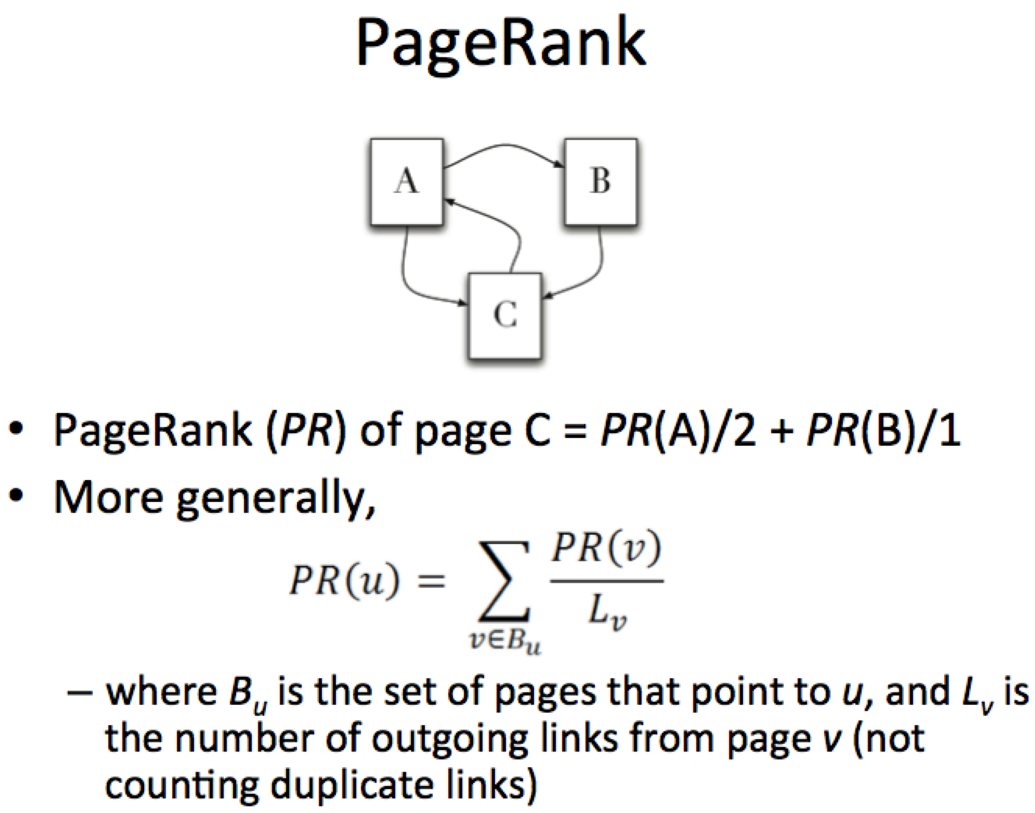
page_rank.py

In this code I have implemented the google's page rank algorithm stated below. Please ignore any mistakes in the usage of python or the code as it was my first attempt at python.



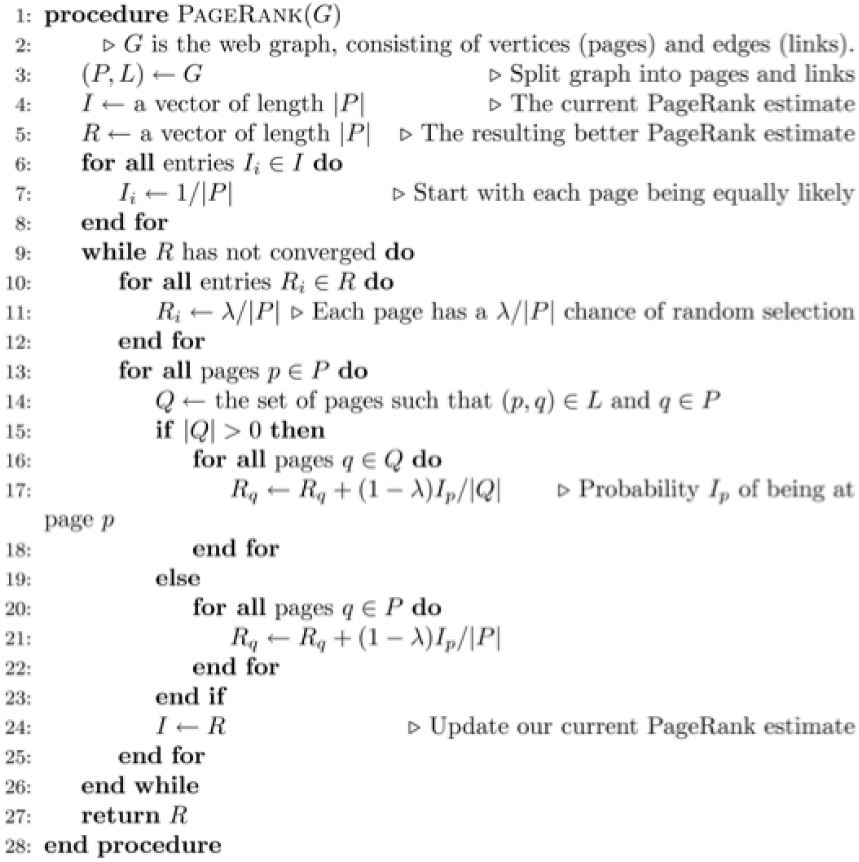
PageRank

- PageRank (*PR*) of page C = *PR*(A)/2 + *PR*(B)/1
- More generally,

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L_v}$$

  – where B_u is the set of pages that point to *u*, and L_v is the number of outgoing links from page *v* (not counting duplicate links)

Algorithm:



```
procedure PageRank($G$)
        ▷ $G$ is the web graph, consisting of vertices (pages) and edges (links).
    $(P, L) \leftarrow G$                                ▷ Split graph into pages and links
    $I \leftarrow$ a vector of length $|P|$              ▷ The current PageRank estimate
    $R \leftarrow$ a vector of length $|P|$    ▷ The resulting better PageRank estimate
    for all entries $I_i \in I$ do
        $I_i \leftarrow 1/|P|$                    ▷ Start with each page being equally likely
    end for
    while $R$ has not converged do
        for all entries $R_i \in R$ do
            $R_i \leftarrow \lambda/|P|$ ▷ Each page has a $\lambda/|P|$ chance of random selection
        end for
        for all pages $p \in P$ do
            $Q \leftarrow$ the set of pages such that $(p, q) \in L$ and $q \in P$
            if $|Q| > 0$ then
                for all pages $q \in Q$ do
                    $R_q \leftarrow R_q + (1 - \lambda)I_p/|Q|$        ▷ Probability $I_p$ of being at page $p$
                end for
            else
                for all pages $q \in P$ do
                    $R_q \leftarrow R_q + (1 - \lambda)I_p/|P|$
                end for
            end if
            $I \leftarrow R$                         ▷ Update our current PageRank estimate
        end for
    end while
    return $R$
end procedure
```

A PageRank Implementation

Iteration:

- Steps:
  1. Make a new output file, R.
  2. Read L and I in parallel (since they're all sorted by URL).
  3. For each unique source URL, determine whether it has any outgoing links:
  4. If not, add its current PageRank value to the sum: T (terminals).
  5. If it does have outgoing links, write (source_url, dest_url, I_p/|Q|), where I_p is the current PageRank value, |Q| is the number of outgoing links, and dest_url is a link destination.
Do this for all outgoing links. Write this to R.
  6. Sort R by destination URL.
  7. Scan R and I at the same time. The new value of R_p is:
(1 - lambda) / #D (a fraction of the sum of all pages)
plus: lambda * sum(T) / #D (the total effect from terminal pages),
plus: lambda * all incoming mass from step 5. ()
  8. Check for convergence
  9. Write new R_p values to a new I file.