Graphs in the Real* World

* “real” as in “virtual”

Examples?

- Already seen a few:
- Cities & Roads
- Airports & Flight paths
- Others?

Web as a graph

- Node in the graph: a web page
- Edge in the graph: an HTML link

Web as a graph

- Web page is a node
- What is a web page?
  - http://www.cs.cornell.edu/
  - http://www.cs.cornell.edu/Courses.html
  - http://www.cs.cornell.edu/index.html
  - http://www.cs/courses.html
- How many web pages?
  - Each frame counts or only the frame set together?
- What about dynamic pages?
  - http://www.google.com/?q=“Nifty Projects”

Web as a graph

- HTML link is an edge
- Self-loops?
- More than one link from A to B?
- Given web page (whatever that means), we can enumerate the links:
  - Just search for …text… tags
  - Then this page has edge to "…url…" page
  - (It's a little more complex in practice, of course)

Web Search

- Given query, find most relevant pages on the web
  - Query = “Cornell cs 211 summer 2004”
  - Answer:
    - http://www.cs.cornell.edu/courses/cs211/2004su/
      - and a bunch of stuff below this
    - http://www.cs.cornell.edu/courses/cs211/2003su/
      - not the right year, but pretty close
    - http://www.cs.cornell.edu/~kwalsh/
      - because I am the instructor, my page is “relevant”
    - many others

Pesky Questions

- What is “the web”?
- What is the query?
- What does “most relevant” mean?
- What should output look like?
- Anything else?
  - What is a page?

Some Simplification

- Assume there is some definition of “a web page”
  - Usually includes being accessible by HTTP from anywhere on the internet
- Assume there is some query format
- Assume there is some primitive evaluate function:

  boolean matches(URL url, Query q)

  that looks at the body of a page, and decides if the query matches the page or not.
  - Might check if all query words appear in the page
  - Or some of the query words
  - Or maybe does text analysis
  - Might look at title, contents, author-supplied keywords, colors, etc.

Pesky Questions

- What is “the web”?
- ✓ What is the query?
- What does “most relevant” mean?
- What should output look like?
- Anything else?
  - ✓ What is a page?

What is the Web? According to Google:

The World Wide Web (WWW) is a hypertext system that provides access to the Internet through programs called network information browsers. In other words, the World Wide Web ("the Web") is only one way to access the Internet. This system allows text, graphics, audio and video files to be mixed together.

→ good definition?

Is “gopher:cit.cornell.edu” part of the web?

How about AOL?

How about C:\kwalsh\mybookmarks.html?

A few more tries (also from Google)

- The Web is an Internet service that links information, business and media into one global network that you can access using a browser. Although technically different, the Web and the Net are referred to interchangeably. "Web Page" refers to a page on the web and "Web Site" refers to related pages on the World Wide Web.
- The profusion of HTML pages that are stored and are accessible via the network of the Internet.
- The World Wide Web (also called WWW) is the fastest growing part of the Internet. It consists of millions of items: text files, sounds, video clips and pictures that have built-in links or connections to other Internet pages. This interconnected system of pages can be visualized as a web and that is the reason for the name World Wide Web.
- Any of these better? worse?

A Quirky Definition

- Web consists of all pages* that can be found by following links* from
  http://www.yahoo.com/index.html

- * where *page* and *link* are defined elsewhere

- Better? Worse?

Searching the Web: Take 1

- Given query, search the web for matches
- Output is randomly-ordered list of URLs that all return *true* when passed to *matches(url, query)*

Web Crawling

- Start with toDo = *root nodes*, web = *root nodes*
  - http://www.yahoo.com
  - http://www.dmoz.org
  - http://directory.google.com
- Then start following links:
  - Pick and remove any page from toDo
  - Enumerate links
    - Add each destination to toDo set
    - Add each destination to web set
- As each url is added to web set, check it for *matches* against the query
- Keep going until you have enough matches

Web Crawling: Problems

- Watch out for self-loops and cycles
- Watch out for useless dead-ends
  - infinitely deep holes?
- Most of crawling time is spent waiting for data to arrive
  - Use threads to do many pages at once
  - Use multi-processors, each processor having many threads
  - Use multiple machines, each with …
- Really an awful lot of work for just one query

Web crawling: Take 2

- Generate a big database of the entire web
- To answer a query, just scan through your database, looking for matches

Output

- A lot of user-friendliness needed here
  - query = “Cornell”
  - Results:
    - http://www.cornell.edu
    - http://www.cornell.edu/index.html
    - http://www.cornell.edu/intro.html
    - etc.
- Group by “site” (whatever that means)
- Eliminate duplicates (or near duplicates?)
- Order by relevance

Relevance

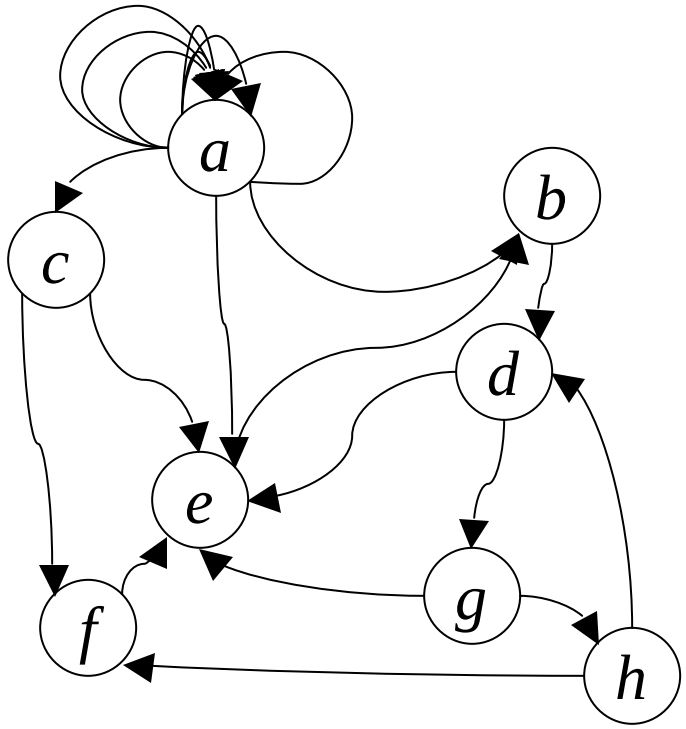
- Given a bunch of URLs that all match the query, decide which are the *most relevant*, and which are not.
- One approach: define a *double score(url, query)* function
  - Should return small numbers for least relevant pages, large numbers for most relevant pages
  - How to implement?

Relevance

- Given a bunch of URLs that all match the query, decide which are the *most relevant*, and which are not.
- One approach: define a *double score(url, query)* function
  - Should return small numbers for least relevant pages, large numbers for most relevant pages
  - How to implement?
    - Number of times a keyword shows up in the page
    - Number of keywords that appear
    - Number of related words that appear
    - How close the keywords are together on the page
    - If the keyword shows up in the title, or just the body
    - etc.

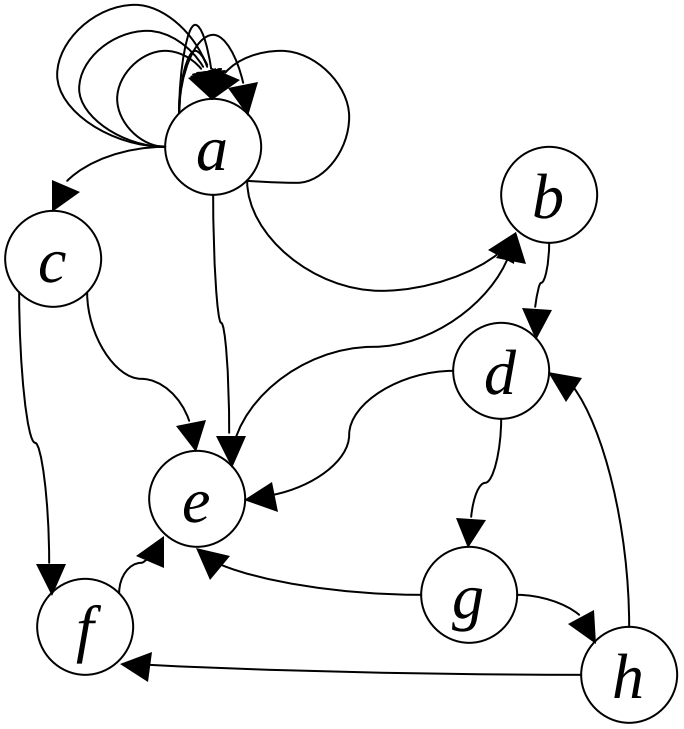
Relevance from graph structure

- E.g. We might think that lots of links to a page means the page is important.



Relevance from graph structure

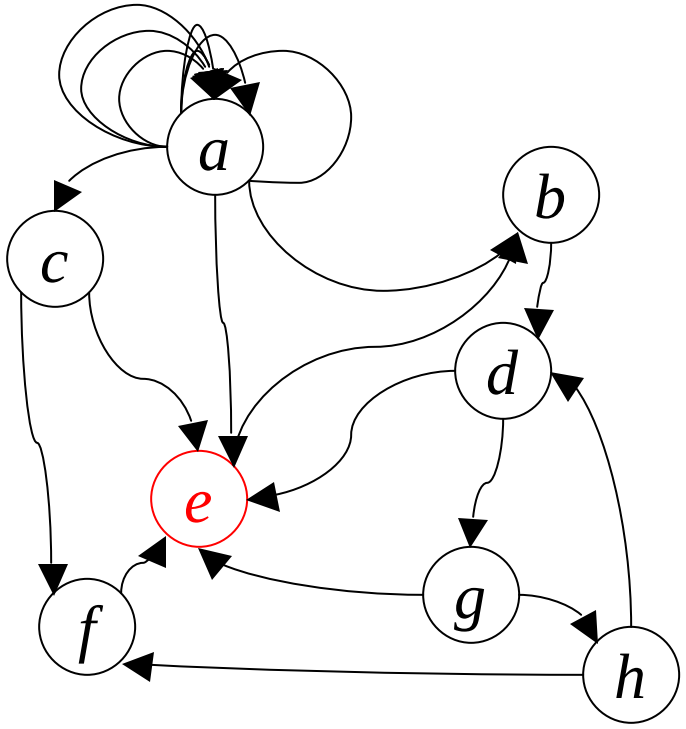
- E.g. We might think that lots of links to a page means the page is important.
  - Self loops don't count
  - Neither do links within a "site"



Score Function Attempt 1

- Score of page is number of (external) links point to the page. Same as *in-degree*.

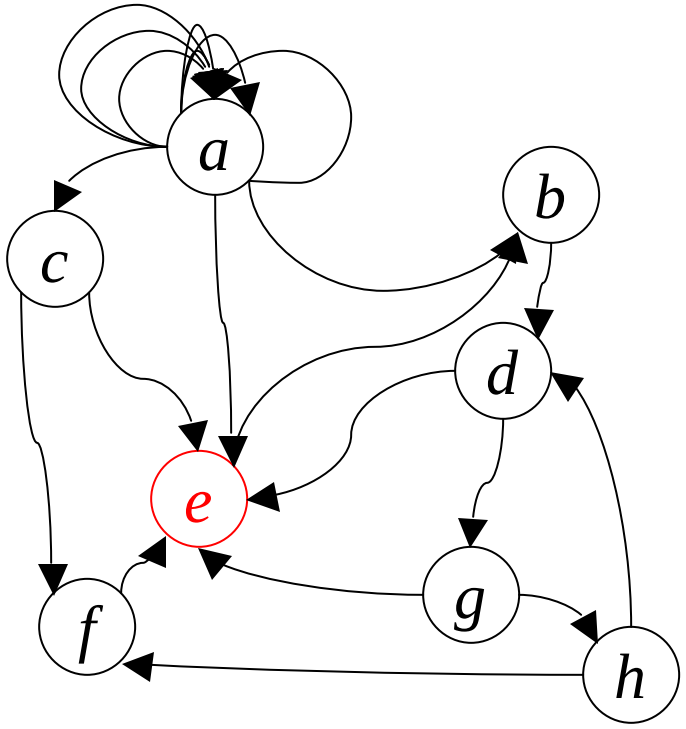
- score(a) = 0
- score(b) = 2
- score(c) = 1
- score(d) = 2
- score(e) = 6
- etc.



Score Function Attempt 2

- Highly scored pages “share” some of their high score with the things they point to.

- score(a) = 0
- score(b) = 2+share(e)
- score(c) = 1+…
- score(d) = 2+share(b,h)
- score(e) = 6+…
- etc.
- how much to share?
- should sharer loose points?



Digression on Randomness

- Randomness is wonderful (sometimes)
  - Quicksort is very fast on random input
  - Quicksort is also very fast if you randomly permute the input before attempting to sort (yes, really).
- How to calculate pi
  - Pick N random points in a 2x2 square centered at origin
  - Calculate M, number of points <= distance 1 from origin
  - pi = 4*M/N
- How to tell if two large files are identical:
  - hash each to a single number (say 100 digits)
  - results are essentially random numbers
  - compare hashes

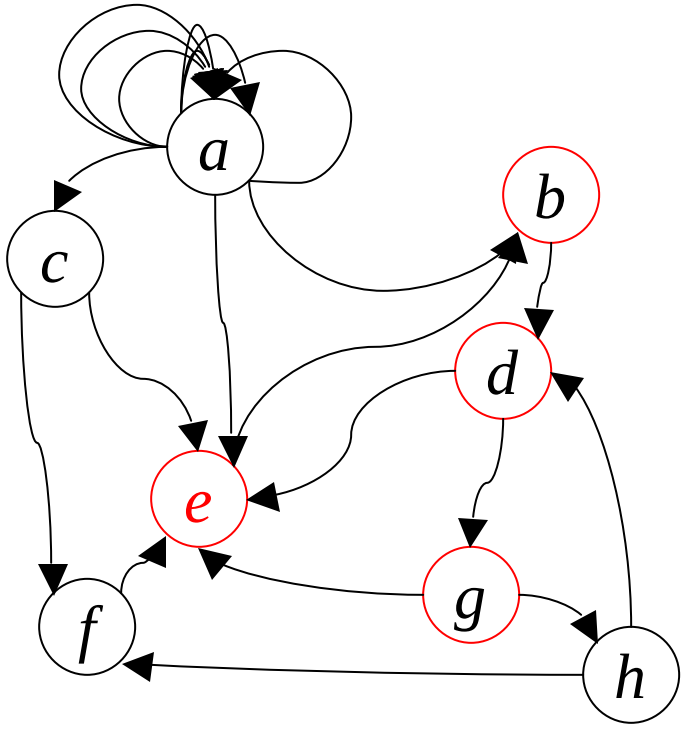
Random Walk on Graphs

- Start at some node M_0
- Follow random edges, passing through M_1, $M_{2, \ldots}$
- End at some node M_n
- Where do you pass through?
- Where do you end up?
- Pitfalls?

Score Function Attempt 3

- Take random walk from anywhere, generating list M
- Score is the number of times URL shows up in M

- g – e – b – d – e – b – d – g – h – f – e …
- score(e) = 3
- score(b) = 2
- score(d) = 2
- score(g) = 2

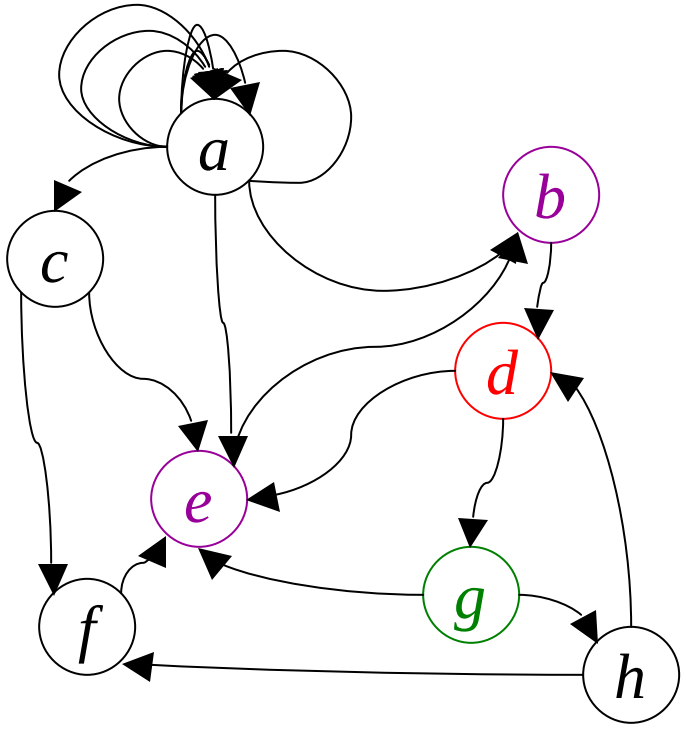


Score Function Attempt 3

- Take random walk from anywhere, generating list M
- Score is the number of times URL shows up in M
- Use fraction instead of raw count

g e b d g e b d g e b d e b d g e b d g e b d e b d g e b d g h f e b
d e b d g h d e b d g e b d g h d g h f e b d g h d g h d e b d g e b
d g e b d e b d g e b d e b d e b d e b d g e b d e b d g h d …
(1000 node walk)

score(a) = 0.0
score(b) = 0.235
score(c) = 0.0
score(d) = 0.268
score(e) = 0.235
score(f) = 0.045
score(g) = 0.141
score(h) = 0.077



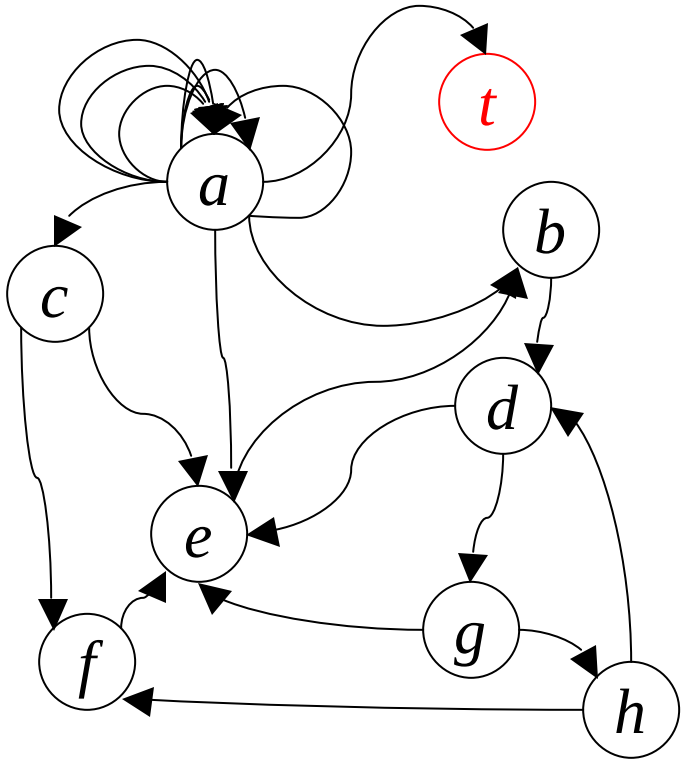
Is it consistent?

score(a) =	0.0	0.0010	0.0	0.0	0.0	0.0	0.0	0.0	0.0010	0.0
score(b) =	0.251	0.251	0.239	0.24	0.248	0.237	0.244	0.252	0.247	0.239
score(c) =	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
score(d) =	0.279	0.278	0.268	0.273	0.278	0.27	0.275	0.285	0.281	0.274
score(e) =	0.25	0.25	0.24	0.24	0.248	0.237	0.244	0.253	0.246	0.238
score(f) =	0.034	0.031	0.038	0.038	0.033	0.038	0.038	0.028	0.028	0.04
score(g) =	0.124	0.131	0.15	0.138	0.131	0.147	0.131	0.122	0.136	0.135
score(h) =	0.063	0.059	0.066	0.072	0.063	0.072	0.069	0.061	0.062	0.075

Problems?

- Might get stuck in a trap
  - Malicious? Of course!

score(a) = 0.0
score(b) = 0.0
score(c) = 0.0
score(d) = 0.0
score(e) = 0.0
score(f) = 0.0
score(g) = 0.0
score(h) = 0.0
score(t) = 1.0

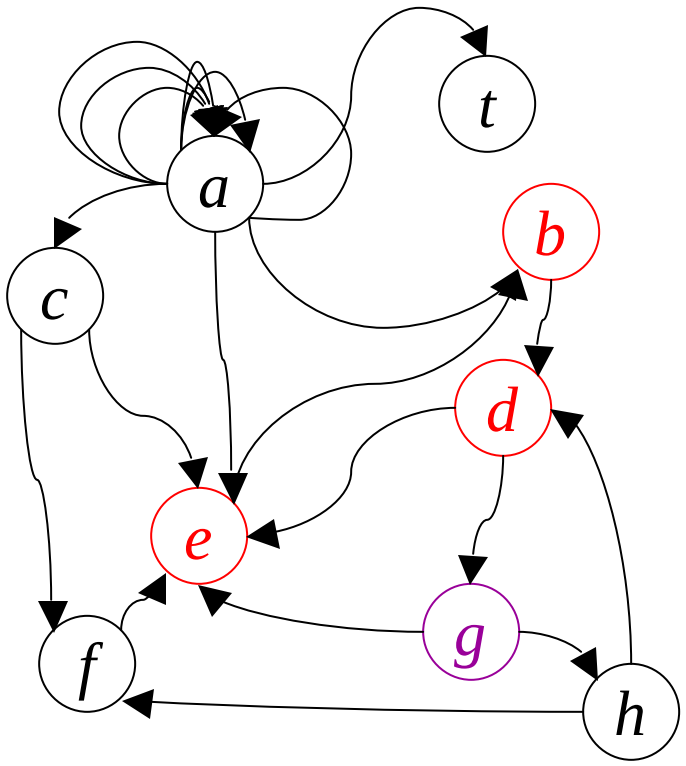


Score Function Attempt 4

- Take many shorter random walks from random places
  - Could set maximum walk to be very small
  - Or could take random jump with some probability (say 10%)

t t t t t t t t t t # d g e b d g e b d e b d e b d # c e b d g h d e b # f e
b d g e # h d # d # f e b d g e b d e b d g h d e # a f e b d g h f e b
d g e b d g h f e b d e b d e b # b d e b d e b d g h f e b d g # h d g
h f e b d e b d g e b d g e # g e b d e b d e b d # c e b d e b d g e b
d g h d e # f e # c # g e b d e b d e b d e b d # t t t t t # e b # a c e
e # t t t t t t # e b # a b d e b d e b d g e b d e b d e b d e b d ...

score(a) =	0.011	0.011	0.010	0.011
score(b) =	0.215	0.219	0.199	0.231
score(c) =	0.009	0.018	0.010	0.011
score(d) =	0.241	0.223	0.213	0.240
score(e) =	0.220	0.224	0.208	0.236
score(f) =	0.042	0.044	0.042	0.044
score(g) =	0.131	0.105	0.100	0.114
score(h) =	0.072	0.055	0.056	0.056
score(t) =	0.060	0.102	0.163	0.058



PageRank

- Start with all pages that *match*
- Add in all pages one-hop away from them
- Run a long random walk from every node
  - ~ 10% random-jump probability
  - ignore ~75% of links within a site
  - some other special cases for dealing with cheaters, odd cases, etc.
- Do this with matrices, probabilities & eigenvectors instead of actual graph walks
- Voila!

Other Applications

- Finding “web communities/clusters”
  - Lots of links between pages inside cluster
  - Fewer links to pages outside cluster
- Finding “index/hub” type pages
  - Lots of links to highly ranked pages
- Finding “definitive” page
  - The most downstream page on a topic?
- Finding a “random” page
- Estimating the size of the web, diameter, connectivity, etc.

→ These are topics of ongoing research

  - Many current approaches use random walks