CS224w:
Social and Information Network Analysis

Assignment Submission Fill in and include this cover sheet with each of your assignments. Assignments are due at 9:00 am. All students (SCPD and non-SCPD) must submit their homeworks via GradeScope (`http://www.gradescope.com`). Students can typeset or scan their homeworks. Make sure that you answer each question on a separate page. That is, one answer per page regardless of the answer length. Students also need to upload their code at `http://snap.stanford.edu/submit`. Put all the code for a single question into a single file and upload it. Please do not put any code in your GradeScope submissions.

Late Day Policy Each student will have a total of *two* free late periods. *One late period expires at the start of each class.* (Homeworks are usually due on Thursdays, which means the first late periods expires on the following Tuesday at 9:00am.) Once these late periods are exhausted, any assignments turned in late will be penalized 50% per late period. However, no assignment will be accepted more than *one* late period after its due date.

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 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: Devon Zuegel
Email: devonz@stanford.edu **SUID:** 005798092

Discussion Group: _____

I acknowledge and accept the Honor Code.

(Signed) Devon K Zuegel

Devon Zuegel // 21 Oct 2015 // CS 224W

Problem Set 2

Spoke with John Luttig and Ilan Goodman while working on this pset.

Problem 1 :: Broadcasting the Network (25 points)

Part A

Undirected graph $G(V,E)$
Adjacency matrix $A \in \{0,1\}^{n \times n}$

For each round k:
- each vertex sends a msg to each of its neighbors at independent times within the interval $[k, k+1)$
  - For each edge (i,j), $T_{ij} \neq T_{ji}$ are the times $i \neq j$ send messages to $j \neq i$ respectively
  - That times are uniformly distributed across the interval
- At any time $t \in [k, k+1)$ we can bug ≥ 1 node, intercepting any incoming msgs.
  - t_i = total time we bugged node i during this round
  - $t_i \beta$ = prob. of us getting caught by node i

Objective: specify $t = [t_1, \ldots, t_n]$ during a single round to max. the expected # of msgs, while ensuring the prob. that we get caught is $< \gamma < 1$

(a) $m_i(t_i) = E(\#$ msgs intercepted from node i when monitoring for $t_i)$

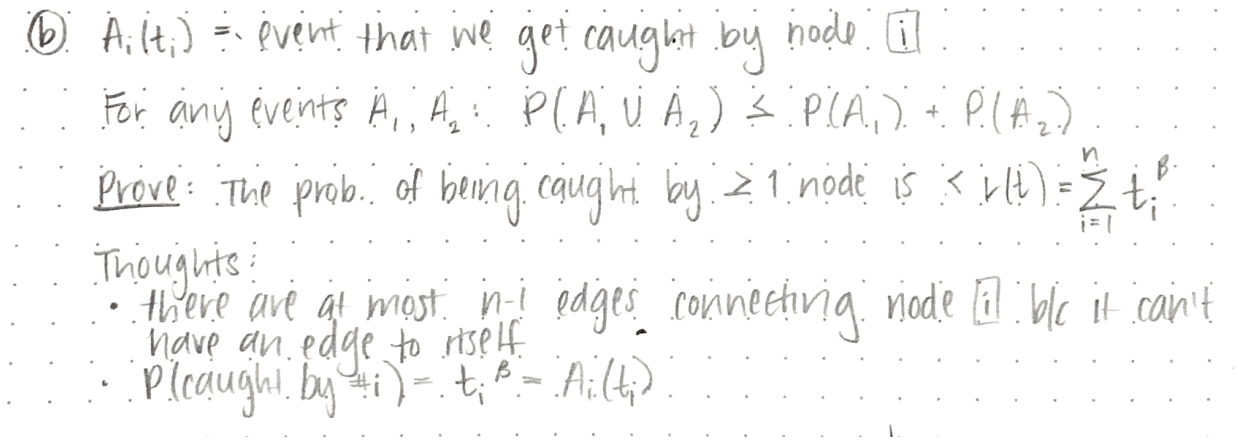
$A[i]$ = adjacency matrix for node i

For a given "friend" j, the prob. that we'll intercept a msg sent from $j \to i$ (lets call this statement A) is:

$\Pr(A) = A[i][j] \cdot t_i$ ← indicator var; ← % of round bugged

$m_i(t_i) = \sum_{j \in n} A[i][j] \cdot t_i$

Part B



For any events A_1, A_2:

$$P(A_1 \cup A_2) \leq P(A_1) + P(A_2)$$

The probability of the event that we get caught by node `i`:

$$P(A_i(t_i)) = t_i^{\beta}$$

Let `P(A)` be the probability that we get caught by one or more nodes:

$$\begin{aligned} A &= A_1(t_1) \cup A_2(t_2) \cup \ldots \cup A_2(t_2) \\ P(A) &\leq P(A_1) + P(A_2) + \ldots + P(A_2) \\ &\leq \sum_{i=1}^{n} P(A_i) = \sum t_i^{\beta} \quad \blacksquare \end{aligned}$$

Part C

Cauchy-Schwarz Inequality

$$\left(\sum_i a_i b_i\right)^2 \le \left(\sum_i a_i^2\right)\left(\sum_i b_i^2\right) \rightarrow (a \cdot b)^2 \le (a \cdot a)^2 (b \cdot b)^2$$

Goal: Find a strategy $t = [t_1, t_2, \ldots, t_n]$ that maximizes the total # of intercepted msgs.

Constraints: $\beta = 2, \quad \gamma < 1, \quad r(t) \le \gamma$

$$M = \sum_i m_i(t_i) = \sum_i \sum_j t_i \cdot A(i,j) = \underbrace{\sum_i \left(t_i \sum_j A(i,j)\right)}_{\text{matrix } 1 \times n} = \sum_i a_i b_i$$

$$r(t) = \sum_i t_i^2 \le \gamma$$

$$\begin{bmatrix} a_i = t_i \\ b_i = \sum_j A(i,j) = A_i \end{bmatrix}$$

$$M = \left[\left(\sum_i a_i b_i\right)^2\right]^{1/2}$$

$$\le \left[\left(\sum_i a_i^2\right)\left(\sum_i b_i^2\right)\right]^{1/2}$$

$$\le \left[\underbrace{\left(\sum_i t_i^2\right)}_{r(t) \le \gamma}\left(\sum_i A_i^2\right)\right]^{1/2} \le \left[\gamma\left(\sum_i A_i^2\right)\right]^{1/2}$$

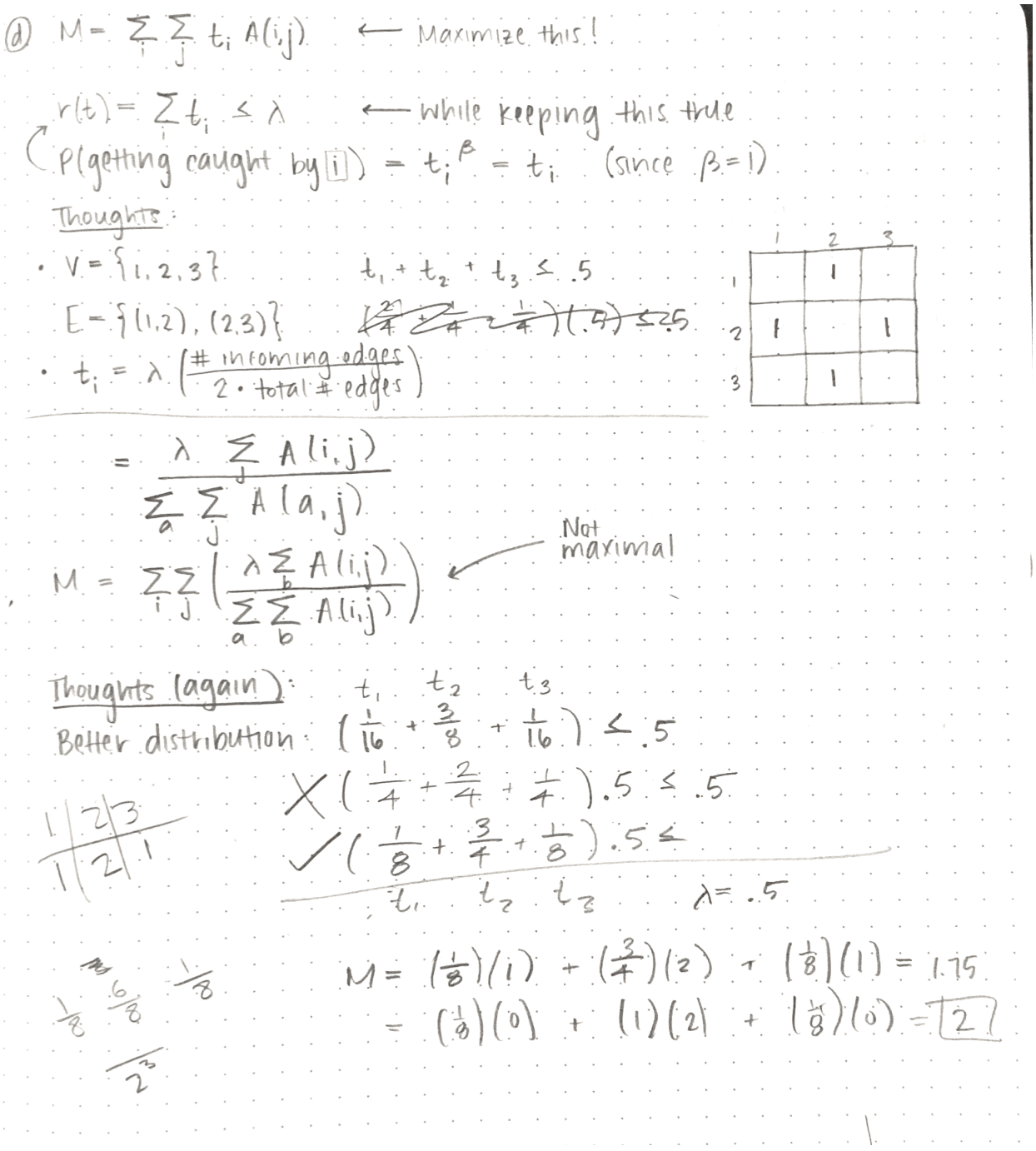
$$\boxed{M = \sqrt{\gamma \sum_i A_i^2}}$$

Part D

NOTE: It is not possible to have an undirected graph without self-loops with distinct degrees. However, for the sake of solving this problem we should maintain this assumption until the end.

Thoughts

- Constraints:
  - $0 \leq t_i \leq 1$ since the intervals are 1 unit of time long
  - $\sum_i t_i \leq \gamma < 1$
- Considering the case where all degrees are the same, we would want to bug each node for the same amount of time.
- If you have three companies that have a certain rates `(r1,r2,r3)` of return on your investment (all of them have zero risk), how would you invest your money?
  - You would invest all of your money into the company with the highest return rate.
  - If you had a constraint that said you could invest at most half of your money in any given company, you'd put half into the company with the best return and half into the one with the second-best return.



Answer

In the case where all node degrees are distinct, there exists a single node `n` whose degree `d` is greater than that of all other nodes in the network. Now let's consider some tiny amount of time `x`, which we can use to bug incoming messages of a single node in the network. We'd get most "bang for our buck" if we spent that `x` time at `n`, because it has `d` potential incoming messages at any given time. Thus, we want to spend every unit of time possible at that highest degree node. We can spend at most `1` unit at any given node since the interval is of length `1`, but since `\gamma < 1`, we don't have to worry about distributing it according to that original constraint since we don't have `> 1` unit of time to spend.

Problem 2 :: Signed Triad Analysis (15 points)

Part A

```
{-6: 58622, 2: 450636, -2: 395357, 6: 3978926}
```

- **3 negative edges:** 58,622 triads (01.2%)
- **3 positive edges:** 3,978,926 triads (81.5%)
- **2 postive, 1 negative:** 450,636 triads (09.2%)
- **2 negative, 1 positive:** 395,357 triads (08.1%)

Part B

Calculated in `hw2p2a.py`:

```
number of positive edges = P = 592,551
number of negative edges = N = 119,232
total number of edges    = T = 711,783
p = P / T = 592,551 / 711,783 = .832    # Fraction of positive
```

Probability of…

- **3 negative edges:** `3(1 - p)^3 = .47%`
- **3 positive edges:** `3p^3 = 57.69%`
- **2 postive, 1 negative:** `3(1 - p)p^3 = 34.83%`
- **2 negative, 1 positive:** `3p(1 - p)^3 = 07.01%`

Part C

The rankings for most probable to least probable triad type are consistent in our expected and actual results. However, there are two critical differences:

- We expected to see all-positive triads < 60% of the time, but in reality we saw them more than 80% of the time!
- We expected to see 2 positive, 1 negative triads > 30% of the time, but in reality we only saw it 9% of the time!

One hypothesis for this shift is the idea in psychology called “The Norm of Reciprocity”, which is “the expectation that people will respond favorably to each other by returning benefits for benefits” (Wikipedia). Another hypothesis is that there’s a social cost associated with negatively reviewing someone, so people tend to do it less frequently in tighter-knit social circles (a.k.a. those that tend to create a disproportionate amount of the triads).

Problem 3 :: Decentralized Search (40 points)

Part A

Setup:

- `T` = complete, perfectly-balanced `b`-nary tree
  - each node has `b` children
  - `b ≥ 2`
- `N` = # of network nodes
- `h(T)` = height of `T`

Thoughts:

- number of leaves grows by a factor of `b` each time the height increases
- the number of leaves on a tree `N` of height `h` is b^h

$$N = b^{h(T)}$$
$$h(T) = \log_b N$$

Part B

The maximum possible value of `h(v, w)` occurs when the ancestor common to `v` and `w` is the root of the original tree `T`. Therefore, this max value is the height of `T`, which we showed in part (a) to be $h(v, w) = \boxed{log_b N}$

Part C

Say that `T_{v, d}` is our subtree that contains the desired height `d` and node `v`. `T_{v,d} - T_{v,d-1}` satisfies `h(v, w) = d`, and the number of leaves of a tree of height `h(T)` is $b^{h(T)}$. Thus, there are $b^d - b^{d-1}$ nodes satisfying h(v, w) = d.

Part D

Goal:
Prove the following:

$$Z = \sum_{w \neq v} b^{-h(h,w)} \leq log_b N$$

Thoughts:

- $$p_v(w) = \frac{1}{Z} b^{-h(h,w)} \rightarrow Z = \frac{1}{p_v(w)} \cdot b^{-h(h,w)}$$
- In part c we figured out that there are `b^i - b^(i - 1)` nodes `w` such that `h(v, w) = i`.

Answer:

$Z = \sum_{w \neq v} b^{-h(h,w)}$

$= \sum_{w \neq v} b^{-i}$

We can rearrange this and look at each level of the tree as our summation instead of looking at every `w ≠ v`. It is equivalent to consider each level `i` from `1` to $log_b N$ if we multiple each step by the number of "`w`s" exist at each level (which we figured out in part c):

$= \sum_{i=1}^{log_b N} (b^i - b^{i-1}) b^{-i}$

$= \sum_{i=1}^{log_b N} (1 - \frac{1}{b})$

$\leq log_b N$

Part E

Setup:

- Let `d = h(v, t)`.
- For any of the b^{d-1} leaves `u` in `T'`, the following equality holds: `h(v, u) = d`.
- `N(T)` = # of leaves in T'
- `prob` = prob of any 2 nodes connecting

The probability of `v` linking an edge to `u` is $\frac{b^{-d}}{Z}$. Therefore the probability of `v` linking an edge to `T'` is:

$$\begin{aligned} N(T) \cdot \text{prob} &= \frac{b^{d-1} \cdot b^{-d}}{Z} \\ &= \frac{1}{bZ} \\ &= \frac{1}{b \sum_{w \neq v} b^{-h(h,w)}} \end{aligned}$$

And we know that $Z \leq log_b N$, so we get the following:

$$\geq \frac{1}{b \log_b N}$$

Part F

The probability of `v` having zero edges into `T'` is less than the value of the following expression:

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

$$= (1 - \frac{1}{b \log_b N})^{b(\log_b N) \frac{c}{b} (\log_b N)}$$

$$= (\frac{1}{e})^{\frac{c \ln N}{b \ln b}}$$

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

Part G

Starting at any given node `v` we can find some some node `u` with `h(u, t) < h(v, t)`. We know already that `h(v, t) < log_b N`, so we also know that we can reach the destination in $O(\log_b N)$ steps.

Part H

My plot shows that the average search time decreases as alpha grows beyond one (there's a little hump where $\alpha = 1$).

In other words, successful searches tend to finish in fewer steps. The reasoning is that there are fewer ways for things to go wrong, plus the longest possible search is limited by the height of the total tree.

Problem 4 :: Variations on a Theme of PageRank (25 points)

4 Personalized Page Rank

(a) Known personalized PageRank Vectors:

$A = \{1, 2, 3\}$ $B = \{3, 4, 5\}$ $C = \{1, 4, 5\}$ $D = \{1\}$

Now can we represent...

i) $E = \{2\}$? ⟶ Yes, $E = 3A - 3B + 3C - 2D$

ii) $F = \{5\}$? ⟶ No, b/c we can't get 5 by itself.

iii) $G = \{1, 2, 3, 4, 5\}$ ⟶ Yes, $G = 6A + 3C + 3B - 2D$

(b) Given V, we can compute any linear combo. of the vectors in it.

(c) $p_0 = \beta\lambda + \beta \sum_i^k p_i + \frac{1-\beta}{n}$ $\quad p_i = \frac{\beta p_0}{k} + \frac{1-\beta}{n}$

$= \beta\lambda + \beta \sum_i^k \left(\frac{\beta p_0}{k} + \frac{1-\beta}{n}\right) + \frac{1-\beta}{n}$

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

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

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

(d) $p_0' = \frac{\beta\lambda}{1-\beta^2} + \frac{\beta k + 1}{n(1+\beta)}$ $\quad q_0' = \frac{\beta\lambda}{1-\beta^2} + \frac{\beta m + 1}{n(1+\beta)}$ ⟶ Both p_0' & q_0' have this exact adder, so we can ignore it.

$\bar{p}_0 = \frac{1-\beta}{n} + \frac{\beta}{2} \sum_i^{k+m} \left(\frac{1-\beta}{n} + \frac{2\beta \bar{p}_0}{k+m}\right)$

$= \beta^2 \bar{p}_0 + \left(\frac{1-\beta}{n}\right)\left[1 + \frac{\beta}{2}(k+m)\right]$

$\frac{\bar{p}_0(1-\beta^2)}{(1+\beta)(1-\beta)} = \frac{\left(\frac{1-\beta}{n}\right)\left[\frac{2}{2} + \frac{\beta}{2}(k+m)\right]}{(1+\beta)(1-\beta)}$

$\boxed{\bar{p}_0 = \frac{2 + \beta(k+m)}{2n(1+\beta)} = \bar{q}_0}$

$p_0' + q_0' = \frac{\beta(k+m) + 2}{n(1+\beta)}$

$\bar{q}_0 + \bar{p}_0 = p_0' + q_0'$

So linking doesn't help them (but doesn't hurt, either)

(e) Again we can ignore the λ term since it's the same:

$$p_0' = \frac{\beta k + 1}{n(1+\beta)} + \dots \qquad q_0' = \frac{\beta m + 1}{n(1+\beta)} + \dots$$

$$\overline{q_0} = \beta\left[\overline{p_0} + \sum_1^m q_i\right] + \left(\frac{1-\beta}{n}\right)$$

$$= (\beta k + 1)\left(\frac{1-\beta}{n}\right) + \beta\overline{p_0}$$

$$\overline{p_0} = (\beta k + 1)\left(\frac{1-\beta}{n}\right) + \beta\overline{q_0}$$ → Basically the same

$$= (\beta k + 1)\left(\frac{1-\beta}{n}\right) + \beta\left[(\beta k + 1)\left(\frac{1-\beta}{n}\right) + \beta\overline{p_0}\right]$$

$$= \text{——————} + \beta^2\overline{p_0} + \beta(\beta k + 1)\left(\frac{1-\beta}{n}\right)$$

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

And symmetrically...

$$\overline{q_0} = \frac{\beta m + \beta^2 k}{(1+\beta)n} + \frac{1}{n}$$

$$\overline{p_0} + \overline{q_0} > p_0' + q_0'$$

↳ So it's beneficial to the spammers! BOTH of them!