CMS/CS/EE/IDS 144 – Problem Set 1

Marco Yang

1 Coding + Data Analysis

1.1 Before you can walk you must…

A web crawler is a program that automatically traverses and collects the web pages on the Internet. Crawlers are widely used by search engines to find and retrieve what's on the web so that they can build an index of the pages for searching. However, they're also useful for many other reasons, e.g., for gathering data, for validating the links on a web page, or for gathering everything from a particular site. You are going to implement a very primitive web crawler to explore the web pages in the Caltech domain (URLs containing ".caltech.edu") and count the number of hyperlinks each page contains.

The basic idea of crawling: A web crawler starts by retrieving a web page, parsing the HTML source and extracting all hyperlinks in it. Then, the crawler uses these hyperlinks to retrieve those pages and extracts the hyperlinks in them. This process repeats until the crawler has followed all links and downloaded all pages. Therefore, a crawler requests web pages just as a web browser does, but it browses automatically, following all the hyperlinks in each page. Although the idea is simple, there are a few issues that come up:

1. Many web pages contain links to multimedia/data files, the crawler should not waste bandwidth downloading these files.
2. Many web sites deliver dynamic content; the web page is generated dynamically based on a query string specified in the URL. A crawler can get trapped on such a site, since there are potentially 'infinitely many' pages. For example, on an online calendar, the crawler can keep following the links of dates/months and get trapped in an endless loop.
3. Given that the crawler has extracted a long queue of links to visit, which one should be selected to visit next?
4. Given that the crawler has seen a page already, when should it go back to revisit the page and check for new and changed links? (You can ignore this for this assignment.)

There are many other issues too, such as the robot exclusion standard, Javascript in web pages, ill-formed HTML and so on.

Your task: Your task is to write a crawler that, given a URL in the Caltech domain, retrieves the web page, extracts all hyperlinks in it, counts the number of hyperlinks, follows the extracted hyperlinks to retrieve more pages in Caltech domain, and then repeats the process for each successive page. During this process you must keep updating the number of hyperlinks on a web page and the number of hyperlinks which point to that page.

Important notes:

- Ensure that your crawler stays within the Caltech domain. Do not crawl library linked services (e.g. JSTOR, IEEE Xplore, Wiley Online Library, etc.) or systematically download academic journal articles.
- An efficient crawler can easily be mistaken for a malicious denial of service (DoS) attack. **Do not do DoS attacks on the Caltech servers!** You should limit the number of parallel requests and/or the time between requests.

What you turn in: You should submit the following after crawling at least 2000 HTML pages in the Caltech domain starting from www.caltech.edu.

1. Your code, including libraries used and how to run it.

Solution: https://mcrco/cs144/tree/main/hw1

Instructions to run in readme.

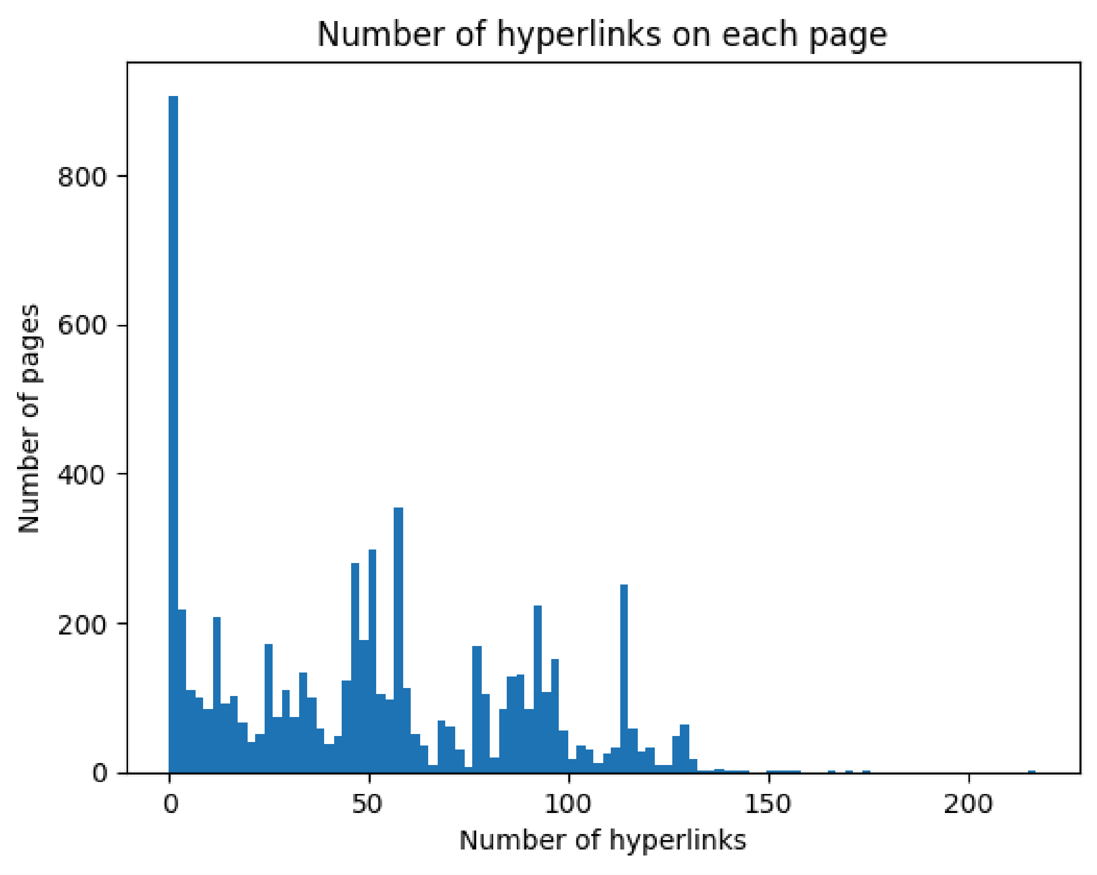
Below graphs and metrics are generated using `p1.ipynb`.

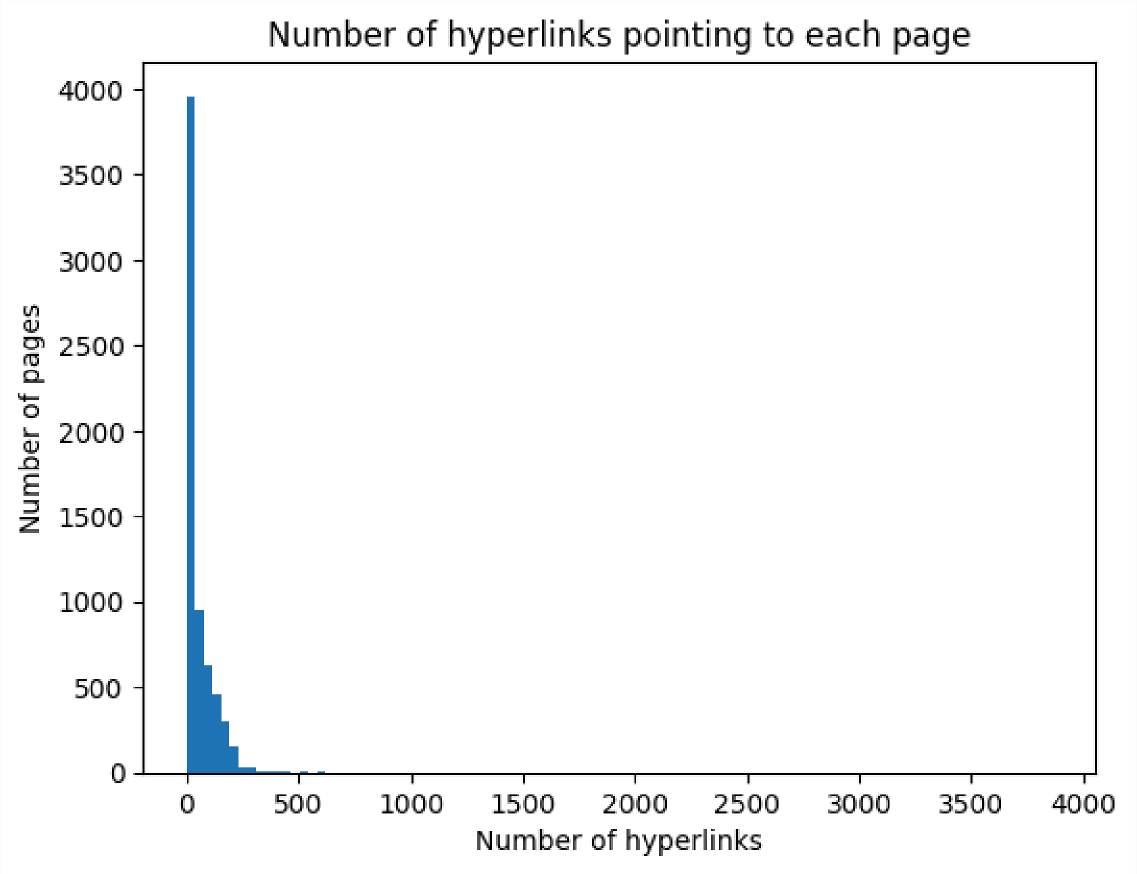
2. An explanation of the selection policy you chose and its strengths and weaknesses.

Solution: I chose BFS (with concurrency, so not really BFS since depth is not guaranteed to be monotonically non-decreasing). I feel like BFS makes more sense because pages closer to the root page (caltech.edu) are probably more relevant. It makes more sense to crawl the more relevant pages (like a Caltech major description page) rather than recurse all the way own to some super niche lab page.

3. Two histograms. One for the number of hyperlinks per page. One for the number of hyperlinks which point to each page.

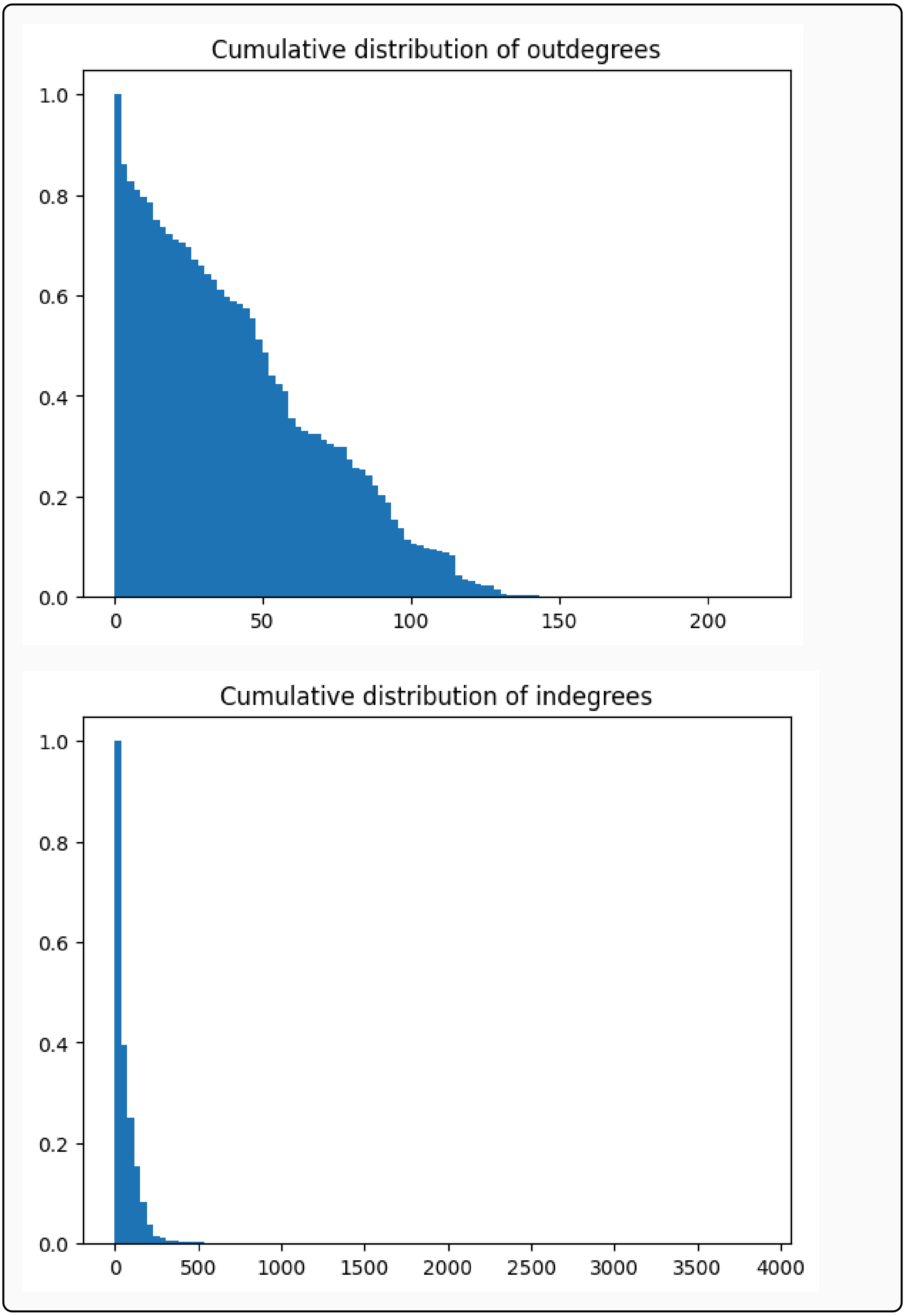
Solution:





4. Two complementary cumulative distribution functions (ccdf). One for the number of hyperlinks per page. One for the number of hyperlinks which point to each page.

Solution:



5. The average clustering coefficient and the overall clustering coefficient of the graph. Treat the edges as undirected for these calculations.

Solution: Average clustering coefficient: 0.7750951022701442 , Overall clustering coefficient: 0.4247585079850745

6. The average and maximal diameter of the graph. Treat the edges as undirected for these calculations.

Solution: Average diameter: 2.6395950762567555, Maximal diameter: 6

1.2 Six degrees of separation

In most real-life network graphs, we find that we can reach from almost any node to any other node within a very small number of hops (often six). This is known as **six degrees of separation**. In this problem, you explore six degrees of separation in a few other networks. Please choose three of the four parts of this problem to do.

1. **Flight to the Queen Bee** In the music-map graph, find paths between:
   - Nikolai Rimsky-Korsakov → Beyoncé

   Solution:
   i. Nikolai Rimsky-Korsakov → Dybbukk → Billie Eillish → Beyonce Knowles
   ii. Above is also the shortest I think.

   - Lyle Mays → Taylor Swift

   Solution:
   i. Lyle Mays → Globe Unity Orchestra → Vienna Art Orchestra → Syndicate → Weather Report → Genesis → Pink Floyd → Tame Impala → Daniel Caesar → The Weeknd → Ariana → Taylor Swift
   ii. Can't find anything else lol

2. **Know your professors** In the co-authorship network, find paths between:
   - Eric Mazumdar → Rudolf E. Kálmán

   Solution:
   i. Eric Mazumdar → Shankar S. Sastry → Brian David Outram Anderson → Rudolph E. Kalman via "On gradient-based learning in continuous games", "Nonlinear averaging theorems, and the determination of parameter convergence rates in adaptive control.", "Equivalence of linear time-invariant dynamical systems."
   ii. I trust the MathSciNet algorithm

   - Georgia Gkioxari → Paul Erdős

   Solution:
   i. Georgia Gkioxari → Pietro Perona → Giorgio Picci → Gyorgy Michaletzky → Laszlo Gerencser → Paul Erdos via "System-theoretic aspects of dynamic vision", "Zeros of spectral factors, the geometry of splitting subspaces, and the algebraic Riccati inequality",

“Stability of block-triangular stationary random matrices”, “Problems of graph theory concerning optimal design”
ii. I trust the MathSciNet algorithm

Note: I used the tool on MathSciNet.

3. **YouTube** Find a path between the given videos on www.youtube.com in ‘Incognito mode’:
   - Adam’s Watson Lecture → Never Gonna Give You Up
   - Cassie Goes For A Walk → What Does the Fox Say

   Describe why it might be easier to get “stuck” when clicking from video to video on YouTube compared to other networks.
4. **Citations** Find paths between seminal papers in different fields:
   - Zipf (1936): “The psycho-biology of language” → Granovetter (1973): “The Strength of Weak Ties”

Solution:
i.
ii. It’s been two hours I give up.

- Kalman (1960): “A New Approach to Linear Filtering and Prediction Problems” → LeCun et al. (1998): “Gradient-Based Learning Applied to Document Recognition”

Solution:
i. “A New Approach to Linear Filtering and Prediction Problems” → “Deep learning in neural networks: An overview”“ → “Gradient-Based Learning Applied to Document Recognition”
ii. Above is probably the shortest?

2 Theory

2.1 Random Bids

Each of the N bidders (including you) choose a bid value following the continuous uniform distribution between 0 and v, $v > 0$. The bid for each participant is i.i.d. The winner is the bidder with the maximum value.

(i) What is the probability that you win the auction?

Solution: $\frac{1}{N}$ since everyone has an equal chance of winning.

(ii) What is the expected number of times that the worker would need to update the highest bid as they go through the N envelopes?

Solution: For each envelope, the worker needs to update the highest bid if the current envelope is the highest bid so far. Since the probability that the last envelope out of i processed envelopes so far has the highest bid is $\frac{1}{i}$, the expected number of times the worker would need to update the highest bid is

$$\sum_{i=1}^{N} \frac{1}{i}.$$

(iii) Does your answers to the previous 2 parts change if the distribution that the bidders sampled from were not uniform, but still i.i.d. and continuous for all bidders?

Solution: No, it doesn't since the chance of each person winning is still $\frac{1}{N}$.

(iv) What is the expected amount of money that you will pay for participating in the auction? If you lose, you pay 0; if you win, you pay the amount bid. How much does your dream car have to be worth for your expected payoff to be non-negative?

Solution: The expected money I pay is

$$\begin{aligned}\int_0^v p(\text{win}|x) \cdot x \,\mathrm{d}x &= \int_0^v \left(\frac{x}{v}\right)^{N-1} x \,\mathrm{d}x \\ &= \frac{1}{v^{N-1}(N+1)} \left[x^{N+1}\right]_0^v \\ &= \frac{v^2}{N+1}.\end{aligned}$$

Since I'm expected to pay the above amount, I would be satisfied as long as my dream car was worth more than that.

2.2 The Friendship Paradox

Consider an undirected graph $G = (V, E)$ with $V = \{1, ..., n\}$. Let μ be the average degree of a node and σ^2 be the variance of the degree of a node.

(i) Prove the classic statement of the friendship paradox by showing that the average degree of a neighbor is $\mu + \frac{\sigma^2}{\mu}$. Show that when the variance is nonzero, the average degree of a neighbor is larger than the average degree of a node.

Solution: The average degree of a neighbor is the expected value of a neighbor that we choose arbitrarily. To randomly choose a neighbor v, we must choose one of its neighbors $u \in N(v)$, and then "traverse" the edge to v to get to the neighbor u. Notice that the number of ways we could do this is just the sum of all the degrees of every node (each degree corresponds to an incoming edge). Thus, the probability of choosing a neighbor v out of all the possible neighbors is

$$P(v) = \frac{d(v)}{\sum_{v \in V} d(v)}.$$

The expected value of the degree of a neighbor is

$$\begin{aligned}E[d_N] &= \sum_{v\in V} P(v)d(v)\\ &= \sum_{v\in V} \frac{d(v)}{\sum_{v'\in V} d(v')} \cdot d(v)\\ &= \frac{\sum_{v\in V} d(v)^2}{\sum_{v\in V} d(v)}.\end{aligned}$$

The definition of variance is

$$\sigma^2 = E[X^2] - E[X]^2 \Rightarrow \frac{1}{n}\sum_{v\in V} d(v)^2 - \mu^2 \Rightarrow \frac{1}{n}\sum_{v\in V} d(v)^2 = \mu^2 + \sigma^2.$$

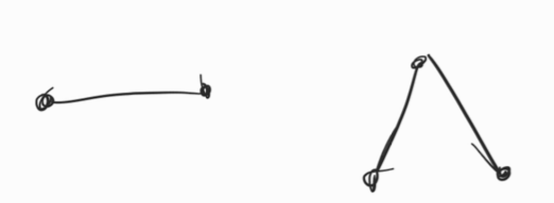
Multiplying our expression for $E[d_N]$ by $1 = \frac{n}{n}$,

$$\begin{aligned}E[d_N] &= \left(\frac{1}{n}\sum_{v\in V} d(v)^2\right)\left(\frac{n}{\sum_{v\in V} d(v)}\right)\\ &= \frac{\mu^2+\sigma^2}{\mu}\\ &= \mu + \frac{\sigma^2}{\mu}.\end{aligned}$$

Thus, when the mean and variance are non-zero, the expected degree of a neighbor is greater than the expected degree of any node selected at random.

(ii) Give an example of a graph where the variance of degrees is non-zero, but every node has the same degree as its neighbors.

Solution:



A disconnected graph consisting of a loop and a fully connected subgraph with 4 vertices.

2.3 Disconcerting Diameter Definitions

Construct an unweighted graph where the maximal diameter is more than 2 times the average distance between nodes.

Solution:

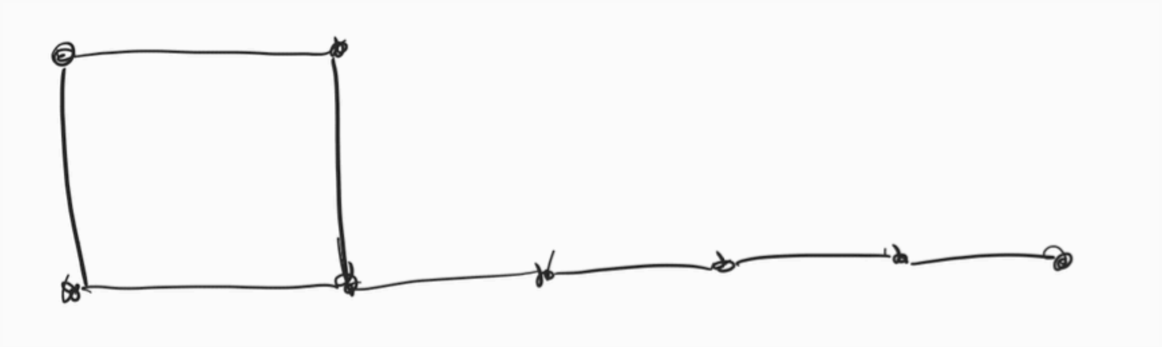


Figure 1: K_4 connected to L_5

Consider complete graph K_4, with a line extending from one of the vertices that forms a line graph L_5 with 5 nodes total (including the node in K_4). The maximal diameter of this graph is 5 (each of the diameters from the 3 nodes in K_4 that aren't a part of the line graph have a distance of 5 from the node at the end of the line graph). The average distance $\bar{l}$ is computed as follows:

$$S = \underbrace{\binom{4}{2} \cdot 1}_{\text{paths within } K_4} + \underbrace{4 \cdot 1 + 3 \cdot 2 + 2 \cdot 3 + 1 \cdot 4}_{\text{paths within } L_5} + \underbrace{3 \cdot (2 + 3 + 4 + 5)}_{\text{paths between } L_5 \text{ and } K_4}$$

$$\bar{l} = \frac{S}{\binom{8}{2}} = \frac{68}{28} = 2.43 < 2.5 = \frac{5}{2}.$$

2.4 Understanding Clustering

We define the average clustering coefficient $\mathrm{Cl}^{\mathrm{avg}}(G)$ and the overall clustering coefficient $\mathrm{Cl}(G)$ as:

$$\mathrm{Cl}^{\mathrm{avg}}(G) := \frac{\sum_{i=1}^{n} \mathrm{Cl}_i(G)}{n}$$

where

$$\mathrm{Cl}_i(G) := \frac{\text{number of triangles centered on vertex } \ i}{\text{number of triples centered on vertex } \ i}$$

$$\mathrm{Cl}(G) := \frac{3 \times \text{number of triangles in the graph}}{\text{number of connected triples of vertices}}$$

Your task: Construct an example where, as the number of nodes in the graph becomes large, $\mathrm{Cl}^{\mathrm{avg}}(G_n) \to 1$ and $\mathrm{Cl}(G_n) \to 0$. Justify your answer.

Solution:

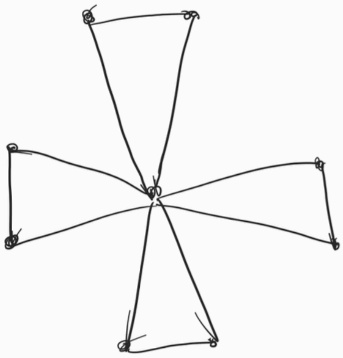


Figure 2: 4 connected triangles. The graph we propose is made up of $n \to \infty$ triangles

Consider a graph consisting of n triangles that all share a single point. Notice that for a vast majority of the points (the points at the base of each triangle), $\mathrm{Cl}_i(G) = 1$ since there is only one triple centered at each of those points. For the central shared point, $\mathrm{Cl}_i(G) < 1$, but that doesn't matter since the average clustering coefficient is averaged over $2n + 1$ points, so even if the central point's clustering coefficient was 0, as $n \to \infty$,

$$\mathrm{Cl}^{\mathrm{avg}}(G) = \frac{2n}{2n+1} \to 1.$$

Now, the total number of triples is much higher since the number of triples is quadratic w.r.t the number of triangles: each of the non-central points form a triple with each of the other non-central points for a total of $2n \cdot (2n-1) \cdot \frac{1}{2} = n \cdot (2n-1)$ triples. However, the total number of triangles is linear, so the global clustering coefficient is

$$\mathrm{Cl}(G) = \lim_{n\to\infty} \frac{3n}{n(2n-1)} \to 0.$$