Suppose the Web has no dead ends. We prove that $w(\mathbf{r}^{'}) = w(\mathbf{r})$.

We need to show that $\sum_{i=1}^{n} \sum_{j=1}^{n} M_{ij} r_j = \sum_{i=i}^{n} r_i$.
We know there are no dead ends, so the condition

$$\sum_{i=1}^{n} M_{ij} = 1$$

holds for each j, following

$$\sum_{i=1}^{n} \sum_{j=1}^{n} M_{ij} r_j = \sum_{j=1}^{n} \left(\sum_{i=1}^{n} M_{ij} \right) r_j = \sum_{i=j}^{n} r_j$$

and the equation $w(\mathbf{r}^{'}) = w(\mathbf{r})$ holds.

Suppose the Web has no dead ends. We use a teleportation probability of $1-\beta$ $(0<\beta<1)$ where we teleport to a random node. The expression for the next estimate of r_i becomes $r_i^{'} = \beta\left(\sum_{j=1}^{n} M_{ij}r_j\right) + \frac{1-\beta}{n}$. We observe under what circumstances will $w(\mathbf{r}^{'}) = w(\mathbf{r})$ hold.

Let us observe $w(r^{'})$.
Using the condition from task 1(a), it follows

$$\begin{aligned} w(r^{'}) &= \beta\sum_{j=1}^{n} r_j + 1 - \beta \\ &= \beta w(r) + 1 - \beta \end{aligned}$$

Now we assume that $w(r^{'}) = w(r)$ is true and we denote it as $w(r^{'}) = w(r) = W$, where W must satisfy the equation

$$W = \beta W + 1 - \beta$$

We thus have

$$\begin{aligned} W(1-\beta) &= 1-\beta \\ \Rightarrow W &= \frac{1-\beta}{1-\beta} = 1 \end{aligned}$$

To conclude, $w(\mathbf{r}^{'}) = w(\mathbf{r})$ will hold iff. they are both equal to 1.

Now let us assume there are one or more dead ends. Call a node 'dead' if it is a dead end and 'live' if not. At each iteration, we teleport from live nodes with probability $1-\beta$ and teleport from dead nodes with probability 1. In both cases, we choose a random node uniformly to teleport to. Assume $w(r) = 1$.
We write the equation for $r_i^{'}$ in terms of β, M, r, n and D (where D is the set of dead nodes). We prove that $w(r^{'})$ is also 1.

Note that if M contains dead ends, then $\sum_{j=1}^{n} r_j < 1$, so we have to 'correct' r from previous formulations so that is sums to 1. We do that by explicitly follow random teleport links from dead ends with probability 1.
We thus have

$$r_i^{'} = \beta \sum_{j=1}^{n} M_{ij} r_j + \frac{1-\beta}{n} + \frac{\beta}{n} \sum_{j \in D} r_j$$

and (considering all previous conditions)

$$w(r^{'}) = \beta \sum_{i=1}^{n} \sum_{j=1}^{n} M_{ij} r_j + \sum_{i=1}^{n} \frac{1-\beta}{n} + \sum_{i=1}^{n} \frac{\beta}{n} \sum_{j \in D} r_j$$

$$w(r^{'}) = \beta \sum_{j \notin D} r_j + 1 - \beta + \beta \sum_{j \in D} r_j$$

$$w(r^{'}) = \beta \left(\sum_{j \notin D} r_j + \sum_{j \in D} r_j \right) + 1 - \beta$$

Summation of $\sum_{j \notin D} r_j$ and $\sum_{j \in D} r_j$ gives us $w(r)$ and $w(r)$ is assumed to be 1. Thus follows

$$w(r^{'}) = \beta + 1 - \beta = 1$$

and the condition is proved.

The code is available in the attached file `HW3_q2.html`.

Top 5 node ids with the highest PageRank scores:

- 263
- 537
- 965
- 243
- 285

Bottom 5 node ids with the lowest PageRank scores:

- 558
- 93
- 62
- 424
- 408

The code is available in the attached file `HW3_q2.html`.

Top 5 node ids with the highest hubbiness scores:

- 893
- 16
- 799
- 146
- 473

Bottom 5 node ids with the lowest hubbiness scores:

- 19
- 135
- 462
- 24
- 910

Top 5 node ids with the highest authority scores:

- 840
- 155
- 234
- 389
- 472

Bottom 5 node ids with the lowest authority scores:

- 23
- 835
- 141
- 539
- 889

Let C_i be the set of nodes of G that are divisible by i, where i is a positive integer greater than 1.

By definition, there is an edge between nodes i and j iff. i and j have a common factor other than 1. In C_i, any pair of two nodes have i as the common factor, hence there exists an edge between them. Since all nodes are connected with an edge, C_i really is a clique for any $i > 1$.
Note that if there is less than two nodes in a graph C_i, it is a clique by definition.

Note that C_1 ($i = 1$) contains all nodes $2, 3, \ldots, 1000000$, since all those numbers are divisible by 1. Clearly, C_1 is not a clique, since every pair of prime numbers does not have a common factor other than 1. Therefore, between every two prime numbers there is no edge.

We observe the circumstances for C_i to be a maximal clique.

A maximal clique is a clique for which it is impossible to add a node and still retain the property of being a clique; i. e., a clique C is maximal if every node not in C is missing an edge to at least one member of C.
We are observing three different cases.

- Firstly, let us assume that i is greater than 1000000, so C_i is an empty clique, since there is no number less or equal to 1000000 that is divisible by something greater than 1000000.
  If we add any node to the C_i, we would get a clique with one node, hence C_i is not maximal.
- Secondly, let us assume that i is less or equal to 1000000 and not a prime number. By 3(a) we know that C_i is a clique. Since i is not prime, there exists at least one prime factor $k \neq 1$, such that k divides i. If all nodes in C_i are divisible by i, they are also divisible by k, since $i = k \cdot n$ for some n. Therefore, node k is connected to every node in C_i, following C_i is not a maximal clique.
- Lastly, let us assume that i is less or equal to 1000000 and a prime number. By 3(a) we know that C_i is a clique. We then know that there does not exist a node outside C_i, let us denote that node as j, that has an edge to the node c in C_i. To see that, let us suppose that node j is connected to c with an edge. Then c and j have a common factor other than 1, which, obviously, must be i, since i is prime. Therefore, node j would be already in the C_i. With that condition, C_i is a maximal clique.

To conclude, C_i is maximal clique iff. i is a prime number less or equal to 1000000, which is both necessary and sufficient condition (first two cases above are necessary conditions and the last one is sufficient).

We now prove that C_2 is the unique largest clique. That is, it has more elements than any other clique.

In C_2 there are all even numbers from 2 to 1000000, that is, it has exactly 500000 nodes.

If we obeserve pairs $(2,3),(4,5),(6,7),(8,9)$ and so on up to 1000000, we know that only one of the member of the pair can be in the clique, since i never divides $i+1$ (unsless it is 1, which is not the case here), thus the largest possible clique has 500000 nodes, exactly the same as C_2.
C_2 really is the largest clique.

Furthermore, clique C_2 is the only clique with 500000 nodes. If we take a look at pairs of form $(i, i+1)$ again and consider previous conclusion, we know that the largest clique must contain either 2 or 3. If it contains 3, it contains all numbers that have 3 as a factor, which is clearly less than 500000. So clique with 500000 nodes must contain 2, therefore all of the number that have 2 as factor.
That clique is clearly C_2, following that C_2 has more elements than any other clique.

C_2 is really the unique and the largest clique.

We show through the following steps that the algorithm terminates in a logarithmic nu,ber of steps.

(i) First we prove that at any interation of the algorithm, $|A(S)| \geq \frac{\epsilon}{1+\epsilon}|S|$ is true. Using the hand-shaking lemma, we have

$$2|E[S]| = \sum_{v \in S} \deg_S(v) \geq \sum_{v \in S \backslash A(S)} \deg_S(v) \tag{1}$$

By definition each node in $S \backslash A(S)$ has degree at least $2(1+\epsilon)\rho(S)$ and there are $|S| - |A(S)|$ nodes in $S \backslash A(S)$, since $A(S) \subseteq S$, so

$$\sum_{v \in S \backslash A(S)} \deg_S(v) \;\geq\; 2(1+\epsilon)\rho(S)\left(|S| - |A(S)|\right) = 2(1+\epsilon)\frac{|E[S]|}{|S|}\left(|S| - |A(S)|\right) \tag{2}$$

From (1) and (2) we have

$$2|E[S]| \geq 2(1+\epsilon)\frac{|E[S]|}{|S|}\left(|S| - |A(S)|\right)$$

following

$$\begin{aligned} |S| &\geq (|S| - |A(S)|)(1+\epsilon) \\ |A(S)| \cdot (1+\epsilon) &\geq \epsilon |S| \end{aligned}$$

so the condition

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

obviously holds.

(ii) Now we prove that the algorithm terminates in $\mathcal{O}(\log_{1+\epsilon}(n))$ iterations, where n is the initial number of nodes.
At each iteration, the size of $|S|$ decreases for $\frac{\epsilon}{1+\epsilon}|S|$ or more nodes, since (i) holds. The new size is therefore at most $|S| - \frac{\epsilon}{1+\epsilon}|S| = \frac{1}{1+\epsilon}\left(|S| + \epsilon|S| - \epsilon|S|\right) = \frac{1}{1+\epsilon}|S|$. In other words, at each iteration, the size of S decreases by a factor greater or equal than $1+\epsilon$. Since $|S| = n$ initially, the algorithm will obviously make $\mathcal{O}(log_{1+\epsilon}(n))$ steps before $S = \emptyset$.

We show through the following steps that the density of the set returned by the algorithm is at most a factor $2(1+\epsilon)$ smaller than $\rho^*(G)$.

(i) Assume S^* is the densest subgraph of G. Prove that for any $v \in S^*$, we have $\deg_{S^*}(v) \geq \rho^*(G)$.
We show that with contradiction. Assume the condition $\deg_{S^*}(v) < \rho^*(G)$ for some $v \in S^*$. Then

$$|E[S^*\backslash\{v\}]| = |E[S^*]| - \deg_{S^*}(v)$$

and

$$|S^*\backslash\{v\}| = |S^*| - 1$$

thus the density of $S^*\backslash\{v\}$ is

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

We now compare the $\rho(S^*)$ with $\rho(S^*\backslash\{v\})$. It follows that $\rho(S^*\backslash\{v\}) > \rho(S^*)$, since

$$\begin{aligned}
\frac{|E[S^*]| - \deg_{S^*}(v)}{|S^*| - 1} &> \frac{|E[S^*]|}{|S^*|} \\
|S^*| \cdot (|E[S^*]| - \deg_{S^*}(v)) &> |E[S^*]| \cdot (|S^*| - 1) \\
|S^*| \cdot |E[S^*]| - |S^*| \cdot \deg_{S^*}(v) &> |E[S^*]| \cdot |S^*| - |E[S^*]| \\
|E[S^*]| &> |S^*| \cdot \deg_{S^*}(v) \\
\frac{|E[S^*]|}{|S^*|} &> \deg_{S^*}(v)
\end{aligned}$$

holds and we come to contradiction (because S^* is the densest subgraph of G). Therefore, for any $v \in S^*$, the condition $\deg_{S^*}(v) \geq \rho^*(G)$ holds.

(ii) Consider the first iteration of the while loop in which there exists a node $v \in S^* \cap A(S)$. We prove that $2(1+\epsilon)\rho(S) \geq \rho^*(G)$.
Let $v \in S^* \cap A(S)$. From (i) it follows $\deg_{S^*}(v) \geq \rho^*(G)$ and we know that $\deg_S(v) \leq 2(1+\epsilon)\rho(S)$. We also know $\deg_{S^*}(v) \leq \deg_S(v)$, since $S^* \subseteq S$, so

$$\begin{aligned}
&2(1+\epsilon)\rho(S) \geq \deg_S(v) \geq \deg_{S^*}(v) \geq \rho^*(G) \\
\Rightarrow\ &2(1+\epsilon)\rho(S) \geq \rho^*(G)
\end{aligned}$$

(iii) Now we conclude that $\rho(\tilde{S}) \geq \frac{1}{2(1+\epsilon)}\rho^*(G)$.
Since the algorithm finishes in $\mathcal{O}(\log_{1+\epsilon}(n))$ iterations (from 4(a)), the subset S eventually becomes empty, so there exists an iteration where we start to remove the nodes from S^*. In that iteration the condition

$$\begin{aligned}
2(1+\epsilon)\rho(S) &\geq \rho^*(G) \\
\rho(S) &\geq \frac{\rho^*(G)}{2(1+\epsilon)}
\end{aligned}$$

holds. Note that $\rho(\tilde{S}) \geq \rho(S)$ (following from the algorithm), so

$$\rho(\tilde{S}) \geq \frac{1}{2(1+\epsilon)}\rho^*(G)$$

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: Sara Bizjak

Email: sarabizjak97@gmail.com **SUID:** 27202020

Discussion Group: Petra Podlogar

I acknowledge and accept the Honor Code.

(Signed)