1.6 Case Study: Random Surfer
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.
World Wide Web


first Web server

Sir Tim Berners-Lee
Web Browser

Web browser. Killer application of the 1990s.
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. 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."
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.
Transition matrix. \( p[i][j] = \text{prob. that surfer moves from page } i \text{ to } j. \)
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 ← N
0 1
1 2 1 2
1 3 1 3 1 4
2 3
3 0
4 0 4 2

% 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.

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

Transition matrix:

\[
\begin{bmatrix}
0.02 & 0.92 & 0.02 & 0.02 & 0.02 \\
0.02 & 0.02 & 0.38 & 0.38 & 0.20 \\
0.02 & 0.02 & 0.02 & 0.92 & 0.02 \\
0.92 & 0.02 & 0.02 & 0.02 & 0.02 \\
0.47 & 0.02 & 0.47 & 0.02 & 0.02 \\
\end{bmatrix}
\]
Random Surfer

**Random move.** Surfer is on page \( p \). How to choose next page \( j \)?

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

![Transition matrix](image)

![Cumulative probabilities](image)
Random move. Surfer is on page $\text{page}$. How to choose next page $j$?

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

```java
// 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[\text{page}][j];
    if (r < sum) { \text{page} = j; break; }
}
```
Random Surfer: Monte Carlo Simulation

```java
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();
    }
}
```

- `StdOut.printf` is used to print page ranks with a precision of 8 decimal places.
- The transition matrix `p` is initialized within the `main` method.
- The program reads the number of moves `T` from the command line argument `args[0]` and the number of pages `N` from the standard input `StdIn.readInt()`.
- The current page is initialized as `page = 0`.
- A two-dimensional array `p` of type `int` is declared to represent the transition matrix.
- The program simulates the random surfer's movement by incrementing the frequency of the current page `freq[page]++` for `T` iterations.
- The page ranks are printed using `StdOut.printf` with a format string `"%8.5f"`, which specifies an 8-character field with 5 decimal places.
- The page rank is calculated as the frequency of each page divided by the total number of moves `T`.

The slide also mentions that the transition matrix is read in, which is not shown in the code snippet provided.
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.

\[
\begin{pmatrix}
428.671 & 417.205 & 229.519 & 388.162 & 106.498 \\
1,570,055 & 1,570,055 & 1,570,055 & 1,570,055 & 1,570,055
\end{pmatrix}
\]

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

\[
\begin{align*}
\text{rank}[] & \quad \text{p[][]} & \quad \text{newRank[]} \\
first\ move & \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} & \begin{bmatrix}
0.02 & 0.92 & 0.02 & 0.02 & 0.02 \\
0.02 & 0.02 & 0.38 & 0.38 & 0.20 \\
0.02 & 0.02 & 0.02 & 0.92 & 0.02 \\
0.02 & 0.02 & 0.02 & 0.02 & 0.20 \\
0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\
\end{bmatrix} & = \begin{bmatrix}
0.02 & 0.92 & 0.02 & 0.02 & 0.02 \\
\end{bmatrix}
\end{align*}
\]

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.

\[
\begin{align*}
\text{first move} & : & \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} \begin{bmatrix} 0.02 & 0.92 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.38 & 0.38 & 0.20 \\ 0.02 & 0.02 & 0.02 & 0.92 & 0.02 \\ 0.47 & 0.02 & 0.47 & 0.02 & 0.02 \end{bmatrix} = \begin{bmatrix} 0.02 & 0.92 & 0.02 & 0.02 & 0.02 \end{bmatrix}
\end{align*}
\]

\[
\text{probabilities of surfing from 0 to } i \text{ in one move}
\]

\[
\begin{align*}
\text{second move} & : & \begin{bmatrix} 0.02 & 0.92 & 0.02 & 0.02 & 0.02 \end{bmatrix} \begin{bmatrix} 0.02 & 0.92 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.38 & 0.38 & 0.20 \\ 0.02 & 0.02 & 0.02 & 0.92 & 0.02 \\ 0.47 & 0.02 & 0.47 & 0.02 & 0.02 \end{bmatrix} = \begin{bmatrix} 0.05 & 0.04 & 0.36 & 0.37 & 0.19 \end{bmatrix}
\end{align*}
\]

\[
\text{probabilities of surfing from 0 to } i \text{ in two moves (dot product)}
\]

\[
\text{probabilities of surfing from 0 to } i \text{ in two moves}
\]
The Power Method

Power method. Repeat until page ranks converge.

\[
\text{rank} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}
\]

**First move**

\[
\begin{bmatrix} .02 & .92 & .02 & .02 & .02 \\ .02 & .02 & .38 & .38 & .20 \\ .92 & .02 & .02 & .02 & .02 \\ .47 & .02 & .47 & .02 & .02 \end{bmatrix}
\]

\[
\text{newRank} = \begin{bmatrix} .02 & .92 & .02 & .02 & .02 \end{bmatrix}
\]

Probabilities of surfing from 0 to 1 in one move.

**Second move**

Probabilities of surfing from 0 to 1 in one move:

\[
\begin{bmatrix} .02 & .92 & .02 & .02 & .02 \end{bmatrix}
\]

Probabilities of surfing from 1 to 2 in one move:

\[
\begin{bmatrix} .02 & .92 & .02 & .02 & .02 \\ .02 & .02 & .38 & .38 & .20 \\ .92 & .02 & .02 & .02 & .02 \\ .47 & .02 & .47 & .02 & .02 \end{bmatrix}
\]

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

\[
\begin{bmatrix} .05 & .04 & .36 & .37 & .19 \end{bmatrix}
\]

Probabilities of surfing from 0 to 1 in two moves:

\[
\begin{bmatrix} .05 & .04 & .36 & .37 & .19 \end{bmatrix}
\]

**Third move**

Probabilities of surfing from 0 to 1 in two moves:

\[
\begin{bmatrix} .05 & .04 & .36 & .37 & .19 \end{bmatrix}
\]

Probabilities of surfing from 0 to 1 in three moves:

\[
\begin{bmatrix} .44 & .06 & .12 & .36 & .03 \end{bmatrix}
\]
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

<table>
<thead>
<tr>
<th>Probabilities of surfing from 0 to i in 19 moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>.27   .26   .15   .25   .07</td>
</tr>
</tbody>
</table>

* = 

\[
\begin{bmatrix}
.02 & .92 & .02 & .02 & .02 \\
.02 & .02 & .38 & .38 & .20 \\
.02 & .02 & .02 & .92 & .02 \\
.92 & .02 & .02 & .02 & .02 \\
.47 & .02 & .47 & .02 & .02 \\
\end{bmatrix}
\]

= 

<table>
<thead>
<tr>
<th>Probabilities of surfing from 0 to i in 20 moves (steady state)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.27   .26   .15   .25   .07</td>
</tr>
</tbody>
</table>

(steady state)
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.

![Page rank diagram]

Scientific challenges. Cope with 4 billion-by-4 billion matrix!
- Need data structures to enable computation.
- Need linear algebra to fully understand computation.