{aligned}$$

If $w(\mathbf{r}) = 1$, we have $w(\mathbf{r}') = \beta \times 1 + (1-\beta) = \beta + (1-\beta) = 1 = w(\mathbf{r})$. If $w(\mathbf{r}) = w(\mathbf{r}') = x$, we have $x = \beta x + (1-\beta)$, which implies that $x = 1$, so $w(\mathbf{r}) = w(\mathbf{r}') = 1$.

In conclusion, $w(\mathbf{r}') = w(\mathbf{r})$ if and only if $w(\mathbf{r}) = 1$.

The equation for r'_i is:

$$r'_i = \sum_{j \notin D} (\beta M_{ij} + \frac{1-\beta}{n}) r_j + \sum_{j \in D} \frac{1}{n} r_j.$$

Then we calculate $w(\mathbf{r}')$ as follow:

$$\begin{aligned}
w(\mathbf{r}') &= \sum_{i=1}^{n} r'_i \\
&= \sum_{i=1}^{n} \left(\sum_{j \notin D} (\beta M_{ij} + \frac{1-\beta}{n}) r_j + \sum_{j \in D} \frac{1}{n} r_j \right) \\
&= \beta \sum_{j \notin D} \left(\sum_{i=1}^{n} M_{ij} \right) r_j + \frac{1-\beta}{n} \sum_{j \notin D} \sum_{i=1}^{n} r_j + \frac{1}{n} \sum_{i=1}^{n} \sum_{j \in D} r_j \\
&= \beta \sum_{j \notin D} 1 \times r_j + (1-\beta) \sum_{j \notin D} r_j + \sum_{j \in D} r_j \\
&= \beta \sum_{j \notin D} r_j - \beta \sum_{j \notin D} r_j + \sum_{j \notin D} r_j + \sum_{j \in D} r_j \\
&= \sum_{j=1}^{n} r_j \\
&= w(\mathbf{r}) \\
&= 1
\end{aligned}$$

In conclusion, we prove that $w(\mathbf{r}')$ is also 1.

The 5 node ids with the highest PageRank scores:

Node id	PageRank score
263	0.002020291181518219
537	0.00194334157145315
965	0.0019254478071662631
243	0.001852634016241731
285	0.0018273721700645144

The 5 node ids with the least PageRank scores:

Node id	PageRank score
558	0.0003286018525215297
93	0.0003513568937516577
62	0.00035314810510596274
424	0.00035481538649301454
408	0.00038779848719291705

The 5 node ids with the highest hubbiness scores:

Node id	hubbiness score
840	1.0
155	0.9499618624906543
234	0.8986645288972264
389	0.863417110184379
472	0.8632841092495217

The 5 node ids with the least hubbiness scores:

Node id	hubbiness score
23	0.042066854890936534
835	0.05779059354433016
141	0.06453117646225179
539	0.06602659373418492
889	0.07678413939216454

The 5 node ids with the highest authority scores:

Node id	authority score
893	1.0
16	0.9635572849634398
799	0.9510158161074016
146	0.9246703586198444
473	0.899866197360405

The 5 node ids with the least authority scores:

Node id	authority score
19	0.05608316377607618
135	0.06653910487622794
462	0.07544228624641902
24	0.08171239406816946
910	0.08571673456144878

We have C_i be the set of nodes of G that are divisible by a positive integer i, so we can denote C_i as:

$$C_i = \{j | j = k \times i, \quad j \in G, \quad k \in N\}$$

If $|C_i| = 1$, which means that there is only 1 node in C_i and C_i is a clique. If $|C_i| \geq 2$, for every two nodes u and v have a common factor i. Hence, there is an edge between nodes u and v, which means that C_i is a clique. In conclusion, we prove that C_i is a clique for any i.

C_i is a maximal clique if and only if i is a prime and i less than or equal to 1000000.

If $i > 1000000$, which means that there does not exist a node in G be divisible by i. Thus, C_i is not a clique.

If i is less or equal to 1000000, but not a prime. This means that there is factor j of i and $1 < j < i$. By the definition of C_i, node j is not in C_i. However, node j has an edge to every node in C_i since it has a common factor j with every node in C_i. Thus, C_i is not a maximal clique.

If i is less or equal to 1000000 and i is a prime. Suppose C_i is not a maximal clique. This means that there is a node j can be added into C_i. However, the only common factor between node i and node j is i since i is a prime, it implies that node j has already in C_i. Therefore, C_i is a maximal clique.

Conversely, suppose C_i is a maximal clique. If $i \geq 1000000$, then C_i can't be a clique since it is empty. If i is less than or equal to 1000000 but not a prime. It implies that there is a factor j of i and node j has a edge to every node in C_i, which means that C_i is not maximal. Thus, i has to be a prime and be less than or equal to 1000000.

We know that $C_2 = \{2k | k = 1 \dots 500000\}$. Thus, there are 500000 elements in C_2. We have to prove C_2 is the unique largest clique.

1. The largest: Suppose there is a clique C with N elements, where $N > 500000$. First of all, we can divide G into 500000 classes as $\{2\}, \{3,4\}, \dots, \{999999, 1000000\}$. By pigeonhole principle, given N elements where $N > 500000$, at least two of them will be in the same class, $\{2k-1, 2k\}$. However, $2k-1$ and $2k$ are relatively prime, which implies that these two nodes have no edge between them, so C is not a clique. Thus, C_2 with 500000 elements is the largest clique.

2. Uniqueness: Suppose there is another clique C with 500000 elements. Again, we can divide G into 500000 classes as $\{2\}, \{3,4\}, \dots, \{999999, 1000000\}$. Since C is a clique, there are no any two nodes relatively prime. Thus, the only way to pick the elements in G is to choose all the even numbers in G. Therefore, $C = C_2$, which means that C_2 is the unique clique.

Therefore, we prove that C_2 is the unique largest clique.

(i) We have

$$|E[S]| = \frac{1}{2}\sum_{v\in S}\deg_S(v) \geq \frac{1}{2}\sum_{v\in\{S\setminus A(S)\}}\deg_S(v) \geq \frac{1}{2}|S\setminus A(S)| \times 2(1+\epsilon)\times\rho(S)$$

By the definition of $\rho(S)$, we have

$$|E[S]| \geq \frac{1}{2}\times 2(1+\epsilon)\times\frac{|E[S]|}{|S|}\times|S\setminus A(S)| \iff \frac{1}{1+\epsilon}|S| \geq |S\setminus A(S)| = |S| - |A(S)|$$

Reformulate the inequality above, we get

$$\frac{1}{1+\epsilon}|S| \geq |S| - |A(S)| \iff |A(S)| \geq \frac{\epsilon}{1+\epsilon}|S|$$

(ii) We have $|A(S)| \geq \frac{\epsilon}{1+\epsilon}|S|$, which implies that $|S| - |A(S)| \leq \frac{1}{1+\epsilon}|S|$. Let's define $S_0 = S$, $|S_0| = n$ and S_i as remaining set of S_0 after i iterations, then we have $|S_i| = |S_{i-1}| - |A(S_{i-1})|$. Suppose the program stops at iteration k, then we have $1 \leq |S_{k-1}| < 1+\epsilon$. Hence, with the inequality above, we have

$$\begin{aligned}
|S_1| &\leq \frac{1}{1+\epsilon}|S_0| \\
|S_2| &\leq \frac{1}{1+\epsilon}|S_1| \\
&\vdots \\
1 \leq |S_{k-1}| &\leq \frac{1}{1+\epsilon}|S_{k-2}|
\end{aligned}$$

Combine all the inequalities, we get $1 \leq (\frac{1}{1+\epsilon})^{k-1}|S_0|$, which implies that $(1+\epsilon)^{k-1} \leq |S_0|$. We can take log on both side, then we have

$$k - 1 \leq \log_{1+\epsilon}|S_0| = \log_{1+\epsilon}(n)$$

Thus, $k \leq \log_{1+\epsilon}(n) + 1 = O(\log_{1+\epsilon}(n))$.

(i) Suppose there is a node $v \in S^*$, $\deg_{S^*}(v) < \rho^*(G)$, and define $S^{*'}$ as $S^{*'} = S^* \setminus v$. Then we can calculate the density of the set $S^{*'}$ as:

$$\rho(S^{*'}) = \frac{|E[S^*]| - \deg_{S^*}(v)}{|S^*| - 1} > \frac{|E[S^*]| - \rho^*(G)}{|S^*| - 1} = \frac{|E[S^*]| - \frac{|E[S^*]|}{|S^*|}}{|S^*| - 1} = \frac{|E[S^*]|}{|S^*|} = \rho(S^*)$$

This implies that the density of $S^{*'}$ is greater than S^*, which contradicts to the definition of S^*. Thus, for any $v \in S^*$, we have $\deg_{S^*}(v) \geq \rho^*(G)$.

(ii) At the start of the algorithm, all nodes in S^* are also in S. Suppose there is a node $v \in S^* \cap A(S)$ before the first iteration, there is never a node in $A(S)$, which means that we never remove any of nodes from S. Thus, S^* is a subgraph of S and $\deg_{S^*}(v) \leq \deg_S(v)$.

In the algorithm, we have $\deg_S(v) \leq 2(1+\epsilon)\rho(S)$. In 4(b)(i), we have $\deg_{S^*}(v) \geq \rho^*(G)$. Combine this two inequalities, we get $2(1+\epsilon)\rho(S) \geq \rho^*(G)$.

(iii) In (ii), we have $\rho(S) \geq \frac{1}{2(1+\epsilon)\rho^*(G)}$. And in the algorithm, there is a condition that "if $\rho(S) > \rho(\tilde{S})$, then $\tilde{S} \leftarrow S$". Thus, $\rho(\tilde{S})$ is greater than or equal to $\rho(S)$ in the end. Combining $\rho(\tilde{S}) \geq \rho(S)$ and $\rho(S) \geq \frac{1}{2(1+\epsilon)}\rho^*(G)$, we prove that $\rho(\tilde{S}) \geq \frac{1}{2(1+\epsilon)}\rho^*(G)$.

Information sheet
CS246: Mining Massive Data Sets

Assignment Submission Fill in and include this information sheet with each of your assignments. This page should be the last page of your submission. Assignments are due at 11:59pm and are always due on a Thursday. All students (SCPD and non-SCPD) must submit their homework via Gradescope (http://www.gradescope.com). Students can typeset or scan their homework. Make sure that you answer each (sub-)question on a separate page. That is, one answer per page regardless of the answer length. Students also need to upload their code on Gradescope. Put all the code for a single question into a single file and upload it.

Late Homework Policy Each student will have a total of *two* late periods. *Homework are due on Thursdays at 11:59pm PT and one late period expires on the following Monday at 11:59pm PT*. Only one late period may be used for an assignment. Any homework received after 11:59pm PT on the Monday following the homework due date will receive no credit. Once these late periods are exhausted, any assignments turned in late will receive no credit.

Honor Code We strongly encourage students to form study groups. Students may discuss and work on homework problems in groups. However, each student must write down their solutions independently, i.e., each student must understand the solution well enough in order to reconstruct it by him/herself. Students should clearly mention the names of all the other students who were part of their discussion group. Using code or solutions obtained from the web (GitHub/Google/previous year's solutions etc.) is considered an honor code violation. We check all the submissions for plagiarism. We take the honor code very seriously and expect students to do the same.

Your name: Yen-Yu Chang

Email: yenyu@stanford.edu **SUID:** 006350488

Discussion Group: Cheng-Min Chiang, Fang-I Hsiao, Alvin Hou

I acknowledge and accept the Honor Code.

(Signed) Y.Y.Chang