4.1 Directed Graphs

Until today, we were only considering graphs with symmetric relations in the edges. Now, we're considering **directed graphs** or **digraphs**, where the edges have a defined directionality. The vertex where an edge starts is the **tail** and the vertex that is pointed to is the **head**. These together are the **endpoints**. We also term the tail as the **predecessor** of the head and the head as the **successor** of the tail. We can easily create a directed graph from an undirected graph by *orienting* each edge. An **orientation** of an undirected graph involves the selection of a direction for each edge, to create a directed graph.

Like with our undirected graphs. We can consider digraphs as **simple digraphs** if they don't have repeated edges or loops. Note that a simple digraph can have two edges between the same two vertices as long as they point in opposite directions. *Loopy digraphs* contain loops and *multi-digraphs* can contain multiple edges of the same directionality between the same two vertices.

We have similar definitions in directed graphs for **walks**, **paths**, **trails**, and **cycles**. Likewise, we have the same concepts of **subgraphs** and **isomorphism**. The **adjacency** matrix is created in a similar row-wise fashion, where a nonzero in position (x, y) indicates one or more edges pointing from vertex x to vertex y except now it is now longer symmetric.

Instead of just degree, digraphs consider both **out degree** ($d^+(v)$) or **in degree** ($d^-(v)$). We also have the out neighborhood ($N^+(v)$) or successor set and the in neighborhood ($N^-(v)$) or predecessor set. We'll use the concept of directed graphs when discussing the PageRank algorithm.

4.2 PageRank

"A modern classic." – Professor Gittens

We're going to talk a bit about the PageRank algorithm. This algorithm serves as a good example of various ways in which to consider computations on graphs, and it has many varied applications. One application of PageRank is to compute some metric of *centrality* (i.e., importance) for all vertices in a directed graph.

4.2.1 Random Walk Model

A **random walk** on a graph is a walk that starts at some v and randomly selects some $u \in N(v)$ to hop to from v. Then, some $w \in N(u)$ is selected and hopped to. This iterates for some number of steps. Random walks are a surprisingly powerful tool, and

they form the basis for a number of approximation and sampling-based algorithms; doing multiple random walks on a small portion of a large graph is a good way to measure some properties that might be extrapolated to the full graph.

In the first PageRank model, we're considering our graph to be a series of web pages connected through directed hyperlinks. We have a hypothetical **surfer** who randomly clicks on links while browsing. The PageRank of a webpage is essentially the probability that this surfer will be viewing that given page at any point in time. We can therefore calculate the PageRank of a page simply by performing a random walk and tracking how often the page is visited versus the total number of page visits.

Issues arise for any page that doesn't have any outgoing links (out degree is zero, called a **sink**) or if the graph isn't strongly connected. So there's two additional considerations. Whenever a random surfer reaches one of these pages, they randomly jump to any other page in the graph. Its common to additionally consider that this surfer will randomly jump with some probability whenever they reach any page; this introduces a form of stability when calculating PageRanks, particularly when the graph isn't strongly connected. The probability that a surfer clicks a link versus randomly jumping is termed as the **damping factor**. For now, we'll just consider the basic model without the damping factor component.

4.2.2 Graph Algorithm Model

---

for all $v \in V(G)$ **do**
  $P(v) \leftarrow \frac{1}{n}$ ▷ Initialize PR equally among vertices
  if $|N^+(v)| = 0$ **then**
    $sink \leftarrow sink + P(v)$ ▷ Total PR of all sinks
for some number of iterations **do**
  $sink_n \leftarrow 0$ ▷ Sink contribution on next iteration
  for all $v \in V(G)$ **do**
    $P(v) \leftarrow \frac{sink}{n}$ ▷ First get an equal portion from the sinks
    for all $u \in N^-(v)$ **do**
      $P(v) \leftarrow P(v) + \frac{P(u)}{|N^+(u)|}$ ▷ All u in $N^-(v)$ shares $P(u)$ with $N^+(u)$
    if $|N^+(v)| = 0$ **then**
      $sink_n \leftarrow sink_n + P(v)$ ▷ Sink contribution for next iteration
  swap$(sink, sink_n)$

Graph algorithmic computations can be typified as performing some kind of per-vertex or per-edge iterative update computation on some per-vertex or per-edge *state*. From the perspective of each vertex during a PageRank computation, we have some initial PageRank value that gets updated through multiple iterations. Going back to the random surfer model, a surfer arrived on that vertex/page either from an in edge or it was the

destination of a random jump. As the surfer travels, we can consider that it takes with it a portion of the prior vertex's PageRank. So for a given vertex, the PageRank is the sum of PageRanks incoming through in edges plus the PageRanks from potential random jumps and from zero out degree vertices.

The algorithm above gives how we'd calculate PageRank using this model.

4.2.3 Linear Algebraic Model

We can also use the adjacency matrix of the graph to formulate this problem from an algebraic perspective. As the transitions of our random surfer through the graph essentially form a Markov chain, we can create a stochastic matrix M of transition probabilities from a given vertex to its neighbors. We define M as:

$$M = (D^{-1}A)^T$$

where D is a diagonal matrix of out degrees and A is the adjacency matrix. We can use a modified adjacency matrix where each vertex with zero out degree is changed to have an out degree of $n-1$ and links to all other vertices in the graph, in order to mirror our prior models. We can then compute updated PageRanks in vector $\mathbf{p}_{i+1}$ as the matrix vector product of the current PageRank values $\mathbf{p}_i$ and the transition probability matrix.

$$\mathbf{p}_{i+1} = M\mathbf{p}_i$$

with steady state solution

$$\mathbf{p}_\infty = M\mathbf{p}_\infty$$

This should look familiar. Recall that an eigenvector $\mathbf{v}$ for some matrix A is defined as

$$A\mathbf{v} = \lambda\mathbf{v}$$

where λ is a corresponding eigenvalue. So, our PageRank steady state solution is actually just an eigenvector of the transition probability matrix with eigenvalue of 1. There's an entire subfield of graph theory devoted to studying the eigenvectors and eigenvalues on matrices derived from graph adjacency matrices. This is called *Spectral Graph Theory*. In this class, this lecture is the deepest we'll go into that field, however.