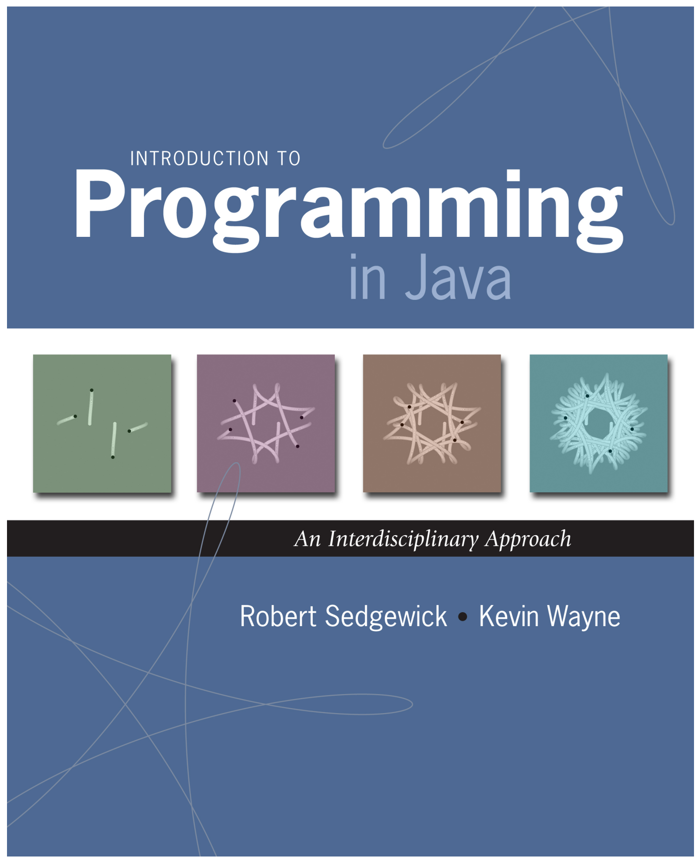
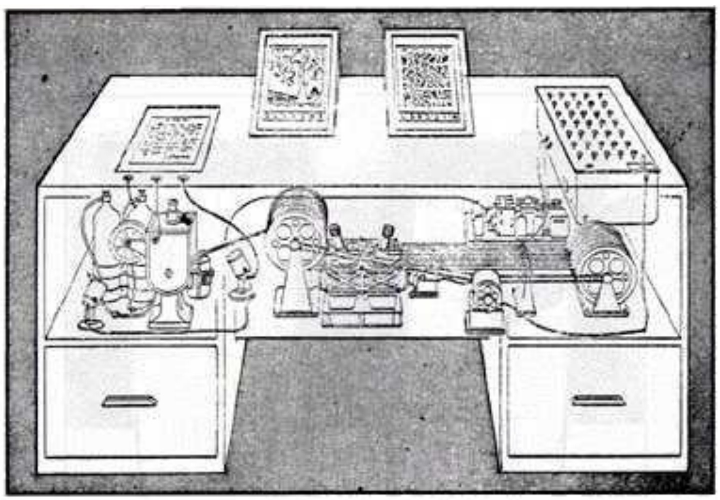
1.6 Case Study: Random Surfer



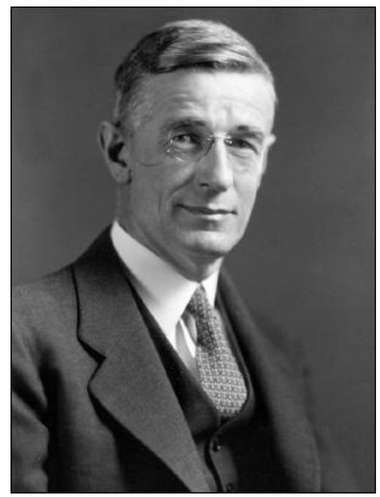
Memex

Memex. [Vannevar Bush, 1936] Theoretical hypertext computer system; pioneering concept for world wide web.

- Follow links from book or film to another.
- Tool for establishing links.



Life magazine, November 1945



Vannevar Bush

World Wide Web

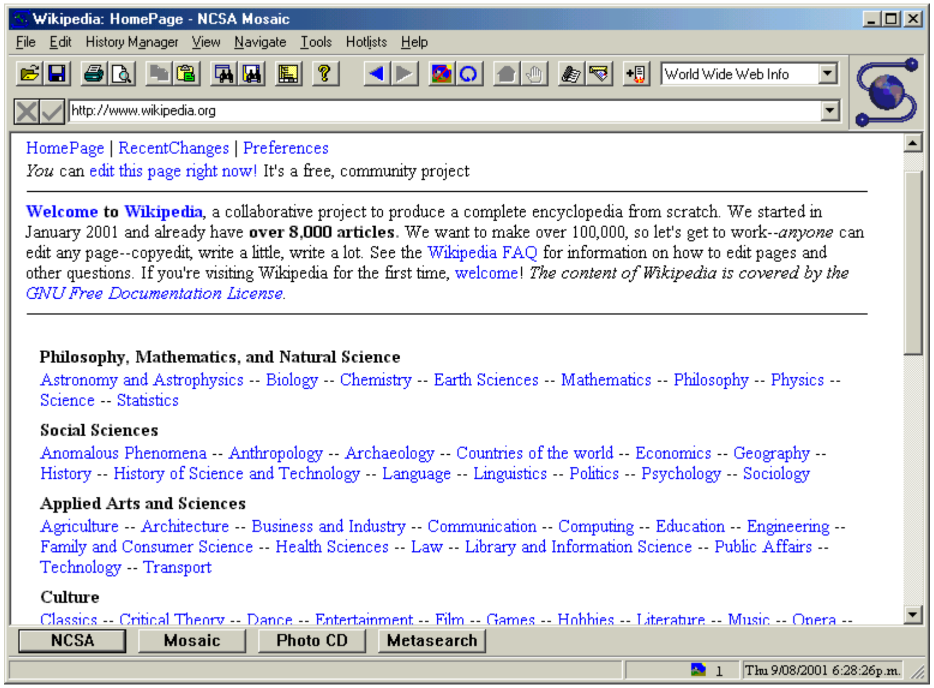
World wide web. [Tim Berners-Lee, CERN 1980] Project based on hypertext for sharing and updating information among researchers.

first Web server

Sir Tim Berners-Lee

Web Browser

Web browser. Killer application of the 1990s.



Library of Babel

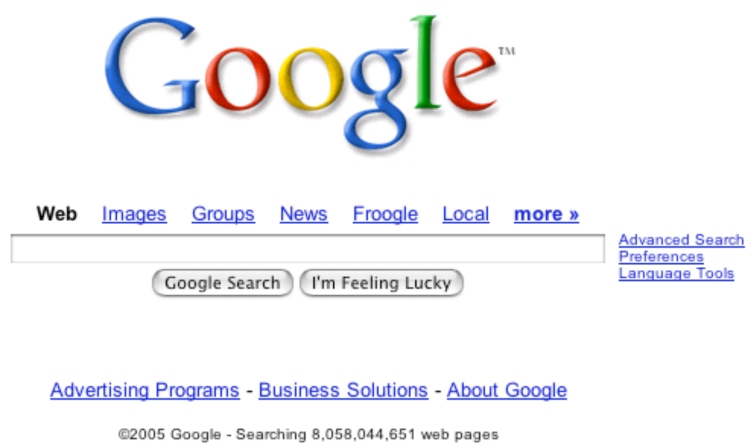
La biblioteca de Babel. [Jorge Luis Borge, 1941]

When it was proclaimed that the Library contained all books, the first impression was one of extravagant happiness… There was no personal or world problem whose eloquent solution did not exist in some hexagon.

this inordinate hope was followed by an excessive depression. The certitude that some shelf in some hexagon held precious books and that these precious books were inaccessible seemed almost intolerable.

Web Search

Web search. Killer application of the 2000s.



Web Search

Web Search

Relevance. Is the document similar to the query term?
Importance. Is the document useful to a variety of users?

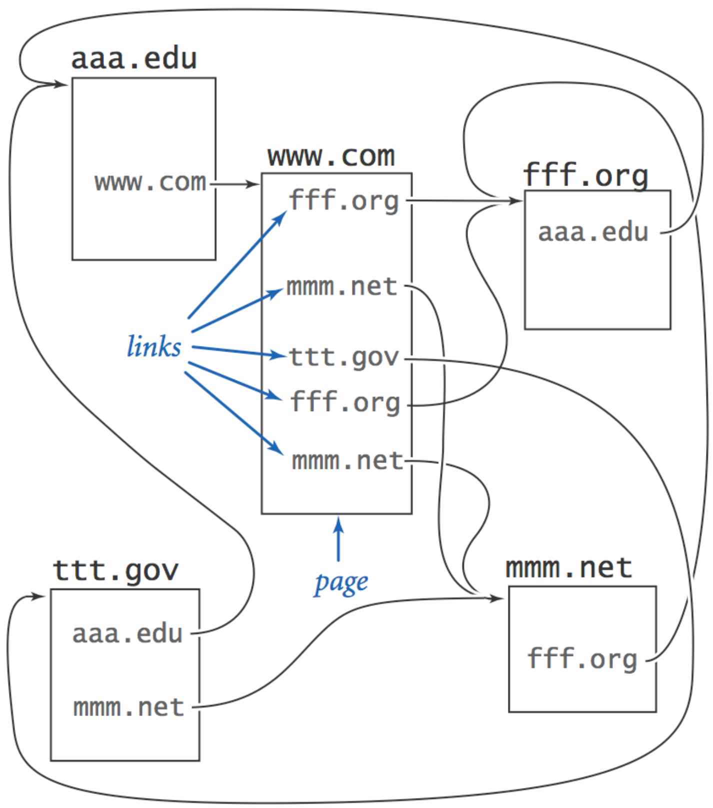
Search engine approaches.

- Paid advertisers.
- Manually created classification.
- Feature detection, based on title, text, anchors, …
- "Popularity."

PageRank

Google's PageRank™ algorithm. [Sergey Brin and Larry Page, 1998]

- Measure popularity of pages based on hyperlink structure of Web.
  Revolutionized access to world's information.



90-10 Rule

Model. Web surfer chooses next page:

- 90% of the time surfer clicks random hyperlink.
- 10% of the time surfer types a random page.

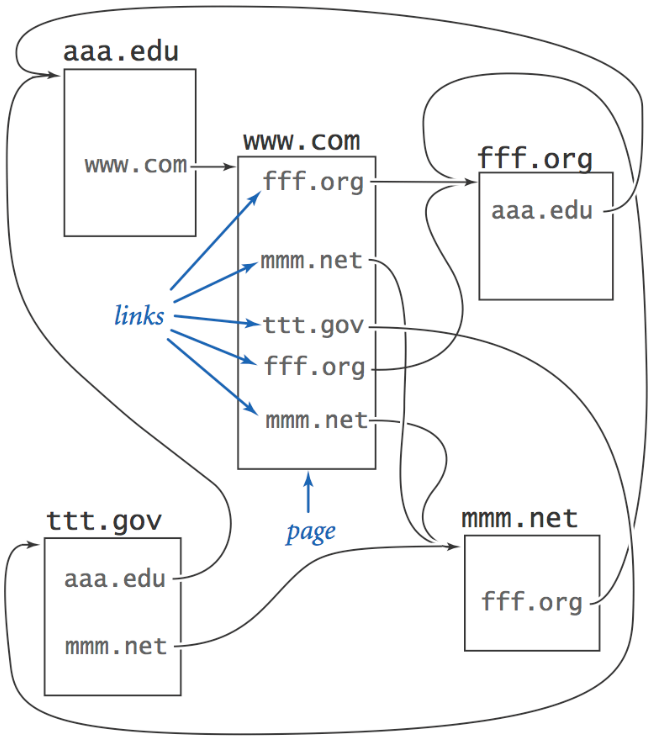
Caveat. Crude, but useful, web surfing model.

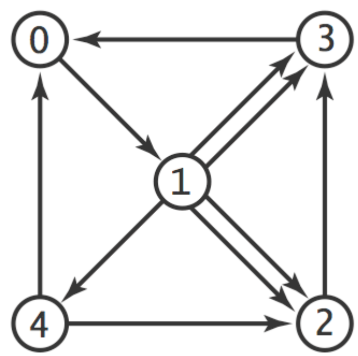
- No one chooses links with equal probability.
- No real potential to surf directly to each page on the web.
- The 90-10 breakdown is just a guess.
- It does not take the back button or bookmarks into account.
- We can only afford to work with a small sample of the web.
- …

Web Graph Input Format

Input format.

- N pages numbered 0 through N-1.
- Represent each hyperlink with a pair of integers.





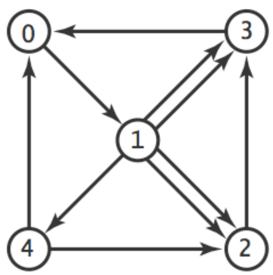

```
% more tiny.txt
5          ← N
0 1
1 2  1 2
1 3  1 3  1 4
2 3
3 0
4 0  4 2
```

links

Transition Matrix

Transition matrix. `p[i][j]`= prob. that surfer moves from page `i` to `j`.

input graph




```
5
0 1
1 2  1 2
1 3  1 3  1 4
2 3
3 0
4 0  4 2
```

link counts

0	1	0	0	0
0	0	2	2	1
0	0	0	1	0
1	0	0	0	0
1	0	1	0	0

degrees

1
5
1
1
2

leap probabilities

.02	.02	.02	.02	.02
.02	.02	.02	.02	.02
.02	.02	.02	.02	.02
.02	.02	.02	.02	.02
.02	.02	.02	.02	.02

+

link probabilities

0	.90	0	0	0
0	0	.36	.36	.18
0	0	0	.90	0
.90	0	0	0	0
.45	0	.45	0	0

=

transition matrix

.02	.92	.02	.02	.02
.02	.02	.38	.38	.20
.02	.02	.02	92	.02
.92	.02	.02	.02	.02
.47	.02	.47	.02	.02

surfer on page 1 goes to page 2 next 38% of the time

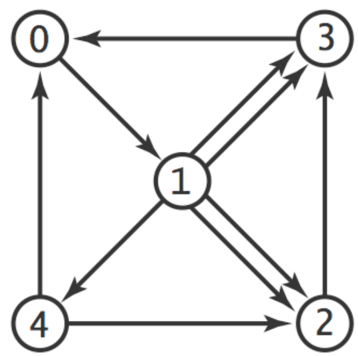
Web Graph to Transition Matrix

```
public class Transition {
   public static void main(String[] args) {
      int N            = StdIn.readInt(); // # number of pages
      int[][] counts   = new int[N][N];   // # links from page i to j
      int[] outDegree = new int[N];       // # links from page

      // accumulate link counts
      while (!StdIn.isEmpty()) {
         int i = StdIn.readInt();
         int j = StdIn.readInt();
         outDegree[i]++;
         counts[i][j]++;
      }

      // print transition matrix
      StdOut.println(N + " " + N);
      for (int i = 0; i < N; i++) {
         for (int j = 0; j < N; j++) {
            double p = .90*counts[i][j]/outDegree[i] + .10/N;
            StdOut.printf("%7.5f ", p);
         }
         StdOut.println();
      }
   }
}
```

Web Graph to Transition Matrix




```
% more tiny.txt
5
0 1
1 2  1 2
1 3  1 3  1 4
2 3
3 0
4 0  4 2
```

N

links

```
% java Transition < tiny.txt
5 5
 0.02000 0.92000 0.02000 0.02000 0.02000
 0.02000 0.02000 0.38000 0.38000 0.20000
 0.02000 0.02000 0.02000 0.92000 0.02000
 0.92000 0.02000 0.02000 0.02000 0.02000
 0.47000 0.02000 0.47000 0.02000 0.02000
```

Monte Carlo Simulation

Monte Carlo Simulation

Monte Carlo simulation.

- Surfer starts on page `0`.
- Repeatedly choose next page, according to transition matrix.
- Calculate how often surfer visits each page.

How? see next slide

	.02	.92	.02	.02	.02
	.02	.02	.38	.38	.20
	.02	.02	.02	92	.02
	.92	.02	.02	.02	.02
page	.47	.02	.47	.02	.02

transition matrix

Random Surfer

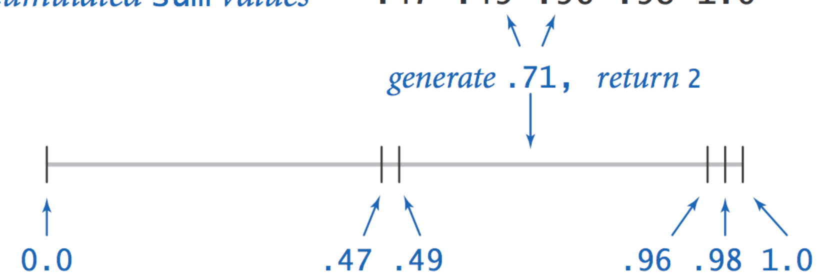
Random move. Surfer is on page `page`. How to choose next page `j`?

- Row `page` of transition matrix gives probabilities.
- Compute cumulative probabilities for row `page`.
- Generate random number `r` between `0.0` and `1.0`.
- Choose page `j` corresponding to interval where `r` lies.

	.02	.92	.02	.02	.02
	.02	.02	.38	.38	.20
	.02	.02	.02	92	.02
	.92	.02	.02	.02	.02
page	.47	.02	.47	.02	.02

transition matrix

`j`	0	1	2	3	4
probabilities `p[page][j]`	.47	.02	.47	.02	.02
cumulated `sum` *values*	.47	.49	.96	.98	1.0



Random Surfer

Random move. Surfer is on page `page`. How to choose next page `j`?

- Row `page` of transition matrix gives probabilities.
- Compute cumulative probabilities for row `page`.
- Generate random number `r` between `0.0` and `1.0`.
- Choose page `j` corresponding to interval where `r` lies.

```
// make one random move
double r = Math.random();
double sum = 0.0;
for (int j = 0; j < N; j++) {
   // find interval containing r
   sum += p[page][j];
   if (r < sum) { page = j; break; }
}
```

Random Surfer: Monte Carlo Simulation

```
public class RandomSurfer {
   public static void main(String[] args) {
      int T = Integer.parseInt(args[0]);      // number of moves
      int N = StdIn.readInt();                 // number of pages
      int page = 0;                            // current page
      double[][] p = new int[N][N];            // transition matrix

      // read in transition matrix
      ...

      // simulate random surfer and count page frequencies
      int[] freq = new int[N];
      for (int t = 0; t < T; t++) {

         // make one random move

         freq[page]++;
      }

      // print page ranks
      for (int i = 0; i < N; i++) {
         StdOut.printf("%8.5f", (double) freq[i] / T);
      }
      StdOut.println();
   }
}
```

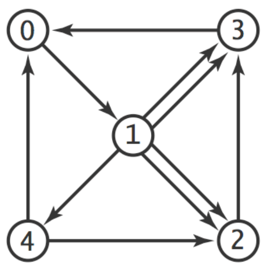
see previous slide

page rank

Mathematical Context

Convergence. For the random surfer model, the fraction of time the surfer spends on each page converges to a unique distribution, independent of the starting page.

"page rank"
"stationary distribution" of Markov chain
"principal eigenvector" of transition matrix



0 .27
1 .27
3 .25
2 .15
4 .07

$$\left[\frac{428{,}671}{1{,}570{,}055}, \frac{417{,}205}{1{,}570{,}055}, \frac{229{,}519}{1{,}570{,}055}, \frac{388{,}162}{1{,}570{,}055}, \frac{106{,}498}{1{,}570{,}055}\right]$$

Mixing a Markov Chain

The Power Method

Q. If the surfer starts on page `0`, what is the probability that surfer ends up on page `i` after one step?

A. First row of transition matrix.

first move

$$\underbrace{[\ 1.0\ \ 0.0\ \ 0.0\ \ 0.0\ \ 0.0\ ]}_{\texttt{rank[]}} * \underbrace{\begin{bmatrix} .02 & .92 & .02 & .02 & .02 \\ .02 & .02 & .38 & .38 & .20 \\ .02 & .02 & .02 & .92 & .02 \\ .92 & .02 & .02 & .02 & .02 \\ .47 & .02 & .47 & .02 & .02 \end{bmatrix}}_{\texttt{p[][]}} = \underbrace{[\ .02\ \ .92\ \ .02\ \ .02\ \ .02\ ]}_{\texttt{newRank[]}}$$

probabilities of surfing from 0 to i in one move

The Power Method

Q. If the surfer starts on page 0, what is the probability that surfer ends up on page i after two steps?

A. Matrix-vector multiplication.

first move

rank[] * p[][] = newRank[]

$$[\ 1.0\ \ 0.0\ \ 0.0\ \ 0.0\ \ 0.0\ ] * \begin{bmatrix} .02 & .92 & .02 & .02 & .02 \\ .02 & .02 & .38 & .38 & .20 \\ .02 & .02 & .02 & .92 & .02 \\ .92 & .02 & .02 & .02 & .02 \\ .47 & .02 & .47 & .02 & .02 \end{bmatrix} = [\ .02\ \ .92\ \ .02\ \ .02\ \ .02\ ]$$

probabilities of surfing from 0 to i in one move

second move

probabilities of surfing from 0 to i in one move

probabilities of surfing from i to 2 in one move

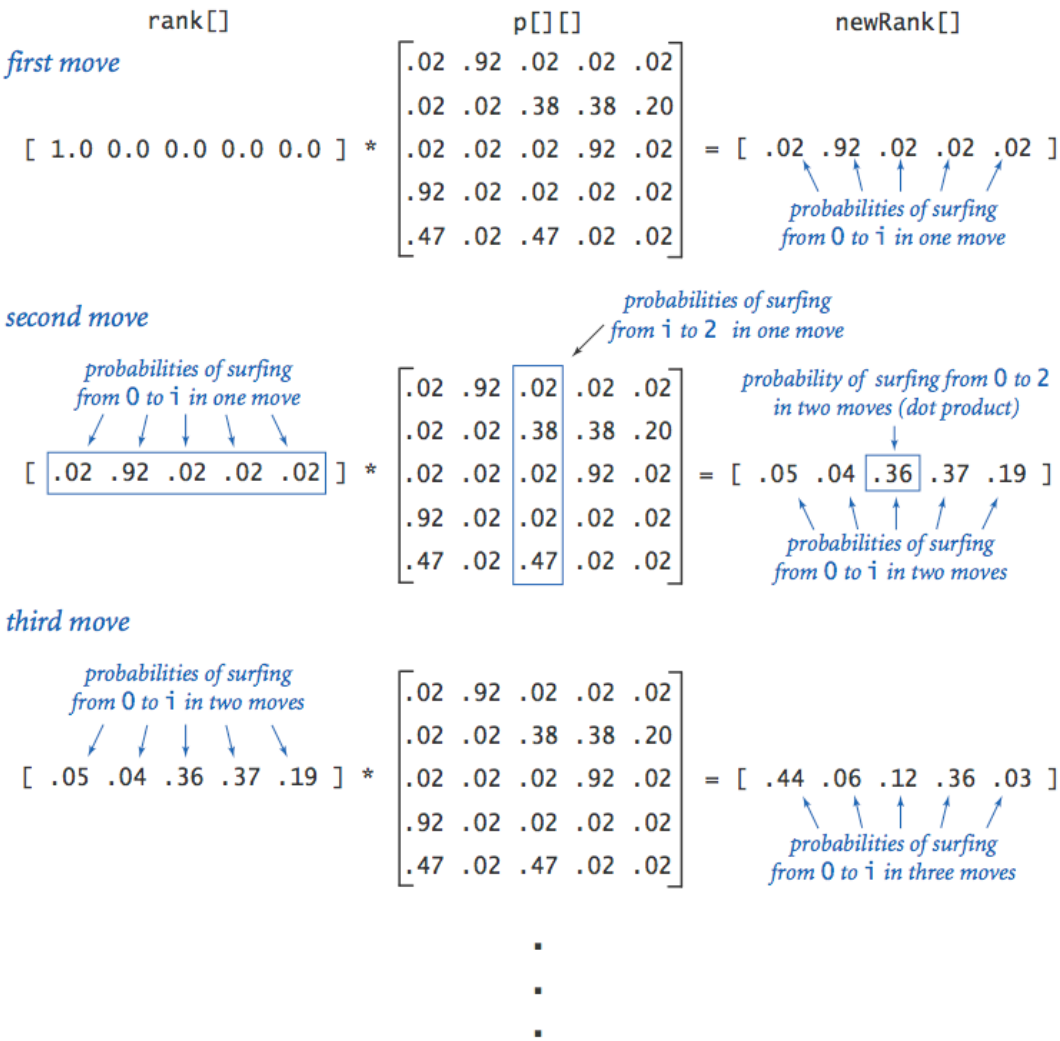
$$[\ .02\ \ .92\ \ .02\ \ .02\ \ .02\ ] * \begin{bmatrix} .02 & .92 & .02 & .02 & .02 \\ .02 & .02 & .38 & .38 & .20 \\ .02 & .02 & .02 & .92 & .02 \\ .92 & .02 & .02 & .02 & .02 \\ .47 & .02 & .47 & .02 & .02 \end{bmatrix} = [\ .05\ \ .04\ \ .36\ \ .37\ \ .19\ ]$$

probability of surfing from 0 to 2 in two moves (dot product)

probabilities of surfing from 0 to i in two moves

The Power Method

Power method. Repeat until page ranks converge.



Mathematical Context

Convergence. For the random surfer model, the power method iterates converge to a unique distribution, independent of the starting page.

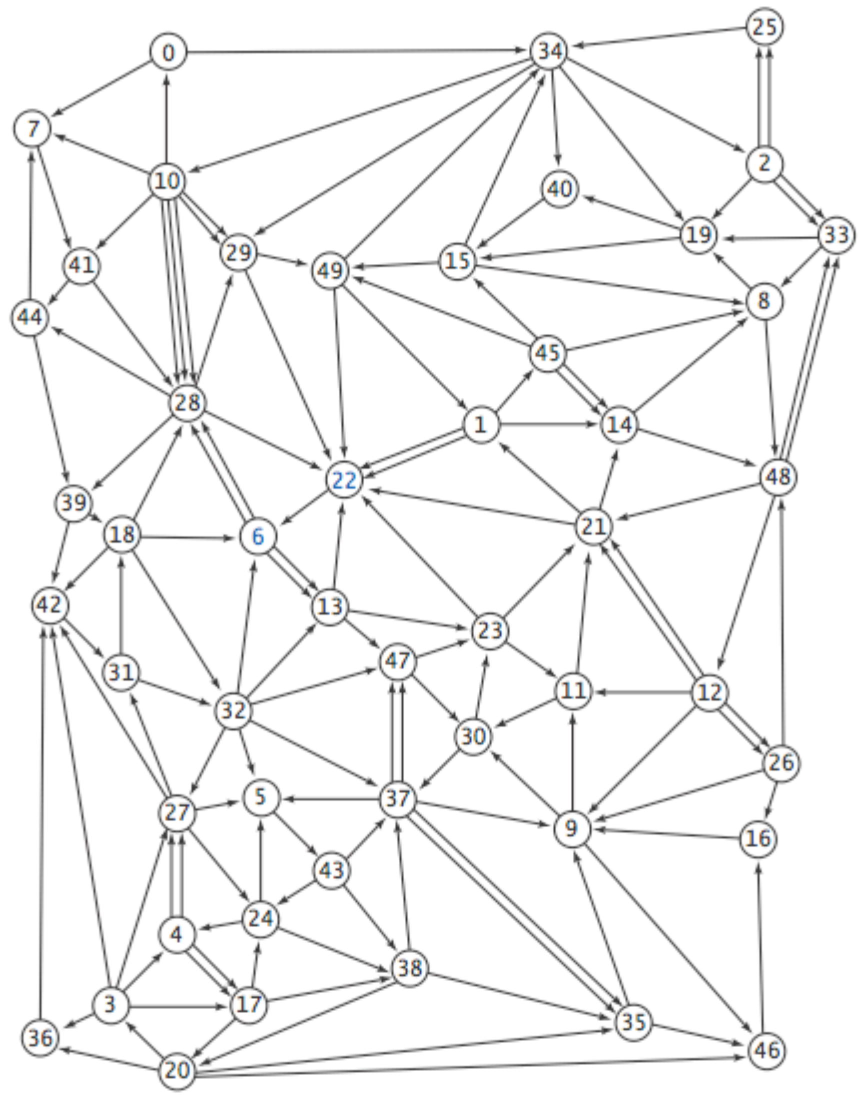
"page rank"
"stationary distribution" of Markov chain
"principal eigenvector" of transition matrix

20th move

probabilities of surfing from 0 *to* i *in 19 moves*

$$[\ .27\ .26\ .15\ .25\ .07\ ] * \begin{bmatrix} .02 & .92 & .02 & .02 & .02 \\ .02 & .02 & .38 & .38 & .20 \\ .02 & .02 & .02 & .92 & .02 \\ .92 & .02 & .02 & .02 & .02 \\ .47 & .02 & .47 & .02 & .02 \end{bmatrix} = [\ .27\ .26\ .15\ .25\ .07\ ]$$

probabilities of surfing from 0 *to* i *in 20 moves (steady state)*



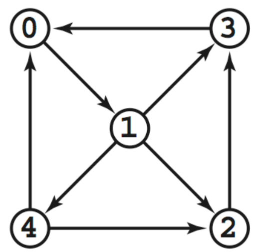
0	.002
1	.017
2	.009
3	.003
4	.006
5	.016
6	.066
7	.021
8	.017
9	.040
10	.002
11	.028
12	.006
13	.045
14	.018
15	.026
16	.023
17	.005
18	.023
19	.026
20	.004
21	.034
22	.063
23	.043
24	.011
25	.005
26	.006
27	.008
28	.037
29	.003
30	.037
31	.023
32	.018
33	.013
34	.024
35	.019
36	.003
37	.031
38	.012
39	.023
40	.017
41	.021
42	.021
43	.016
44	.023
45	.006
46	.023
47	.024
48	.019
49	.016

Page ranks with histogram for a larger example

Random Surfer: Scientific Challenges

Google's PageRank™ algorithm. [Sergey Brin and Larry Page, 1998]

- Rank importance of pages based on hyperlink structure of web, using 90-10 rule.
- Revolutionized access to world's information.



0	.269
1	.262
3	.227
2	.143
4	.099

Page ranks

Scientific challenges. Cope with 4 billion-by-4 billion matrix!

- Need data structures to enable computation.
- Need linear algebra to fully understand computation.