1 Orientations and Tournaments

An **orientation** of a graph is a digraph obtained by choosing a direction for each of its undirected edges. An **oriented graph** is an orientation of a simple graph. A **tournament** is an orientation of a complete graph.

A **king** is a vertex that can reach every other vertex in a directed graph in at most two hops.

Prove that every tournament has a king, and correspondingly that every vertex of maximal out degree in a tournament is a king.

Orientations can serve as representative of the outcomes of competitions between individuals/teams, where a winner is designated by the direction of each edge. Given an orientation that represents the outcomes after a season of competition, how might we be able to determine a champion if multiple teams have the same out/in degree? We'll talk about using a PageRank computation to do just that.

2 PageRank

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 has many varied applications. The purpose of PageRank is to compute some metric of *importance* for all vertices in a directed graph.

2.1 Random Walk Model

In the first 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**). So there's two additional considerations. Whenever a random surfer reaches ones of these pages, they randomly jump to any other page in the graph. Additionally, this surfer will randomly jump with some probability whenever they reach any page; this introduces stability when calculating PageRanks using the other methods. The probability that a surfer clicks a link versus randomly jumping is termed as the **damping factor**.

2.2 Graph Algorithm Model

Graph algorithms are typified as performing some kind of per-vertex or per-edge iterative computation. From the perspective of a vertex, we have some initial PageRank value that gets updated through multiple iteration. 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.

We calculate PageRank as follows:

---

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**
  $sink_n$ ▷ Sink contribution on next iteration
  for all $vinV(G)$ **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 equal of $P(u)$ with $N^+(u)$
    $P(v) \leftarrow P(v) \times \Delta$ ▷ Contributions scaled by damping factor
    $P(v) \leftarrow P(v) + \frac{1-\Delta}{n}$ ▷ Now add in contribution of random jumps
    if $|N^+(v)| = 0$ **then**
      $sink_n \leftarrow sink_n + P(v)$ ▷ Sink contribution for next iter
  swap$(sink, sink_n)$

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 model. We can then compute updated PageRanks P_{i+1} as the matrix vector product of the current PageRank values P_i and the transition probability matrix.

$$P_{i+1} = MP_i$$

with steady state solution

$$P_\infty = MP_\infty$$

However, this calculation doesn't include the damping factor. With the damping factor, we have:

$$P_\infty = \Delta MP_\infty + \frac{1-\Delta}{n}\mathbf{1}$$

or just

$$P = (\mathbf{I} - \Delta M)^{-1}\frac{1-\Delta}{n}\mathbf{1}$$