Mandatory Assignment 4

Samuel Ørsnæs

November 13, 2018

Exercise 1

a. Compute eigenvalues and eigenvectors for the following matrix:

$$B = \begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix}$$

b. Diagonalize the matrix using an orthogonal matrix, i.e. find orthogonal Q, diagonal Λ such that $B = Q\Lambda Q^T$.

c. Write $\begin{bmatrix} 2 \\ 0 \end{bmatrix}$ as a linear combination of eigenvectors and use it to compute a closed formula for

$$B^n \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

Solution:

a. First, we find the eigenvalues of B by using the formula $\det(\lambda I - A) = 0$:

$$\begin{aligned} |B| &= \begin{vmatrix} \lambda - 2 & -3 \\ -3 & \lambda - 2 \end{vmatrix} = 0 \\ &= (\lambda - 2)^2 - 9 = 0 \\ &= (\lambda^2 - 4\lambda + 4) - 9 = 0 \\ &= \lambda^2 - 4\lambda - 5 = 0 \\ &\boldsymbol{= (\lambda + 1)(\lambda - 5) = 0} \end{aligned}$$

Having found a representation of the characteristic polynomial of B as shown above in bold, **we can conclude from its roots that the eigenvalues of B are -1 and 5.** Next, we use these values to calculate B's eigenvectors:

(a) $\boldsymbol{\lambda_1 = -1}$

Using the formula $(\lambda_1 I - A)\mathbf{x_1} = \mathbf{0}$, by row reducing the coefficient matrix, we can find the eigenvectors corresponding to $\lambda_1 = -1$:

$$\begin{aligned}\begin{bmatrix} -1-2 & -3 \\ -3 & -1-2 \end{bmatrix} &= \begin{bmatrix} -3 & -3 \\ -3 & -3 \end{bmatrix} \\ &\Rightarrow \begin{bmatrix} -3 & -3 \\ 0 & 0 \end{bmatrix} \; R_2 - R_1 \to R_2 \\ &\Rightarrow \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \; -\frac{1}{3}R_1 \to R_1\end{aligned}$$

Solving the formula, we find that $a_1 + a_2 = 0$. Choosing $a_2 = s$ for $s \in \mathbb{R} \wedge s \neq 0$, we see that the eigenvectors for $\lambda_1 = -1$ are of the form:

$$\mathbf{x_1} = \boldsymbol{s}\begin{bmatrix} \mathbf{-1} \\ \mathbf{1} \end{bmatrix}$$

(b) $\boldsymbol{\lambda_2 = 5}$

Applying the same logic above for $\lambda_2 = 5$:

$$\begin{aligned}\begin{bmatrix} 5-2 & -3 \\ -3 & 5-2 \end{bmatrix} &= \begin{bmatrix} 3 & -3 \\ -3 & 3 \end{bmatrix} \\ &\Rightarrow \begin{bmatrix} 3 & -3 \\ 0 & 0 \end{bmatrix} \; R_2 + R_1 \to R_2 \\ &\Rightarrow \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix} \; \frac{1}{3}R_1 \to R_1\end{aligned}$$

Here, we find that $a_1 - a_2 = 0$. Choosing $a_2 = s$ for $s \in \mathbb{R} \wedge s \neq 0$, we see that the eigenvectors for $\lambda_2 = 5$ are of the form:

$$\mathbf{x_2} = \boldsymbol{s}\begin{bmatrix} \mathbf{1} \\ \mathbf{1} \end{bmatrix}$$

b. To find an orthogonal matrix Q, we will use the eigenvectors found in the last exercise as our guide. Specifically, normalizing the eigenvectors and using these values as column vectors will yield matrix Q. There are no extra steps in this case as both eigenvalues have a multiplicity of 1. Thus:

$$\begin{aligned}\mathbf{q_1} &= \frac{(-1,1)}{||(-1,1)||} = \frac{(-1,1)}{\sqrt{(-1)^2 + 1^2}} = \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right) \\ \mathbf{q_2} &= \frac{(1,1)}{||(1,1)||} = \frac{(1,1)}{\sqrt{1^2 + 1^2}} = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)\end{aligned}$$

$$\boldsymbol{Q} = \begin{bmatrix} -\frac{\mathbf{1}}{\sqrt{\mathbf{2}}} & \frac{\mathbf{1}}{\sqrt{\mathbf{2}}} \\ \frac{\mathbf{1}}{\sqrt{\mathbf{2}}} & \frac{\mathbf{1}}{\sqrt{\mathbf{2}}} \end{bmatrix}$$

To find diagonal matrix Λ, we apply the formula $\Lambda = Q^T BQ$. One could just as well use Q^{-1} instead of Q^T, as they are equivalent here since Q is an orthogonal matrix. However, Q^T is immediately apparent from Q in this instance, so I notate it as such in the formula I use.

$$\begin{aligned}
\Lambda &= \frac{1}{2}\begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} -\frac{2\sqrt{2}}{2} & \frac{2\sqrt{2}}{2} \\ \frac{2\sqrt{2}}{2} & \frac{2\sqrt{2}}{2} \end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} -\frac{4\sqrt{2}}{2} + \frac{6\sqrt{2}}{2} & \frac{4\sqrt{2}}{2} + \frac{6\sqrt{2}}{2} \\ -\frac{6\sqrt{2}}{2} + \frac{4\sqrt{2}}{2} & \frac{6\sqrt{2}}{2} + \frac{4\sqrt{2}}{2} \end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{2} & 5\sqrt{2} \\ -\sqrt{2} & 5\sqrt{2} \end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix} -1-1 & -5+5 \\ 1-1 & 5+5 \end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix} -2 & 0 \\ 0 & 10 \end{bmatrix} \\
\boldsymbol{\Lambda} &= \begin{bmatrix} \mathbf{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{5} \end{bmatrix}
\end{aligned}$$

For my calculations, notice that I rationalized the denominator for Q above and scaled it up by a factor of 2, which is why I multiplied the entire expression by $\frac{1}{2}$ to balance it out. This was to make the calculations a bit easier to perform by hand. In any case, we have found our Λ and can check that it is correct by applying the formula $B = Q\Lambda Q^T$ to see if we get B back:

$$\begin{aligned}
B &= \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & 5 \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \\
B &= \begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix}
\end{aligned}$$

We do indeed, and so we can conclude that Q and Λ are respective orthogonal and diagonal matrices for B.

c. To represent (2, 0) as a linear combination of the eigenvectors of B, we can use the formula $A\mathbf{c} = (2, 0)$, where A is the matrix composed of the eigenvectors of B (in column vector form), and $\mathbf{c}$ is the matrix of the scalar values we will multiply the eigenvectors by to get our linear combination:

$$\begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

We first compute A^{-1} so that we can multiply both sides by it to find $\mathbf{c}$. The determinant of A is equal to $-1 - 1 = -2$, so using the formula for the inverse of a 2×2 matrix, we find:

$$A^{-1} = -\frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & -1 \end{bmatrix}$$

Therefore:

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = -\frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & -1 \end{bmatrix}\begin{bmatrix} 2 \\ 0 \end{bmatrix} = -\frac{1}{2}\begin{bmatrix} 2 \\ -2 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

So, $\begin{bmatrix} 2 & 0 \end{bmatrix}^T$ can be written as $-\mathbf{v_1} + \mathbf{v_2}$, where $\mathbf{v_1} = \begin{bmatrix} -1 & 1 \end{bmatrix}^T$ and $\mathbf{v_2} = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$.

Next, to find a closed formula for

$$B^n \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

We can derive it through the following steps:

$$\begin{aligned} \begin{bmatrix} a_n \\ b_n \end{bmatrix} &= B^n \begin{bmatrix} 2 \\ 0 \end{bmatrix} \\ &= B^n(-\mathbf{v_1} + \mathbf{v_2}) \\ &= -B^n\mathbf{v_1} + B^n\mathbf{v_2} \\ &= -\lambda_1^n\mathbf{v_1} + \lambda_2^n\mathbf{v_2} \\ &= -(-1)^n\mathbf{v_1} + 5^n\mathbf{v_2} \end{aligned}$$

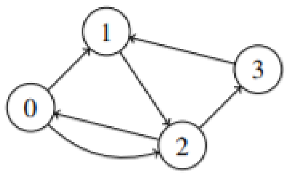
Thus the closed formula is:

$$\begin{bmatrix} a_n \\ b_n \end{bmatrix} = -(-1)^n \begin{bmatrix} -1 \\ 1 \end{bmatrix} + 5^n \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

As a perfunctory check, plugging in the first several powers of n into both the expression given and the closed formula both yield the same values ((2, 0), (4, 6), (26, 24), (124, 126), etc.).

PageRank Theoretical Exercise

Consider the web corresponding to the graph below. The web consists of 4 web pages with links as indicated, i.e. page 0 links to pages 1 and 2, etc.



Construct the matrix M for which the importance score vector $\mathbf{x}$ should be an eigenvector. The matrix M should depend on an unknown damping factor m. For the specific case of $m = 0$, **write out** the 4 linear equations defining the importance scores x_0, x_1, x_2, x_3, and **solve these** to derive a ranking of the pages. Of the two highest ranking pages, **explain in words** why one is considered more important than the other.

Solution:

a. As matrix M is dependent on damping factor m, the matrix is best expressed through the formula:

$$M = (1 - m)(A + D) + mS \tag{1}$$

We first find matrix A with the help of the following formula, letting N represent the set of nodes {0, 1, 2, 3}:

$$A_{rc} = \sum_{c \in L_r} \frac{1}{n_c}$$

The first element of A is given by A_{00} in this case; A_r indicates the row matrix of page $r \in N$, which is why I set matrix indexing to start at 0; L_r is the set of r's backlinks, with $L_r \subset N$; 1 is the naïve importance score; I hardcoded the 1 because we are only computing the values of matrix A with this formula, not the weighted score for a page itself; n_c is the total number of outgoing links from page $c \in N$. We find that matrix A is:

$$A = \begin{bmatrix} 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & 1 \\ \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

As it is clear that A is a column-stochastic matrix, we know that all entries of matrix D are 0, as there are no dangling nodes. S is the $|N| \times |N|$ matrix that contains $\frac{1}{|N|}$ for every entry, or in other words, $\frac{1}{4}$. Thus, M can be written as:

$$M = (1-m)A + mS$$
$$= \begin{bmatrix} 0 & 0 & \frac{1-m}{2} & 0 \\ \frac{1-m}{2} & 0 & 0 & 1-m \\ \frac{1-m}{2} & 1-m & 0 & 0 \\ 0 & 0 & \frac{1-m}{2} & 0 \end{bmatrix} + \begin{bmatrix} \frac{m}{4} & \frac{m}{4} & \frac{m}{4} & \frac{m}{4} \\ \frac{m}{4} & \frac{m}{4} & \frac{m}{4} & \frac{m}{4} \\ \frac{m}{4} & \frac{m}{4} & \frac{m}{4} & \frac{m}{4} \\ \frac{m}{4} & \frac{m}{4} & \frac{m}{4} & \frac{m}{4} \end{bmatrix}$$
$$M = \underline{\underline{\begin{bmatrix} \frac{m}{4} & \frac{m}{4} & \frac{2-m}{4} & \frac{m}{4} \\ \frac{2-m}{4} & \frac{m}{4} & \frac{m}{4} & \frac{4-3m}{4} \\ \frac{2-m}{4} & \frac{4-3m}{4} & \frac{m}{4} & \frac{m}{4} \\ \frac{m}{4} & \frac{m}{4} & \frac{2-m}{4} & \frac{m}{4} \end{bmatrix}}}$$

b. When $m = 0$, notice that the term mS becomes 0, thus $M_0 = (1-0)A = A$:

$$M_0 = A = \begin{bmatrix} 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & 1 \\ \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

The system of linear equations that can be derived from $\boldsymbol{M_0}$ using x as the eigenvector of importance scores is thus (using $\boldsymbol{M_0}\mathbf{x} = \mathbf{x}$ $(\boldsymbol{\lambda} = \mathbf{1})$):

$$\boldsymbol{x_0 = \frac{1}{2}x_2}$$
$$\boldsymbol{x_1 = \frac{1}{2}x_0 + x_3}$$
$$\boldsymbol{x_2 = \frac{1}{2}x_0 + x_1}$$
$$\boldsymbol{x_3 = \frac{1}{2}x_2}$$

c. As M_0 is a column-stochastic matrix (i.e. has no dangling nodes) whose graph is strongly connected, we can directly calculate the importance score vector $\mathbf{x}$ using the formula $(I - M_0)\mathbf{x} = \mathbf{0}$ $(\lambda = 1)$. Thus, row reducing $(I - M_0)$ will give us a matrix that we can use to calculate $\mathbf{x}$:

$$\begin{bmatrix} 1 & 0 & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & 0 & -1 \\ -\frac{1}{2} & -1 & 1 & 0 \\ 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix} \qquad = I - M_0$$

$$\begin{bmatrix} 1 & 0 & -\frac{1}{2} & 0 \\ 0 & 2 & -1 & -1 \\ -\frac{1}{2} & -1 & 1 & 0 \\ 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix} \qquad R_2 - R_3 \to R_2$$

$$\begin{bmatrix} 0 & -2 & \frac{3}{2} & 0 \\ 0 & 2 & -1 & -1 \\ -\frac{1}{2} & -1 & 1 & 0 \\ 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix} R_1 + 2R_3 \to R_1$$

$$\begin{bmatrix} 0 & -2 & \frac{3}{2} & 0 \\ 0 & 2 & -1 & -1 \\ 1 & 2 & -2 & 0 \\ 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix} \quad -2R_3 \to R_3$$

$$\begin{bmatrix} 0 & -2 & \frac{3}{2} & 0 \\ 0 & 2 & -1 & -1 \\ 1 & 0 & -1 & 1 \\ 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix} \quad R_3 - R_2 \to R_3$$

$$\begin{bmatrix} 0 & 0 & \frac{1}{2} & -1 \\ 0 & 2 & -1 & -1 \\ 1 & 0 & -1 & 1 \\ 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix} \quad R_1 + R_2 \to R_1$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & -1 & -1 \\ 1 & 0 & -1 & 1 \\ 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix} \quad R_1 + R_4 \to R_1$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & -1 & -1 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix} \quad R_3 - 2R_4 \to R_3$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & -3 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix} \quad R_2 - 2R_4 \to R_2$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -\frac{3}{2} \\ 1 & 0 & 0 & -1 \\ 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix} \quad \frac{1}{2}R_2 \to R_2$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -\frac{3}{2} \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -2 \end{bmatrix} \quad -2R_4 \to R4$$

$$\begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -\frac{3}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 \end{bmatrix} \quad R_1 \leftrightarrow R_3$$

$$\begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -\frac{3}{2} \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad R_3 \leftrightarrow R_4$$

Solving for $(I - M)\mathbf{x} = \mathbf{0}$, we can set x_3 to be the free variable, thus:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = s \begin{bmatrix} 1 \\ \frac{3}{2} \\ 2 \\ 1 \end{bmatrix}, \; s \in \mathbb{R} \wedge s \neq 0$$

However, we want to scale the above eigenvector such that the sum of its entries adds to 1. We note that the sum as it is right now equals $\frac{11}{2}$, thus we can use $s = \frac{2}{11}$ to get the desired result. **After doing so, we find that the final importance scores are:**

$$\boldsymbol{x_0 = \frac{2}{11}}$$
$$\boldsymbol{x_1 = \frac{3}{11}}$$
$$\boldsymbol{x_2 = \frac{4}{11}}$$
$$\boldsymbol{x_3 = \frac{2}{11}}$$

As a note, using the power method is also valid. That is, we can also calculate $\mathbf{x}$ using the formula:

$$\mathbf{x_k} = M_0 \mathbf{x_{k-1}} = M_0^k \mathbf{x_0}$$
$$\mathbf{x_0} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}^T$$

I assumed here that—since eigenvalue 1 is the dominant eigenvalue of M_0—I could also do without normalizing $\mathbf{x_k}$ each iteration, so I could simply compute $M_0^k \mathbf{x_0}$. Plugging in a sufficiently large k would yield a decent approximation, and as such I used $0 \leq 10k \leq 100$ as a sanity check for my results.

d. The two most important pages are x_2 and x_1 respectively. The former is considered more important because, while both pages receive half of the score of x_0, x_2 also receives the full weight of x_1 added to its score, whereas x_1 receives the full weight of x_3. Because x_3 has a lower total importance to give out compared to x_1, its fully weighted score being added to x_1 is less than the fully weighted score of the x_1 being added to x_2. Thus, the difference between the two most important pages is due to the difference of the non-x_0 pages' scores being added to each respective page x_1 and x_2.

PageRank Efficiency Discoveries (Optional Read)

Before presenting the results, I thought it best to explain how I figured out the efficiency shortcuts for the PageRank algorithm using the hints given in the lecture notes. My code takes advantage of these shortcuts, but I do not attempt to explain how I determined them in the comments. Thus, I will do it here. I will not try to prove anything, and therefore I will be a bit loose by sometimes reasoning from examples and specific cases.

a. $mS\mathbf{x}$

   We can expand this expression for some $n \times n$ matrix S with n nodes to read:

$$m \begin{bmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \ddots & \\ \frac{1}{n} & & \frac{1}{n} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

   Multiplying S through by m so that each term reads $\frac{m}{n}$, and then multiplying the first row of this matrix with $\mathbf{x}$, we see that its value is:

$$\sum_{i=1}^{n} x_i \frac{m}{n}$$

   However, we are given that the sum of the x_i values is 1, so the above expression simplifies to $\frac{m}{n}$. As the rows of S are identical, carrying out multiplication will yield the same value for each component in the resulting column vector. Thus, the entire term simplifies to the constant $\frac{m}{n}$ (repeated as necessary to fill the column vector).

b. $(1-m)D\mathbf{x}$

   As per the definition of D, its columns are either all zeroes or all a constant value. Again by definition, this constant value is identical across all non-zero columns of D (having the value $\frac{1}{n}$). We will reason by example and claim that it can be generalized. Using matrix A below:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 0 \end{bmatrix}$$

   We see that the third column represents a dangling node. Thus, D is:

$$D = \begin{bmatrix} 0 & 0 & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} \end{bmatrix}$$

Writing out the full expression $(1-m)D\mathbf{x}$ (multiplying $(1-m)$ with D), we get:

$$\begin{bmatrix} 0 & 0 & \frac{1-m}{3} \\ 0 & 0 & \frac{1-m}{3} \\ 0 & 0 & \frac{1-m}{3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Multiplying through, we see that the resulting column vector contains the value $x_3\frac{1-m}{3}$ for all of its components. It is easy to see that if there were more dangling nodes in the graph, then this value would—for each additional dangling node—increase by the score of that node multiplied by $\frac{1-m}{3}$. The claim is that this generalizes to any $n \in \mathbb{Z}^+$ nodes with any $1 \leq d \leq n$ dangling nodes. Thus we are in a situation similar to $mS\mathbf{x}$, except that we have to multiply through by the new $\mathbf{x}$ each iteration. In any case, using the generalization that we assume holds, every element in the column vector of $(1-m)D\mathbf{x}$ is:

$$\sum_{x_i \in D} x_i \frac{1-m}{n}, \; D = \{d \mid d \,\text{is a dangling node}\}$$

c. $(1-m)A\mathbf{x}$

For this term, I implemented A and the column vector of $(1-m)A\mathbf{x}$ as `dict`s. This was what allowed me to only multiply for those non-empty components of A. However, this implementation requires a number of `dict` look-ups each iteration, which makes my code about 10 times as slow as what I have heard others be able to achieve for 1000 iterations of Gnutella (after optimizing my code so I don't spend extra cycles worrying about all the stability and precision checks I typically perform).

PageRank and Random Surfer Results

First, to run the program provided: `python pagerank.py [filename]`

```
m = 0.15
MIN_SCORE: 300000

Most visited nodes according to random surfer:

Time: 2000.006557
Iterations: 125528082 (total score)
Normalized score: 1.000000

Node  367 (normalized: 0.002390; score: 300000)
Node  249 (normalized: 0.002183; score: 273988)
Node  145 (normalized: 0.002059; score: 258457)
Node  264 (normalized: 0.001999; score: 250897)
Node  266 (normalized: 0.001960; score: 246091)
Node  123 (normalized: 0.001866; score: 234283)
Node  127 (normalized: 0.001854; score: 232729)
Node  122 (normalized: 0.001853; score: 232606)
Node 1317 (normalized: 0.001840; score: 231029)
Node    5 (normalized: 0.001831; score: 229832)

m = 0.15
DELTA_NORMAL: 1e-06

Highest ranking nodes according to PageRank:

Time: 8.771836
Iterations: 8
Stable at: 5
Sum of scores: 1.000000

Node  367 (score: 0.002388)
Node  249 (score: 0.002184)
Node  145 (score: 0.002055)
Node  264 (score: 0.001999)
Node  266 (score: 0.001963)
Node  123 (score: 0.001863)
Node  127 (score: 0.001861)
Node  122 (score: 0.001853)
Node 1317 (score: 0.001844)
Node    5 (score: 0.001831)
```

Above is a slightly trimmed down version of actual output from the program using the Gnutella graph, removing a few superfluous lines so that it can be shown on one page. Before I discuss some of the secondary information, I should first mention how many iterations it took for Gnutella's top nodes to stabilize in the PageRank algorithm.

In my view, there are two ways to define ”stable” in this instance: a strict definition and a loose definition. The strict version requires that the top nodes have been in the same ordering since at least the previous iteration, whereas the loose version only requires all subsequent iterations to have the same ordering. Using the strict definition, it takes 5 iterations (as can be seen in the output above); using the loose definition, it only takes 4. I employed the strict definition in my code, as it was easier to implement.

Onto explaining a bit of the output: Starting from the top, I output the constants that were used in running the random surfer algorithm. You can see the comments towards the top of the program or deduce from context how `MIN_SCORE` works. While the main loop in random surfer is a **`while`** loop, it essentially functions as a glorified **`for`** loop (since `MIN_SCORE` is some constant), with `MIN_SCORE` playing a key role in determining number the loops that will be completed, in combination of course with how many nodes there are in the graph. `MIN_SCORE`'s purpose is to give a more intuitive feel for how the accuracy of the output changes as its value changes, which is why I kept it in favor of using a **`for`** loop with some adjustable value. This could lead to slower performance for larger n, as I check to see whether the highest score needs to be updated each iteration. Note that the number of iterations for random surfer also represents the total number of times that nodes were looked up, whether by traversing an edge or by randomly jumping. The time is in seconds, and it is so high here due to both the value of `MIN_SCORE` and the number of nodes in Gnutella. You can lower `MIN_SCORE` to greatly reduce the time taken by random surfer, but it will also lower the accuracy. For example, setting `MIN_SCORE` to `10000`, random surfer took about a minute to run using the Gnutella graph on my machine, and it still produced decent results.

Moving on to PageRank, the algorithm uses a constant called `DELTA_NORMAL`, which essentially determines the precision to which the normalized scores of the top nodes are calculated. `Iterations` then is a measure of how many times the mathematical formula was applied until the scores of all the top nodes did not change by more than `DELTA_NORMAL`.

The reason why I set `MIN_SCORE` so high was to try and find where random surfer's top nodes stabilize. As you can see, the ordering does indeed match, but it took over 33 minutes and $1.26 \cdot 10^8$ iterations to get normalized scores that were no more than $7 * 10^{-6}$ off from PageRank's scores. Even still, random surfer isn't yet stable at this number of iterations, as this maximum difference is still greater than the maximum tolerance of $4 * 10^{-6}$ allowed for by PageRank—see the scores for nodes 123 and 127 under PageRank, keeping in mind the precision to which they were calculated. It just so happened that in this run, the differences were such that the ordering matched.

Below I've included the extended output from the same run as above. This output is turned off by default in the version I've turned in, as it is not part of the requirements. Nevertheless, I find it useful, and you can follow the instructions in the comments in order to enable it. Finally, make sure to check the value of `MIN_SCORE` before you attempt to run the code on a large dataset. The version I've turned in has it set to `10000` by default. You may choose not to run random surfer at all for a large dataset, in which case you can simply comment out `surf_top_nodes = print_surfer_results(dg)` at the bottom of the code, meaning you also cannot run the comparison function.

```
Top 10 nodes for PageRank and random surfer respectively:

['367', '249', '145', '264', '266', '123', '127', '122', '1317', '5']
['367', '249', '145', '264', '266', '123', '127', '122', '1317', '5']

Absolute differences between the top 10 nodes:
(Using top nodes of PageRank as point of comparison)

Node  367 (difference: 0.000002)
Node  249 (difference: 0.000002)
Node  145 (difference: 0.000004)
Node  264 (difference: 0.000000)
Node  266 (difference: 0.000003)
Node  123 (difference: 0.000003)
Node  127 (difference: 0.000007)
Node  122 (difference: 0.000000)
Node 1317 (difference: 0.000003)
Node    5 (difference: 0.000000)

Greatest absolute difference: 0.000007
Sum of differences: 0.000024
```