Google produced search results based on the connectivity of webpages.
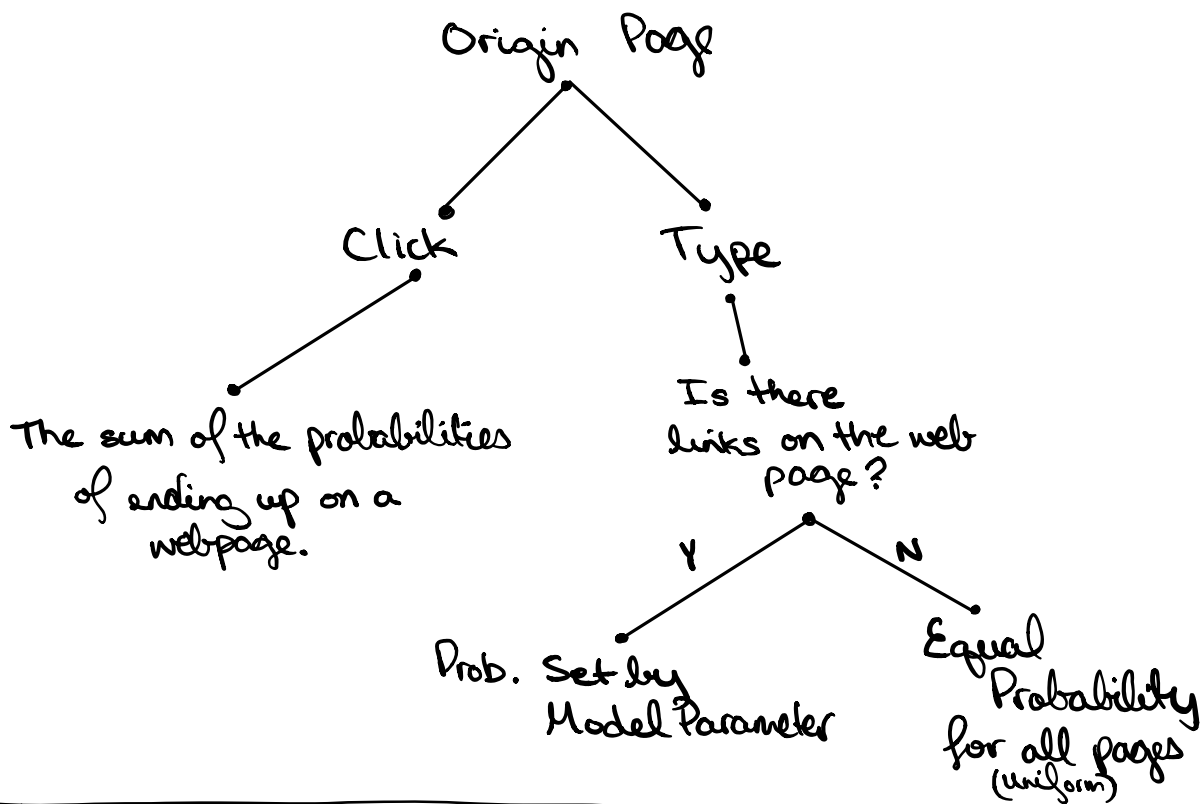
A web-surfer navigates the web through these actions.

① Clicking on webpage links

② Navigating to a new page by typing.

A workflow would look like.

Origin Page

Click

Type

The sum of the probabilities of ending up on a webpage.

Is there links on the web page?

Y → Prob. Set by Model Parameter

N → Equal Probability for all pages (uniform)
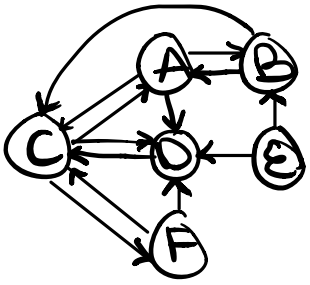
Inherent in the design of PageRank algorithm is.

- the underlying connectivity is reflective a page's search rank.

- second is that important websites link to other important websites

A Graph of Page Connectivity

Each node represents a page and each <mark>directed edge is</mark> a link that connects the source webpage to the target.

<mark>Page surfers can</mark> also <mark>teleport</mark> (by typing).

<mark>Page surfers</mark> are modelled as <mark>making random walks</mark> choosing edges travel with a uniform probability dictated by the number of outbound edges of the current node.

Given the random walks, the graph, and a random initial state of users in the network — the traffic through those cycles will converge. Users can then be found predictably on certain pages with certain ratios/densities.

The number of page surfers on each page once the network activity converges is expressed by the vector $\underline{r}$.

$$r = \begin{bmatrix} r_A \\ r_B \\ r_C \\ r_D \\ r_E \\ r_F \end{bmatrix}$$

The number of users on each website can be generalized with the expression

$$r^{i+1} = L r^i$$

This models the number of users per page at minute i. It holds true before and after convergence.

After convergence $r^{(i+1)} = r^i$; can instead write

$$L r = r$$

<mark>This is an eigenequation for matrix L with eigenvalue 1, determined by the probabilistic structure of the matrix L.</mark> (i.e network has no loss & const. traffic) ↳ normalized pr.

The matrix L.

$$\begin{bmatrix} L_{A\to A} & L_{B\to A} & L_{C\to A} & L_{D\to A} & L_{E\to A} & L_{F\to A} \\ L_{A\to B} & L_{B\to B} & L_{C\to B} & L_{D\to B} & L_{E\to B} & L_{F\to B} \\ L_{A\to C} & L_{B\to C} & L_{C\to C} & L_{D\to C} & L_{E\to C} & L_{F\to C} \\ L_{A\to D} & L_{B\to D} & L_{C\to D} & L_{D\to D} & L_{E\to D} & L_{F\to D} \\ L_{A\to E} & L_{B\to E} & L_{C\to E} & L_{D\to E} & L_{E\to E} & L_{F\to E} \\ L_{A\to F} & L_{B\to F} & L_{C\to F} & L_{D\to F} & L_{E\to F} & L_{F\to F} \end{bmatrix}$$

Each column represent the (sum=1) Probability leaving a page for another page. Each row represents how likely you are to enter a website from another (doesn't need to sum=1).

Can try to calculate each of the eigenvectors of the network graph; however this isn't scalable for high dimension systems. Instead power iteration can be used — the power series.

To add the opportunity for walks to jump/teleport to another vertice.
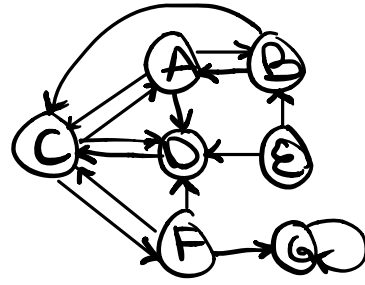The model can be adjusted to include additional variables

$$M = dL + \frac{1-d}{n} J$$, d is the prob of clicking a link

1-d is the prob of typing

$J_{n \times n}$ is a matrix of 1 elements computing to equal probabilities of leaving any page for any other page.

Typically 1-d is small.

The power iteration applied to the network graph will yield G as the highest rank page. This is due to an oversimplified model which only allows for walks — once a path enters the reflexive G node the walk only has the opportunity to revisit the G node.