General Instructions

Submission instructions: You should submit your answers via GradeScope and your code via Snap submission site.

Submitting answers: Prepare answers to your homework into a single PDF file and submit it via http://gradescope.com. Make sure that answer to each question is on a separate page. This means you should submit a 22-page PDF (1 page for the cover sheet, 5 pages for the answers to question 1, 3 pages for answers to question 2, 8 pages for question 3, and 5 pages for question 4). On top of each page write the number of the question you are answering. Please find the cover sheet and the recommended templates located here:
http://cs224w.stanford.edu/homework(hw2/submission_template_hw2.tex
http://cs224w.stanford.edu/homework(hw2/submission_template_hw2.pdf

Not including the cover sheet in your submission will result in a 2-point penalty.

Submitting code: Upload your code at http://snap.stanford.edu/submit. Put all the code for a single question into a single file and upload it.

Questions

We strongly encourage you to use Snap.py for Python. However, you can use any other graph analysis tool or package you want (SNAP for C++, NetworkX for Python, JUNG for Java, etc.). See http://cs224w.stanford.edu/resources.html for more details.

1 Broadcasting in the Network [25 points - Paris]

A very fruitful avenue to design centrality measures is to consider a stochastic (random) process on the graph and extract the importance of nodes by optimizing some “natural” objective. Here we consider that there is an undirected graph \( G(V, E) \) with adjacency matrix \( A \in \{0, 1\}^{n \times n} \) and that during each round \( k \):

- **Graph process:** each vertex sends a (different) message to each of its neighbors independently at a uniformly random time between \([k, k+1)\), i.e., for each edge \((i, j)\), \(T_{ij}\) (respectively \(T_{ji}\)) is the time \(i\) (resp. \(j\)) sends a message to \(j\) (resp. \(i\)) and is uniformly distributed in \([k, k+1)\) independently from all other edges.

- **Action:** at any time \(t \in [k, k+1)\) we can “bug” any (more than one) node in the network and intercept the incoming messages. Let \(t_i\) be the total time we spent at node \(i\) during this round. The probability of us being caught by node \(i\) at the end of the round is \(t_i^\beta\), where \(\beta > 0\).
Our objective is to specify the total times \( t = (t_1, \ldots, t_n) \) that we are going to monitor each node, in order to maximize the expected number of messages intercepted during a single round, while making sure that the probability that we get caught is smaller than a number \( \gamma < 1 \). We assume that messages from previous rounds are not retransmitted.

(a) [5 points] Give an expression for the expected number of messages \( m_i(t_i) \) intercepted from node \( i \), if we monitor it for \( t_i \) time in total.

(b) [5 points] Let \( A_i(t_i) \) be the event that we get caught at node \( i \) at the end of the round if we monitor it for \( t_i \) total time. Using the fact that for any set of events \( A_1, A_2: \Pr(A_1 \cup A_2) \leq \Pr(A_1) + \Pr(A_2) \), prove that the probability of being caught by the end of the round is less than \( r(t) := \sum_{i=1}^{n} t_i^{\beta} \).

(c) [5 points] For \( \beta = 2 \) and any \( \gamma < 1 \) find a strategy \( (t_1, \ldots, t_n) \) that maximizes the total expected number of intercepted messages \( \sum_i m_i(t_i) \) subject to the fact that \( r(t) \leq \gamma \). Comment on the resulting centrality measure. Hint: Cauchy-Schwarz inequality.

(d) [5 points] Repeat part (c) for \( \beta = 1 \) and assuming that all degrees are distinct. What changed? Comment on the solution when the degrees are not necessarily distinct.

(e) [5 points, Bonus] What is the expression for any \( \beta \neq 1 \)?

★ SOLUTION:

(a) Let \( S_i \) be the set of times (intervals) that we bugged node \( i \). For each edge \((i, j)\) The probability of intercepting the message sent by node \( j \) is:

\[
\Pr(\text{intercepted } (i, j)) = \Pr(T_{ij} \in S_i) = \frac{|S_i|}{1} = t_i
\]

Thus, the expected number of messages intercepted at \( i \) are:

\[
m_i(t_i) = \mathbb{E}[\sum_j \mathbb{I}(\text{intercepted } (i, j))] = \sum_{j: (i,j) \in E} \Pr(T_{ij} \in S_i) = t_i d_i
\]

(b) To be caught at the end of the round it is enough to be caught by at least one node. If \( A(t) \) denotes the event that we are caught, then \( A(t) = A_1(t_1) \cup \ldots \cup A_n(t_n) \). Hence,

\[
\Pr(A(t)) = \Pr(A_1(t_1) \cup \ldots \cup A_n(t_n)) \leq \sum_i \Pr(A_i(t_i)) = \sum_i t_i^{\beta}
\]

where the inequality follows by invoking iteratively the given inequality.

(c) The problem we wish to solve is the following:

\[
\max \quad \sum_i t_i d_i
\]

subject to\[
\sum_i t_i^2 \leq \gamma
\]

Using our linear algebra background we identify the constraint as \( \|t\|_2^2 \leq \gamma \) and the objective as \( t^\top d \), where \( d \) is the vector of degrees. By the inequality Cauchy-Schwarz (or directly by solving the optimization problem), we know that \( t^\top d \leq \|t\|_2 \|d\| \leq \sqrt{\gamma} \|d\| \). If we set \( t^* = \frac{d}{\sqrt{d}} \sqrt{\gamma} \), the upper bound is achieved and the constraint obviously satisfied. Thus, for \( \beta = 2 \) this strategy recovers degree centrality (up to a proportionality) constant.
(d) For $\beta = 1$, we have the linear constraint $\sum_i t_i \leq \gamma$ and we wish to maximize $\sum_i t_i d_i$. Let $i^*$ be the index with the maximum degree $d_{i^*}$. Then for all $\sum_i t_i \leq \gamma$ we have that: $\sum_i t_i d_i \leq d_{i^*} \sum_i t_i \leq d_{i^*} \gamma$. Hence the optimal strategy is to only bug the node with the highest degree. In this question, if there are more than one nodes with maximum degree then any strategy that bugs only highest degree nodes is optimal and our centrality measure essentially identifies the highest degree nodes.

(e) To get a general expression for $\beta \neq 1$, we may use Lagrangian multiplier method. We form the augmented Lagrangian:

$$L(t, \lambda) = t^\top d - \lambda \sum_i t_i^\beta$$

Setting the gradient equal to zero, we get that $d_i - \lambda \beta t_i^{\beta-1} = 0$ and hence that:

$$t_i = \beta^{-1} \sqrt{\frac{d_i}{\lambda \beta}}$$

In order to satisfy the constraint, we get that $\lambda \beta = \|d\|_{\frac{1}{\beta-1}} \gamma^{-\frac{\beta-1}{\beta-1}}$. This gives us the required expression. We see that by tweaking $\beta$ we can add more or less weight to degrees and thus make our centrality measure more or less concentrated on high degree nodes. The extreme case as we saw was for $\beta = 1$ where only high nodes are meaningful. Equivalently, one could use the same approach with the previous two parts and derive an upper bound using Hölder’s inequality:

$$\sum_i t_i d_i \leq \|t\|_p \|d\|_p$$

for any $p, q \geq 0$ such that $\frac{1}{p} + \frac{1}{q} = 1$ (a generalization of the Cauchy-Schwarz inequality).

2 Signed Triad Analysis [15 points - Sameep, Nihit]

Download the Epinion dataset:

Epinion is a consumer review site where members can decide if they trust each other or not. This leads to a web of trust which you will analyze in this question.

Since a member can either trust or distrust other members, we have a special type of network called signed network where an edge has a sign. Positive sign on edge indicates trust relationship whereas negative edge indicates distrust relationship.

We will consider the graph as undirected and study the various forms of triads as shown in Figure 1.

(a) [5 points] Calculate the count and fraction of triads of each type in the opinion network.

(b) [5 points] Calculate the fraction of positive and negative edges in the graph. Let fraction of positive edges be $p$. Assuming that an edge of triad will take positive sign with probability $p$ and negative sign with probability $(1-p)$ independently, calculate the probability of triad of each combination of signs to be formed.

(c) [5 points] Compare the probabilities from part (b) with fractions calculated in part (a). Which type of triads do you see more in data as compared to baseline values? Which type of triads do you see less? Provide an explanation for this observation.
Figure 1: Triads

(a) $t_0$

(b) $t_1$

(c) $t_2$

(d) $t_3$

★ SOLUTION:

(a) Results given in table 1

<table>
<thead>
<tr>
<th>Triad type</th>
<th>Count</th>
<th>Fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_0$</td>
<td>58732</td>
<td>0.012</td>
</tr>
<tr>
<td>$t_1$</td>
<td>396548</td>
<td>0.081</td>
</tr>
<tr>
<td>$t_2$</td>
<td>451711</td>
<td>0.092</td>
</tr>
<tr>
<td>$t_3$</td>
<td>4003085</td>
<td>0.815</td>
</tr>
</tbody>
</table>

(b) Number of positive edges = 592551
Number of negative edges = 119232
Fraction of positive edges = $p = 0.832$
Fraction of negative edges = 0.168

Baseline probabilities are given in table 2

<table>
<thead>
<tr>
<th>Triad type</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_0$</td>
<td>0.005</td>
</tr>
<tr>
<td>$t_1$</td>
<td>0.070</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0.349</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0.576</td>
</tr>
</tbody>
</table>

(c) Triads $t_0, t_1, t_3$ are seen more in the graph as compared to baseline whereas triads $t_2$ are seen significantly less than baseline.
Reason: Triad $t_2$ is intuitively unstable since B’s distrust of C conflicts with B’s trust of A who trusts C.
3 Decentralized Search [40 points – Caroline]

In class, we saw a decentralized search algorithm based on geography that seeks a path between two nodes on a grid by successively taking the edge towards the node closest to the destination.

Here we will examine an example of a decentralized search algorithm on a network whose nodes reside in the leaves of a tree. The tree may, for instance, be interpreted as representing the hierarchical organization of a university where one is more likely to have friends inside the same department, a bit less likely in the same school, and the least likely across schools.

Let us organize students at Stanford into a tree hierarchy, where the root is Stanford University and the second level contains the different schools (engineering, humanities, etc.). The third level represents the departments and the final level (i.e., the leaves) are the Stanford students.

Tom, a student from the computer science department, wants to hang out with Mary, who is in sociology. If Tom does not know Mary, he could ask a friend in the sociology department to introduce them. If Tom does not know anybody in the sociology department, he may seek a friend in the Stanford humanities school instead. In general, he will try to find a friend who is “close” to Mary in the tree.

There are three parts in this problem. The first two parts explore an effective decentralized search algorithm on the hierarchical model in a specific setting. The third part involves simulation experiments on the model under a more general setting. There are many subproblems, but do not panic. A good understanding of the math involved in the lattice model we covered in class will make this problem easy and shorten each subproblem to just a few line proofs.

Basic Tree Properties [13 points]

This part covers some basic facts of the setting of the hierarchical model and defines the notion of tree based “distance” between the nodes.

Consider a complete, perfectly balanced $b$-ary tree (each non-leaf node has $b$ children and $b \geq 2$) $T$, and a network whose nodes are the leaves of $T$ (the red nodes in the picture). Let the number of network nodes (equivalently, the number of leaves in the tree) be $N$ and let $h(T)$ denote the height of $T$.

**One important thing to keep in mind:** In this problem, the network nodes are only the leaf nodes in the tree. Other nodes in the tree are virtual and only there to determine the edge creation probabilities between the nodes of the network.

(a) [4 points] Write $h(T)$ in terms of $N$.

★ SOLUTION: Since the number of leaves of a tree of height $h(T)$ is $b^{h(T)}$, we have $h(T) = \log_b N$.

Given two network nodes (leaf nodes) $v$ and $w$, let $L(v, w)$ denote the subtree of $T$ rooted at the lowest common ancestor of $v$ and $w$, and let $h(v, w)$ denote its height (that is, $h(L(v, w))$). In our illustration Fig 2, $L(u, t)$ is the tree in the circle and $h(u, t) = 2$. 
Figure 2: Illustration for Question 3 of the hierarchical graph. Black nodes and edges are used to illustrate the hierarchy and structure, but are not part of our network. Red nodes (leaf nodes) and red edges are the ones in our network. The lowest common ancestor of \( s \) and \( t \) is the root of the tree. The decentralized search proceeds as follows. Denote the starting node by \( s \) and the destination by \( t \). At each step, the algorithm looks at the neighbors of the current node \( s \) and moves to the one “closest” to \( t \), that is, the algorithm moves to the node with the lowest common ancestor with \( t \). In this graph, from \( s \) we move to \( u \).

(b) [4 points] Next we want to define the notion of “tree distance.” The intuition we want to capture is that students that share the same department are closer than for example students sharing schools. For instance, in the tree in Figure 2 nodes \( u \) and \( t \) are “closer” than nodes \( u \) and \( s \). We formalize the notion of “distance” as follows:

Given two network nodes (leaf nodes) \( v \) and \( w \), let \( L(v, w) \) denote the subtree of \( T \) rooted at the lowest common ancestor of \( v \) and \( w \), and \( h(v, w) \) denote its height (that is, \( h(L(v, w)) \)). In Figure 2, \( L(u, t) \) is the tree in the circle and \( h(u, t) = 2 \). Note that we can think of \( h(v, w) \) as a “distance” between nodes \( v \) and \( w \).

For a given node \( v \), what is the maximum possible value of \( h(v, w) \)?

★ SOLUTION: The largest \( h(v, w) \) will happen when the least common ancestor of \( v \) and \( w \) is the root of \( T \). Such a node \( w \) always exists. Therefore \( h(v, w) = \log_b(N) \).

(c) [5 points] Given a value \( d \) and a network node \( v \), show that there are \( b^d - b^{d-1} \) nodes satisfying \( h(v, w) = d \).

★ SOLUTION: Let \( T_{v,d} \) be the subtree of \( T \) containing \( v \) and of height \( d \). It is easy to see that the leaves of \( T_{v,d} - T_{v,d-1} \) satisfy \( h(v, w) = d \). Since the number of leaves of a tree of height \( h(T) \) is \( b^{h(T)} \), there are \( b^d - b^{d-1} \) nodes satisfying \( h(v, w) = d \).
Network Path Properties [20 points]

This part helps you design a decentralized search algorithm in the network.

We will generate a random network on the leaf nodes in a way that models the observation that a node is more likely to know “close” nodes than “distant” nodes according to our university organizational hierarchy captured by the tree $T$. For a node $v$, we define a probability distribution of node $v$ creating an edge to any other node $w$:

$$p_v(w) = \frac{1}{Z} b^{-h(v,w)}$$

where $Z = \sum_{w \neq v} b^{-h(v,w)}$ is a normalizing constant. By symmetry, all nodes $v$ have the same normalizing constant.

Next, we set some parameter $k$ and ensure that every node $v$ has exactly $k$ outgoing edges. We do this with the following procedure. For each node $v$, we repeatedly sample a random node $w$ according to $p_v$ and create edge $(v,w)$ in the network. We continue this until $v$ has exactly $k$ neighbors. Equivalently, after we add an edge from $v$ to $w$, we can set $p_v(w)$ to 0 and renormalize with a new $Z$ to ensure that $\sum_w p(w) = 1$. This results in a $k$-regular directed network.

(d) [5 points] Show that $Z \leq \log_b N$. (Hint: use the results in (ii) and (iii).)

★ SOLUTION: Notice that we can group nodes $w$ according to $h(v,w)$, which ranges from 1 to $\log N$. From (c), the number of nodes $w$ satisfying $h(v,w) = i$ is $b^i - b^{i-1}$. This substitution also allows us to say that $b^{-h(v,w)} = b^{-i}$.

\[
\sum_{w \neq v} b^{-h(v,w)} = \sum_{i=1}^{\log_b N} (b^i - b^{i-1}) b^{-i} = \sum_{i=1}^{\log_b N} (1 - \frac{1}{b}) \leq \log_b N
\]

(e) [5 points] For two leaf nodes $v$ and $t$, let $T'$ be the subtree of $L(v,t)$ satisfying:

- $T'$ is of height $h(v,t) - 1$,
- $T'$ contains $t$,
- $T'$ does not contain $v$.

For instance, in Fig. 2, $T'$ of $L(s,t)$ is the tree in the circle.
Let us consider node $v$ and an edge $e$ from $v$ to a random node $u$ sampled from $p_v$. We say that $e$ points to $T'$ if $u$ is a leaf node of $T'$. Show that the probability of $e$ pointing to $T'$ is no less than $\frac{1}{b \log_b N}$.

★ SOLUTION: Let $d = h(v,t)$. For any of the $b^{d-1}$ leaves $u$ in $T'$, we have $h(v,u) = d$. The probability of $v$ linking an edge to $u$ is $\frac{b^{-d}Z}{Z}$. Therefore the probability of $v$ linking an edge to $T'$ is larger than

$$b^{d-1}b^{-d}Z = \frac{1}{bZ} \geq \frac{1}{b \log_b N} \quad (4)$$

(f) [5 points] Let the out-degree $k$ for each node be $c \cdot (\log_b N)^2$, where $c$ and $b$ are constants. Show that when $N$ grows very large, the probability of $v$ not having any edge pointing to $T'$ is asymptotically no more than $N^{-\theta}$, where $\theta$ is a positive constant which you need to compute.

(Hints: Use the result in (v) and recall that each of the $k$ edges is independently created. Also, use $\lim_{x \to \infty} (1 - \frac{1}{x})^x = \frac{1}{e}$.)

Argue why the above result indicates that for any node $v$, we can, with high probability, find an edge to a (leaf) node $u$ satisfying $h(u,t) < h(v,t)$.

★ SOLUTION: We can view the creation of edges from $v$ to be independent. Therefore, the probability of $v$ having no edges to $T'$ is smaller than

$$ (1 - \frac{1}{b \log_b N})^{c(\log_b N)^2} \quad (5) $$

$$ = \left( (1 - \frac{1}{b \log_b N})^{b(\log_b N) \frac{c}{b}} \right) \quad (6) $$

$$ = \left( \frac{1}{e} \right)^{c \frac{\ln N}{\ln b}} \quad (7) $$

$$ = N^{-\frac{c}{b \ln b}} \quad (8) $$

As $\theta$ is a positive constant, $N^{-\theta}$ is very small, so the probability that a node $v$ cannot find an edge to a (leaf) node $u$ satisfying $h(u,t) < h(v,t)$ is very small.

(g) [5 points] Show that starting from any (leaf) node $s$, within $O(\log_b N)$ steps, we can reach any (leaf) node $t$. You do not need to prove it in a strict probabilistic argument. You can just assume that for any (leaf) node $v$, you can always get to a (leaf) node $u$ satisfying $h(u,t) < h(v,t)$ and argue why you can reach $t$ in $O(\log_b N)$ steps.

★ SOLUTION: Notice that at each node $v$, we can always find $v$ pointing to a node $u$ with $h(u,t) < h(v,t)$. Since $h(v,t) < \log_b N$, in $\log_b(N)$ steps we can reach the destination.
Simulation [7 points]

In (i) to (vii), we have set the theory to find an efficient decentralized search algorithm, assuming that for each edge of \(v\), the probability of it going to \(w\) is proportional to \(b^{-h(v,w)}\). Now we experimentally investigate a more general case where the edge probability is proportional to \(b^{-\alpha h(v,w)}\). Here \(\alpha > 0\) is a parameter in our experiments.

In the experiments below, we consider a network with the setting \(h(T) = 10\), \(b = 2\), \(k = 4\), and a given \(\alpha\). That is, the network consists of all the leaves in a binary tree of height 10; the out degree of each node is 5. Given \(\alpha\), we create edges according to the distribution described above.

(h) [7 points] Create random networks for \(\alpha = 0.1, 0.2, \ldots, 10\). For each of these networks, sample 1,000 random \((s,t)\) pairs \((s \neq t)\). Then do a decentralized search starting from \(s\) as follows. Assuming that we are currently at (leaf) node \(s\), we pick its neighbor \(u\) (also a leaf node) with smallest \(h(u,t)\) (break ties arbitrarily). If \(u = t\), the search succeeds. If \(h(s,t) > h(u,t)\), we set \(s\) to \(u\) and repeat. If \(h(s,t) \leq h(u,t)\), the search fails.

For each \(\alpha\), pick 1,000 pairs of nodes and compute the average path length for the searches that succeeded. Then draw a plot of the average search time (number of steps it takes to reach \(t\)) as a function of \(\alpha\). Also, plot the search success probability as a function of \(\alpha\).

★ SOLUTION: As long as your plots showed the tendency of increasing as \(\alpha\) approaches 1, decreasing as \(\alpha\) is away from 1, and can reasonably explain the shape, you should have gotten full credit on this problem.

Figure 3: Search Time Alpha

Briefly comment on the plots and explain the shape of the curve.

★ SOLUTION: Since our out-degree is pretty small, we can think that most successful searches can finish in only a few steps.
For small $\alpha$, we will have many long edges. For large $\alpha$, we will have too many short edges. If we have too many long edges (small $\alpha$), we fail for those close pairs (low success rate) but will quickly finish the search for distant pairs (short search length). On the contrary we will fail for distant pairs (low success rate) but quickly finish the search for close pairs (short search length). If we balance the trade off between short and long edges, we can manage more pairs (high success rate) but need to take more time to finish the search (long search length).

What to submit

(a) Write the expression and a short explanation.
(b) Write the expression and a short explanation.
(c) Write a short proof.
(d) Write a short proof.
(e) Write a short proof.
(f) Write a short proof, give an expression for $\theta$ and brief argument.
(g) Write a short proof.
(h) Provide both plots and a brief comment. Upload code to [http://snap.stanford.edu/submit](http://snap.stanford.edu/submit)

4 Variations on a Theme of PageRank [25 points – Omid]

Personalized PageRank

Personalizing PageRank is a very important real-world problem: different users find different pages relevant, so search engines can provide better results if they tailor their page relevance estimates.
to the users they are serving. Recall from class that PageRank can be specialized with clever modifications of the teleport vector. In this question, we’ll explore how this can be applied to personalize the PageRank algorithm.

Assume that people’s interests are represented by a set of representative pages. For example, if Zuzanna is interested in sports and food, then we could represent her interests with the set of pages \{www.espn.com, www.epicurious.com\}. For notational convenience, we’ll use integers as names for webpages.

(a) [7 points] Suppose you have already computed the personalized PageRank vectors for the following users:

- Agatha, whose interests are represented by the teleport set \{1, 2, 3\},
- Bertha, whose interests are represented by the teleport set \{3, 4, 5\},
- Clementine, whose interests are represented by the teleport set \{1, 4, 5\}, and
- DeShawn, whose interests are represented by the teleport set \{1\}.

Without looking at the graph, can you compute the personalized PageRank vectors for the following users? If so, how? If not, why not? Assume a fixed teleport parameter \(\beta\).

\[ M' r_C = \beta Mr_C + (1 - \beta) s_C \]

i. [2 points] Eloise, whose interests are represented by the teleport set \{2\}.

\[ M' r_C = \beta Mr_C + (1 - \beta) s_C = q (\beta Mr_A + (1 - \beta) s_A) + (1 - q) (\beta Mr_B + (1 - \beta) s_B) = qr_A + (1 - q)r_B = r_C \]

\[ \text{SOLUTION:} \quad \text{Yes.} \ 3A - 3B + 3C - 2D. \]

ii. [2 points] Felicity, whose interests are represented by the teleport set \{5\}.

\[ \text{SOLUTION:} \quad \text{No. There is no way to isolate 5 without involving B, which in turn involves A, and there is no way to cancel out the resultant 2.} \]

iii. [3 points] Glynnis, whose interests are represented by the teleport set \{1, 2, 3, 4, 5\} with weights 0.1, 0.2, 0.3, 0.2, 0.2, respectively.
(b) **[3 points]** Suppose that you’ve already computed the personalized PageRank vectors of a set of users (denote the computed vectors $V$). What is the set of all personalized PageRank vectors that you can compute from $V$ without accessing the web graph?

**SOLUTION:** Any linear combination of the vectors in $V$ can be computed without accessing the graph, so the set of all PPR vectors is span$(V)$.

**Spam Farms**

The staggering number of people who use search engines to find information every day makes having a high PageRank score a valuable asset, which creates an incentive for people to game the system and artificially inflate their website’s PageRank score. Since the PageRank algorithm is based on link structure, many PageRank spam attacks use special network configurations to inflate a target page’s PageRank score. We’ll explore these configurations, called *spam farms*, in this part of the question.

(c) **[5 points]** Consider the spam farm shown in Figure 5. The spammer controls a set of *boosting pages* $1, \ldots, k$ (where page $i$ has PageRank score $p_i$) and is trying to increase the PageRank of the *target page* $p_0$. The target page receives $\lambda$ amount of PageRank from the rest of the graph (represented by the dotted arrow). This means that $\lambda = \sum_{i \in S} \frac{r_i}{d_i}$, where $S$ is the set of nodes in the rest of the network that link to $p_0$ and $r_i$ and $d_i$ represent, respectively, node $i$’s PageRank score and outdegree. Let $N$ denote the number of pages in the entire web, including the boosting pages and target page. Calculate the PageRank of the target page $p_0$ with this configuration as a function of $\lambda$, $k$, $\beta$, and $N$. Your solution should not include other parameters (such as $p_i$).

*Hint: You can write an expression for $p_0$ in terms of $p_1, \ldots, p_k$, $\beta$, $k$, and $N$. You can also write the PageRanks of the boosting pages in terms of $p_0$, $\beta$, $k$, and $N$.**

**SOLUTION:** [Note: this is assuming that $\lambda$ is multiplied by $\beta$]

$p_0$ receives PageRank from the accessible pages, the boosting pages, and random restarts. Specifically,

$$p_0 = \beta \lambda + \beta \sum_{i=1}^{k} p_i + \frac{1-\beta}{N}$$

In addition, for each $i \in [1, k]$, $p_i = \frac{1-\beta}{N} + \frac{\beta}{k} p_0$.

Substituting this back in, we have:

$$p_0 = \beta \lambda + \beta \sum_{i=1}^{k} \left( \frac{1-\beta}{N} + \frac{\beta}{k} p_0 \right) + \frac{1-\beta}{N}$$

$$= \beta \lambda + \frac{1-\beta}{N} (\beta k + 1) + \beta^2 p_0$$

$$= \frac{1}{1-\beta^2} \left( \beta \lambda + \frac{1-\beta}{N} (\beta k + 1) \right)$$

$$= \frac{\beta \lambda}{1-\beta^2} + \frac{\beta k + 1}{N (1 + \beta)}$$
Figure 5: A spam farm.

(d) [5 points] It turns out that the structure in Figure 5 is optimal, in the sense that it maximizes the target’s PageRank $p_0$ with the resources available. However, it may still be possible to do better by joining forces with other spammers. It also turns out that the $\lambda$ contribution from the rest of the graph is mathematically equivalent to having some extra number of boosting pages, so for the rest of this question we’ll ignore $\lambda$. Consider the case where two spammers link their spam farms by each linking to the other’s target page as well as their own, as shown in Figure 6. Let $p'_0$ and $q'_0$ denote the PageRank values of the target pages if the spammers use the individual spam configuration that we discussed in the previous question (without $\lambda$ this time). Calculate the PageRanks of the target pages $p_0$ and $q_0$ with the new configuration as a function of $k$, $m$, $\beta$, and $N$ (where again $N$ denotes the number of pages in the web graph).

What are $p_0 - p'_0$ and $q_0 - q'_0$? Are the spammers better off than they would be if they operated independently (i.e. is $p_0 + q_0 > p'_0 + q'_0$)?

★ SOLUTION: They do not benefit by linking this way.

From (c), $p'_0 = \frac{\beta k+1}{N(1+\beta)}$ and $q'_0 = \frac{\beta m+1}{N(1+\beta)}$, so $p'_0 + q'_0 = \frac{\beta(k+m)+2}{N(1+\beta)}$.

Note also that $\overline{p_0} = \overline{q_0}$. Let $y = p_i = q_j \forall i \in [1,k]$ and $\forall j \in [1,m]$. Then $y = \frac{1-\beta}{N} + \frac{2\beta p_0}{k+m}$.

$$\overline{p_0} = \overline{q_0} = \frac{1-\beta}{N} + \frac{2}{N} \sum_{i=1}^{k+m} y$$

$$= \frac{1-\beta}{N} + \frac{\beta}{N} \sum_{i=1}^{k+m} \left( \frac{1-\beta}{N} + \frac{2\beta p_0}{k+m} \right)$$

$$= \frac{1-\beta}{N} \left( 1 + \frac{\beta}{2} (k+m) \right) + \beta^2 p_0$$

$$= \frac{2+\beta(k+m)}{2N(1+\beta)}$$

so $\overline{p_0} + \overline{q_0} = \frac{\beta(k+m)+2}{N(1+\beta)} = p'_0 + q'_0$. Linking the spam farms does not help.

(e) [5 points] There are other ways spammers can form alliances. Consider the setup shown in Figure 7, where the spammers only link their target pages together. Again let $p'_0$ and $q'_0$ denote the PageRank values of the target pages that the spammers would get on their own. Calculate the PageRank of the target pages $\overline{p_0}$ and $\overline{q_0}$ with this configuration as a function of $k$, $m$, $\beta$, 

...
and $N$ (where again $N$ denotes the number of pages in the web graph). What are $\overline{p_0} - p_0'$ and $\overline{q_0} - q_0'$? Are the spammers better off than they would be if they operated independently (i.e. is $\overline{p_0} + \overline{q_0} > p_0' + q_0'$)?

**SOLUTION:** The spammers benefit by linking.

From (c), $p_0' = \frac{\beta k + 1}{N(1+\beta)}$ and $q_0' = \frac{\beta m + 1}{N(1+\beta)}$, so $p_0' + q_0' = \frac{\beta(k+m)+2}{N(1+\beta)}$.

\[
\overline{p_0} = \frac{1-\beta}{N} + \beta \overline{p_0} + \beta \sum_{i=1}^{k} p_i = \beta \overline{p_0} + (\beta k + 1) \frac{1-\beta}{N},
\]
\[
\overline{q_0} = \frac{1-\beta}{N} + \beta \overline{q_0} + \beta \sum_{i=1}^{m} q_i = \beta \overline{q_0} + (\beta m + 1) \frac{1-\beta}{N}.
\]

Comparing with $p_0'$ and $q_0'$, we can state that $\overline{p_0} = p_0' + \mu k + \eta$ and $\overline{q_0} = q_0' + \mu m + \eta$, where $\mu = \frac{\beta^2}{(1+\beta) N}$ and $\eta = \frac{\beta}{(1+\beta) N}$, so both spammers benefit by linking together with this configuration.

What to submit

Page 2: For each of (i), (ii), and (iii), ‘yes’ or ‘no’ and an explanation of why or why not.
Figure 7: Another way of linking two spam farms.

Page 3: A mathematical expression for the set in terms of $V$.

Page 4: An expression for $p_0$ in terms of $k, \beta$, and $N$. Also show how you derived the expression.

Page 5:  
- Expressions for $p'_0 + q'_0$ and $\overline{p}_0 + \overline{q}_0$ in terms of $k, m, \beta$ and $N$. Show how you derived these expressions.
- A yes/no answer to whether spammers are better off and a brief explanation.

Page 6:  
- Expressions for $p'_0 + q'_0$ and $\overline{p}_0 + \overline{q}_0$ in terms of $k, m, \beta$ and $N$. Show how you derived the expressions.
- A yes/no answer to whether spammers are better off and a brief explanation.