Eric Roberts
CS 106B

Handout #53
March 11, 2013

More Algorithms

More Algorithms for Trees and Graphs

Eric Roberts
CS 106B
March 11, 2013

Outline for Today

- The plan for today is to walk through some of my favorite tree and graph algorithms, partly to demystify the many real-world applications that make use of those algorithms and partly to emphasize the elegance and power of algorithmic thinking.
- These algorithms include:
  – Google's Page Rank algorithm for searching the web
  – The Directed Acyclic Word Graph format used in `Lexicon`
  – The heap data structure used to create efficient priority queues

Google

The big innovation of the late 1990s is the development of search engines, which began with Alta Vista at DEC's Western Research Lab and reaching its modern pinnacle with Google, which was founded by Stanford graduate students Larry Page and Sergey Brin in 1998.

Larry Page and Sergey Brin

Page Rank

The heart of the Google search engine is the page rank algorithm, which was described in a 1999 by Larry Page, Sergey Brin, Rajeev Motwani, and Terry Winograd.

**The PageRank Citation Ranking:
Bringing Order to the Web**

January 29, 1998

Abstract

The importance of a Webpage is an inherently subjective matter, which depends on the reader's interests, knowledge and attitudes. But there is still much that can be said objectively about the relative importance of Web pages. This paper describes PageRank, a method for rating Web pages objectively and mechanically, effectively measuring the human interest and attention devoted to them.

We compare PageRank to an idealized random Websurfer. We show how to efficiently compute PageRank for large numbers of pages. And,we show how to apply PageRank to search and to user navigation.

Page Rank Algorithm

The page rank algorithm gives each page a rating of its *importance*, which is a recursively defined measure whereby a page becomes important if other important pages link to it.

One way to think about page rank is to imagine a random surfer on the web, following links from page to page. The page rank of any page is roughly the probability that the random surfer will land on a particular page. Since more links go to the important pages, the surfer is more likely to end up there.

Markov Processes

A simple example of a Markov process is illustrated by this table, which shows the likelihood of a particular weather pattern for tomorrow given the weather for today.

Tomorrow will be

If today is

	☀	⛅	🌧
☀	0.85	0.10	0.05
⛅	0.60	0.25	0.15
🌧	0.40	0.40	0.20

Markov Processes

What, then, is the likely weather two days from now, given that you know what the weather looks like today?

The day after tomorrow will be

If today is

	☀️	🌥️	🌧️
☀️	0.81	0.13	0.06
🌥️	0.72	0.18	0.10
🌧️	0.66	0.22	0.12

Markov Processes

What if you then repeat the process for ten days?

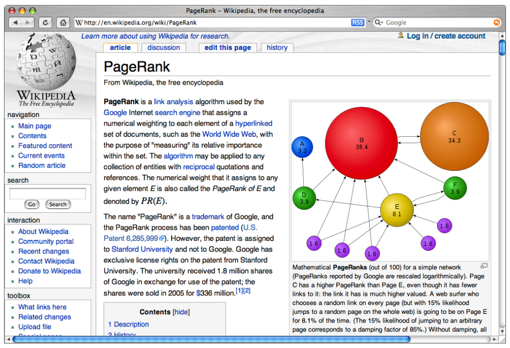
That far out, it doesn't matter what today's weather is.

Ten days from now will be

If today is

	☀️	🌥️	🌧️
☀️	0.77	0.14	0.07
🌥️	0.77	0.14	0.07
🌧️	0.77	0.14	0.07

Google's Page Rank Algorithm



PageRank

From Wikipedia, the free encyclopedia

PageRank is a link analysis algorithm used by the Google Internet search engine that assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set. The algorithm may be applied to any collection of entities with reciprocal quotations and references. The numerical weight that it assigns to any given element *E* is also called the *PageRank of E* and denoted by $PR(E)$.

The name "PageRank" is a trademark of Google, and the PageRank process has been patented (U.S. Patent 6,285,999). However, the patent is assigned to Stanford University and not to Google. Google has exclusive license rights on the patent from Stanford University. The university received 1.8 million shares of Google in exchange for use of the patent; the shares were sold in 2005 for $336 million.[1][2]

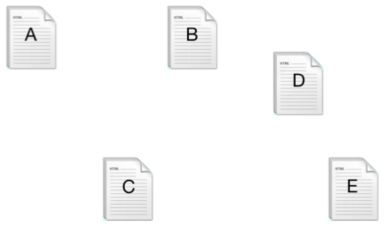
Contents [hide]

1 Description

Mathematical **PageRanks** (out of 100) for a simple network (PageRanks reported by Google are rescaled logarithmically). Page C has a higher PageRank than Page E, even though it has fewer links to it: the link it has is much higher valued. A web surfer who chooses a random link on every page (but with 15% likelihood jumps to a random page on the whole web) is going to be on Page E for 8.1% of the time. (The 15% likelihood of jumping to an arbitrary page corresponds to a damping factor of 85%.) Without damping, all

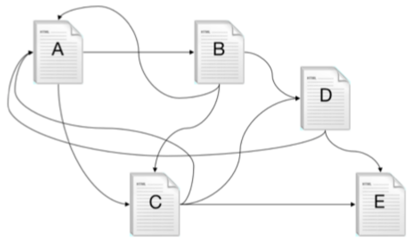
The Page Rank Algorithm

1. Start with a set of pages.



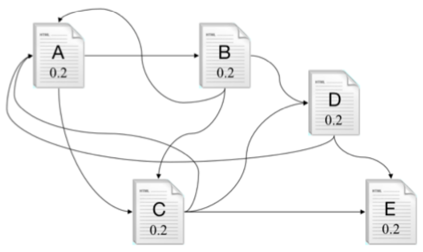
The Page Rank Algorithm

2. Crawl the web to determine the link structure.



The Page Rank Algorithm

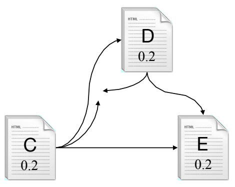
3. Assign each page an initial rank of 1 / *N*.



The Page Rank Algorithm

4. Successively update the rank of each page by adding up the weight of every page that links to it divided by the number of links emanating from the referring page.

   - In the current example, page E has two incoming links, one from page C and one from page D.
   - Page C contributes 1/3 of its current page rank to page E because E is one of three links from page C. Similarly, page C offers 1/2 of its rank to E.
   - The new page rank for E is

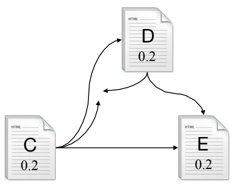


$$PR(E) = \frac{PR(C)}{3} + \frac{PR(D)}{2} = \frac{0.2}{3} + \frac{0.2}{2} \approx 0.17$$

The Page Rank Algorithm

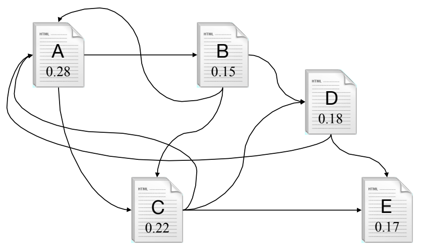
5. If a page (such as E in the current example) has no outward links, redistribute its rank equally among the other pages in the graph.

   - In this graph, 1/4 of E's page rank is distributed to pages A, B, C, and D.
   - The idea behind this model is that users will keep searching if they reach a dead end.



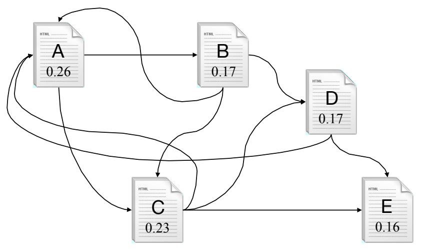
The Page Rank Algorithm

7. Apply this redistribution to every page in the graph.



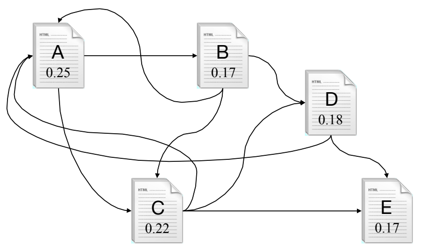
The Page Rank Algorithm

8. Repeat this process until the page ranks stabilize.

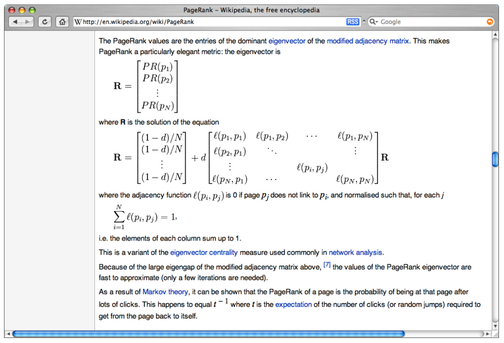


The Page Rank Algorithm

9. In practice, the Page Rank algorithm adds a ***damping factor*** at each stage to model the fact that users stop searching.



Page Rank Relies on Mathematics


PageRank – Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/PageRank
RSS
Google

The PageRank values are the entries of the dominant eigenvector of the modified adjacency matrix. This makes PageRank a particularly elegant metric: the eigenvector is

$$\mathbf{R} = \begin{bmatrix} PR(p_1) \\ PR(p_2) \\ \vdots \\ PR(p_N) \end{bmatrix}$$

where **R** is the solution of the equation

$$\mathbf{R} = \begin{bmatrix} (1-d)/N \\ (1-d)/N \\ \vdots \\ (1-d)/N \end{bmatrix} + d \begin{bmatrix} \ell(p_1,p_1) & \ell(p_1,p_2) & \cdots & \ell(p_1,p_N) \\ \ell(p_2,p_1) & \ddots & & \vdots \\ \vdots & & \ell(p_i,p_j) & \\ \ell(p_N,p_1) & \cdots & & \ell(p_N,p_N) \end{bmatrix} \mathbf{R}$$

where the adjacency function $\ell(p_i,p_j)$ is 0 if page p_j does not link to p_i, and normalised such that, for each j

$$\sum_{i=1}^{N} \ell(p_i,p_j) = 1,$$

i.e. the elements of each column sum up to 1.

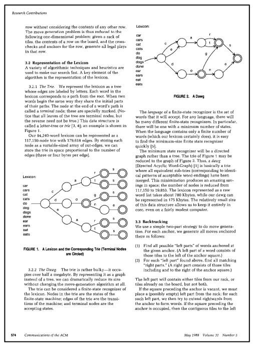
This is a variant of the eigenvector centrality measure used commonly in network analysis.

Because of the large eigengap of the modified adjacency matrix above, [7] the values of the PageRank eigenvector are fast to approximate (only a few iterations are needed).

As a result of Markov theory, it can be shown that the PageRank of a page is the probability of being at that page after lots of clicks. This happens to equal t^{-1} where t is the expectation of the number of clicks (or random jumps) required to get from the page back to itself.

Directed Acyclic Word Graphs

- The **`Lexicon`** class provides a highly time- and space-efficient representation of a word list.
- The data representation used in the **`Lexicon`** class is called a ***Directed Acyclic Word Graph*** or DAWG, which was first described in a 1988 paper by Appel and Jacobson.
- The DAWG structure is based on a much older data structure called a ***trie***, developed by Ed Fredkin in 1960. (The trie data structure is described in the text on page 490.)



The Trie Data Structure

- The trie representation of a word list uses a tree in which each arc is labeled with a letter of the alphabet. In a trie, the words themselves are represented implicitly as paths to a node.

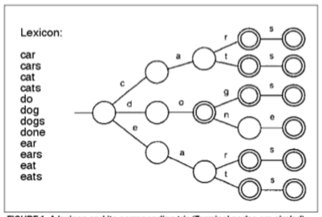


FIGURE 1. A lexicon and its corresponding trie (Terminal nodes are circled)

Going to the DAWGs

- The new insight in the DAWG is that you can combine nodes that represent common endings.

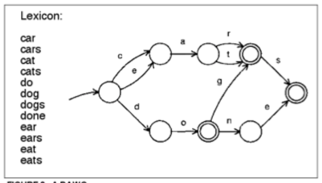


FIGURE 2. A DAWG

- As the authors note, "this minimization produces an amazing savings in space; the number of nodes is reduced from 117,150 to 19,853."

The Heap Algorithm

- If you implement them in the obvious way using either arrays or linked lists, priority queues require $O(N)$ time. Because priority queues are essential to both Dijkstra's and Kruskal's algorithms, making them efficient improves performance.
- The standard algorithm for implementing priority queues uses a data structure called a ***heap***, which makes it possible to implement priority queue operations in $O(\log N)$ time.
- A heap is a binary tree with three additional properties:
  - The tree is ***complete***, which means that it is not only completely balanced but that each level of the tree is filled as far to the left as possible.
  - The root node of the tree has higher priority than the root of either of its subtrees.
  - Every subtree is also a heap.

Exercise: Tracing the Heap Algorithm

Insert in order: 17 93 20 42 68 11

Dequeue the top priority element