Google's pagerank.

Type "Markov chain" in google.
get 1,000,000 results
order them somehow. use pagerank.

Pagerank: don't consider content,
only the links.

rank of a page = the popularity of a page.
A page has a high rank if many
other pages link to it.
high ranked pages.
Monte Carlo method:

1,000,000 people surfing the web, each one starts at a random page. 
step: click one link at random. 
Do 1,000,000 steps each $\rightarrow 10^{12}$ pages visited 
For every page, count how many visitors there were $\rightarrow$ rank.
Markov chain.

states = web pages
arrows = links.

probability: same probability for each link.

Monte Carlo simulation → stationary vector.

\[ A \cdot \mathbf{v} = \mathbf{v} \]

\[ \mathbf{v} = \left[ \begin{array}{c} v_1 \\ v_2 \\ \vdots \\ v_{100\text{ billion}} \end{array} \right] \]

\[ v_i = \text{"rank" of a page} \]
Transition matrix: $A = \begin{bmatrix}
0 & 2/3 & 0 & 1/3 & 1/2 & 0 \\
1/3 & 0 & 0 & 0 & 0 & 0 \\
1/3 & 1/3 & 0 & 0 & 0 & 0 \\
1/3 & 1/3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}$

$\text{Find stationary vector } A\vec{v} = \vec{v}^*$

$\vec{v}^* \in \text{Nul} (A - I_6)$

$\begin{bmatrix}
v_1 \\
v_2 \\
v_3 \\
v_4 \\
v_5 \\
v_6 \\
\end{bmatrix}$

$v_i = \text{rank of page } i$. 
"Dangling nodes": a page with no links.
go to a random page with equal probability.

Find eigenvector \( \vec{v} \), \( \lambda = 1 \).

\[
A = a \times n, \quad n = 100 \text{ billion} = 10^n
\]

We know \( \lambda = 1 \) is eigenvalue.

\[ N(A - I_n) = \text{eigenspace } E_1 \]

\( A \) has many zeros, \( \lambda = 1 \) may have high multiplicity.

What to do: add more randomness.

At every step: either
prob \( x \rightarrow 1 \). Follow a link with equal prob.
prob \( 1 - x \rightarrow 2 \). Jump to a random page.

Transition matrix becomes:

\[
M = \alpha \cdot A + (1 - \alpha) \cdot B
\]

Same as before.

\[ M \text{ has all entries positive.} \]

\[ M \text{ has } \lambda = 1 \text{ eigenvalue with multiplicity 1.} \]

\[ \Rightarrow \text{unique eigenvector } \vec{v}. \]
How to find this $\tilde{v}$:

1. Solve $M \cdot \tilde{v} = \tilde{v}$
   
   $$(M - I_n) \cdot \tilde{v} = \tilde{0}.$$ 

   Row-reduction: $n^3$ steps.

   $n = 10^n, \quad n^3 = 10^{3n}$

   cannot do so many operations.

2. Use the power method.

   $\tilde{X}_0$ = random vector.

   $M_1$ not $\tilde{r}^2 = A \cdot \tilde{X}_0, \quad \tilde{X}_1 = A^2 \tilde{X}_0, \ldots, \quad \tilde{X}_k = A^{k-1} \tilde{X}_0$

   $\tilde{X}_k \xrightarrow{k \to \infty} \tilde{v}$

   Brain < Page: 50-100 steps, $\tilde{X}_k$ close enough.

   $\tilde{v} \approx \tilde{X}_{100}$
How to multiply?

\[ A \cdot X_0 \quad \text{\(n^2\) steps} \]
\[ n \times n \quad n \times 1 \quad n=10^n, \quad n^2 = 10^{2n} \]

\[ M = \alpha A + (1-\alpha) B \]

\(A\) is sparse, has many zeros.

every page has \(\approx 10\) links
every column of \(A\) has \(\approx 10\) nonzero entries.

\(A\) has \(10 \cdot n\) nonzero entries.

\(A \cdot X\) takes \(10 \cdot n\) steps \(10^{12}\)

\(B \cdot X\) easy. \(B X = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \alpha = \text{average of } x_1, x_2, \ldots, x_{10^n}\)

Conclusion: \(M \cdot X\) can be done reasonably fast

Page & Brain: it takes \(\approx 2\) days to compute \(X_{10^n}\)

Repeat this every month.

Use \(\alpha \approx .85\).

\textit{Storage: matrix }A\textit{ has }10 \cdot n \approx 10^{12}\textit{ nonzero entries.}