Web Services and Web Data

XJCO3011

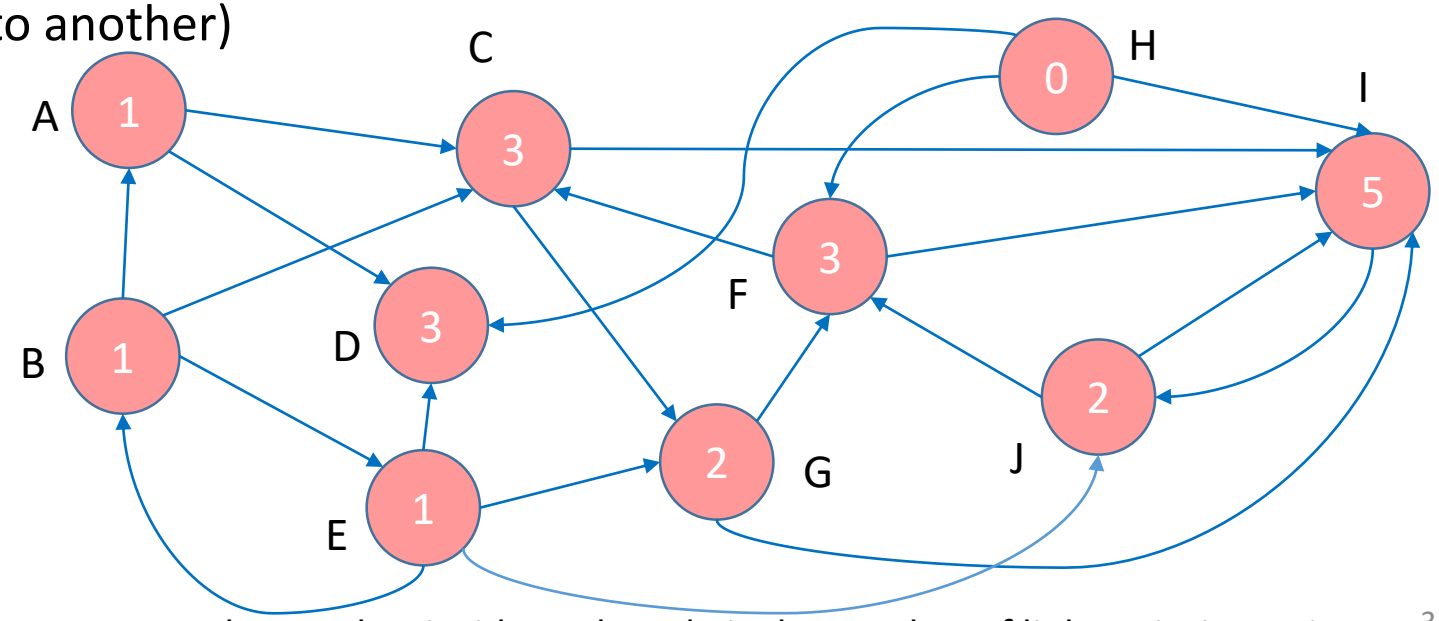
Session 11. Link Analysis and the PageRank Algorithm

Ranking Web Pages

- There are tens of billions of web pages, but most of them are not very interesting.
- Many of those pages are spam and contain no useful content at all.
- Other pages are personal blogs, wedding announcements, or family picture albums. These pages are interesting to a small audience, but not broadly.
- On the other hand, there are a few pages that are popular and useful to many people, including news sites and the websites of popular companies.
- How can search engine choose the most popular (and probably the correct) page?

Simple Link Analysis

- Links connecting pages are a key component of the Web.
- They help search engines rank web pages effectively.
- The simplest form of page ranking is based on counting the number of links pointing to a particular web page. The fact that a link exists at all is a vote of importance for the destination page.
- The higher this count, the higher the rank of this page in the search results, see the web shown below.
- This is based on the assumption that important pages are referenced (linked to) by many other pages.
- The problem with this method is that it does not take into account the importance (rank) of the source page (the page that points to the other page), for example , in the web sample shown below, which page is more important D, F or C?
- A second problem is that this method does not take into account the topic of the link (a page could be important to one topic but less important to another)
  - E.g., evaluating the importance of a health website
  - Page 1: A page specifically discussing heart disease, with lots of relevant information and links.
  - Page 2: An article about fitness, which includes some links about heart disease but is not the primary focus of the website.



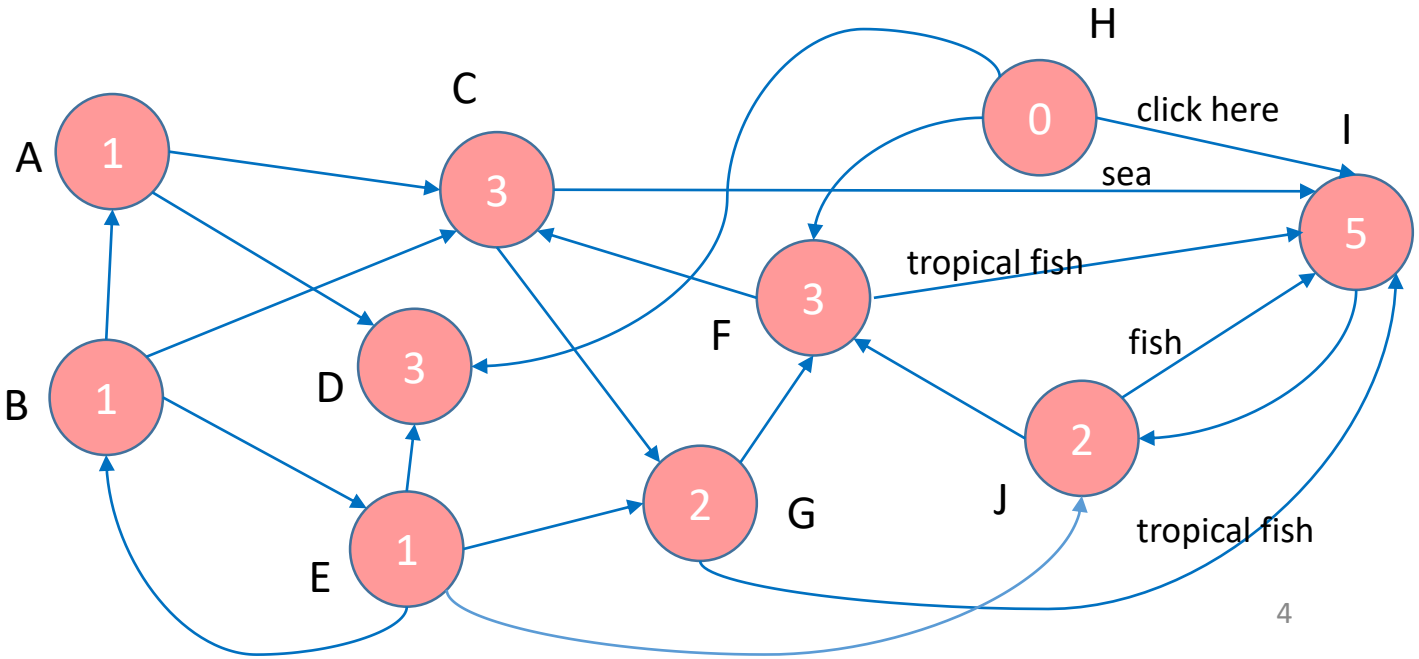
The number inside each node is the number of links pointing to it

Anchor Text Analysis

- A link in a web page is encoded in HTML with a statement such as:

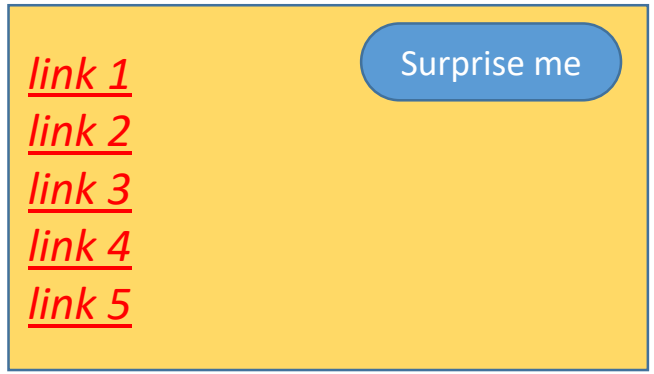
  tropical fish.

  In this link, "tropical fish" is called the anchor text, and http://www.somewhere.com is the *destination*. Both components are useful in the ranking process.
- Anchor text is particularly useful for ranking web pages because it is very short, two or three words, and those words succinctly describe the topic of the linked page.
- Many queries are very similar to anchor text in that they are also short topical descriptions of web pages.
- Search through all links in the collection of pages, looking for anchor text that matches the user's query. Each time there is a match, add 1 to the score of the destination page. Pages would then be ranked in decreasing order of this score.
- This algorithm has some glaring faults, not the least of which is how to handle the query "click here", and it does not take into account the rank of the source page.



Alice Again!

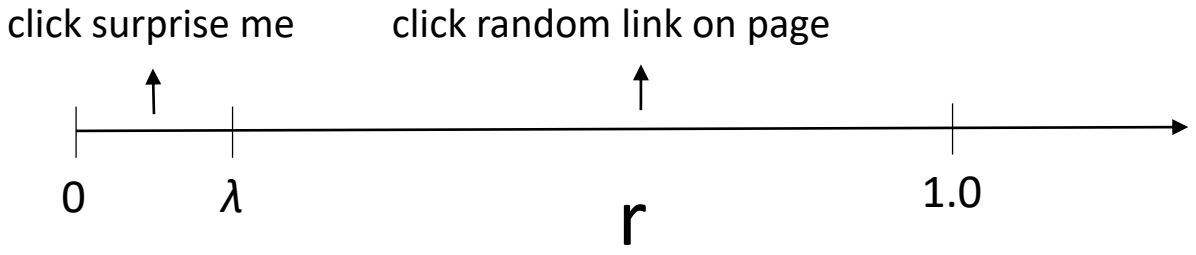
- One of the most popular page ranking algorithms is called PageRank and has been used by Google for many years.
- PageRank is based on the idea of a random surfer (as in web surfer).
- Imagine that Alice is using her web browser to wander aimlessly between web pages.
- Her browser has a special "surprise me" button at the top that will jump to a random web page when she clicks it.
- Each time a web page loads, she chooses whether to click the "surprise me" button or whether to click one of the links on the web page.
- If she clicks a link on the page, she has no preference for any particular link; she just picks one randomly.
- Alice is sufficiently bored that she intends to keep browsing the Web like this forever



Lambda

- To put this in a more structured form, Alice browses the Web using this algorithm:
  1. Choose a random number *r* between 0 and 1.
  2. If $r < \lambda$:
     - Click the "surprise me" button.
  3. If $r \geq \lambda$:
     - Click a link at random on the current page.
  4. Start again.

- Typically we assume that λ is fairly small, so Alice is much more likely to click a link than to pick the "surprise me" button



The PageRank Web Surfing Assumptions

- Even though Alice's path through the web pages is random, Alice will still see popular pages more often than unpopular ones because Alice often follows links, and links tend to point to popular pages
- So, we expect that Alice will end up at a university website, for example, more often than a personal website, but less often than the CNN website.
- The "surprise me" button can guarantee that eventually she will reach every page on the Internet.
- Without the "surprise me" button, she would get stuck on pages that no longer pointed to any page, or pages that formed a loop

A Page Rank is a Probability Value

- Now suppose that while Alice is browsing, you happened to walk into her room and glance at the web page on her screen.
- What is the probability that she will be looking at the CNN web page when you walk in? That probability is CNN's PageRank.
- Every web page on the Internet has a PageRank, and it is uniquely determined by the link structure of web pages.
- PageRank has the ability to distinguish popular pages, those with many incoming links, or those that have links from popular pages, from unpopular ones.
- The PageRank value can help search engines sift through millions of pages that contain the word "eBay" to find the one that is most popular (www.ebay.com).

Computing PageRank

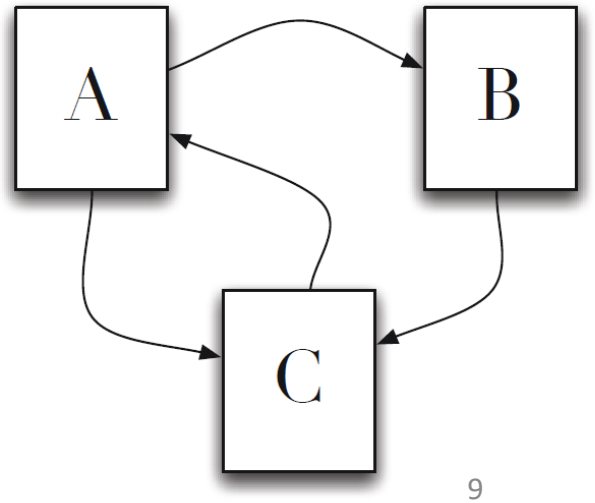
- Suppose for the moment that the Web consists of just three pages, A, B, and C, with links as shown.
- The PageRank of page C, which is the probability that Alice will be looking at this page, will depend on the PageRank of pages A and B.
- Since Alice chooses randomly between links on a given page, if she starts in page A, there is a 50% chance that she will go to page C
- Another way of saying this is that the PageRank for a page is divided evenly between all the outgoing links
- If we ignore the "surprise me" button, this means that the PageRank of page C, represented as PR(C), can be calculated as

$$PR(C) = \frac{PR(A)}{2} + \frac{PR(B)}{1}$$

- More generally, we could calculate the PageRank for any page *u* as

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

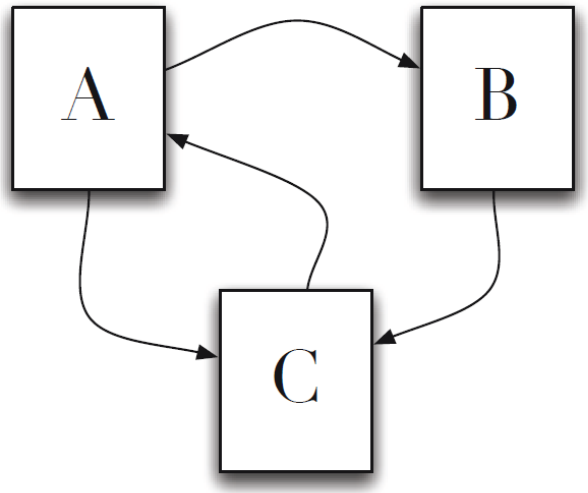
where Bu is the set of pages that point to u,
and Lv is the number of outgoing links from page v (not counting duplicate li



Computing PageRank …

- There is an obvious problem here: we don't know the PageRank values for the pages, because that is what we are trying to calculate.
- If we start by assuming that the PageRank values for all pages are the same (1/3 in this case), then it is easy to see that we could perform multiple iterations of the calculation.
- For example, in the first iteration

  PR(C) = 0.33/2 + 0.33 = 0.5, PR(A) = 0.33, and PR(B) = 0.17.



- In the next iteration:

  PR(C) = 0.33/2 + 0.17 = 0.33, PR(A) = 0.5, and PR(B) = 0.17.
- In the third iteration, PR(C) = 0.42, PR(A) = 0.33, and PR(B) = 0.25.
- After a few more iterations, the PageRank values converge to the final values of

  PR(C) = 0.4, PR(A) = 0.4, and PR(B) = 0.2.

Computing PageRank …

- If we take the "surprise me" button into account, part of the PageRank for page C will be due to the probability of coming to that page by pushing the button.
- Given that there is a 1/3 chance of going to any page when the button is pushed, and that the probability of pushing the button is λ, the contribution of the button to the PageRank of C will be λ/3. This means that the total PageRank for C is now:

$$PR(C) = \frac{\lambda}{3} + (1-\lambda) \cdot (\frac{PR(A)}{2} + \frac{PR(B)}{1})$$

- Similarly, the general formula for PageRank is:

$$PR(u) = \frac{\lambda}{N} + (1-\lambda) \cdot \sum_{v \in B_u} \frac{PR(v)}{L_v}$$

where *N* is the number of pages being considered. The typical value for *λ* is 0.15

Simplified PageRank Algorithm

```
// initialize all page rank values
for each page p in web
    page rank of  p = 1/number of pages

repeat until page rank values converge
{
   for each page p in web
   {
       new page rank of p = 0    // we will accumulate a new value here
       for each page q pointing at p
             new page rank of p += old page rank of q / number of links coming out of q
    }
    // now make the old values = the new values, to prepare for the next iteration
    for each page p in web
      old value of page rank of p  = new value of page rank of p
}
```

PageRank Algorithm with λ

Note: Pages with no outbound links are *rank sinks*, in that they accumulate PageRank but do not distribute it. In this algorithm, we assume that these pages link to all other pages in the collection.

```
1:  procedure PageRank(G)
2:                  ▷ G is the web graph, consisting of vertices (pages) and edges (links).
3:      (P, L) ← G                                   ▷ Split graph into pages and links
4:      I ← a vector of length |P|                   ▷ The current PageRank estimate
5:      R ← a vector of length |P|                   ▷ The resulting better PageRank estimate
6:      for all entries I_i ∈ I do
7:          I_i ← 1/|P|                              ▷ Start with each page being equally likely
8:      end for
9:      while R has not converged do
10:         for all entries R_i ∈ R do
11:             R_i ← λ/|P|          ▷ Each page has a λ/|P| chance of random selection
12:         end for
13:         for all pages p ∈ P do
14:             Q ← the set of pages such that (p, q) ∈ L and q ∈ P
15:             if |Q| > 0 then
16:                 for all pages q ∈ Q do
17:                     R_q ← R_q + (1 − λ)I_p/|Q| ▷ Probability I_p of being at page p
18:                 end for
19:             else
20:                 for all pages q ∈ P do
21:                     R_q ← R_q + (1 − λ)I_p/|P|
22:                 end for
23:             end if
24:             I ← R                                ▷ Update our current PageRank estimate
25:         end for
26:     end while
27:     return R
28: end procedure
```

That's it Folks

Further Reading

Chapter 4: Search Engines Information Retrieval in Practice by W. Bruce Croft, Donald Metzler, and Trevor Strohman