Rough incomplete sketch of history of the WorldWideWeb
(Note first that WorldWideWeb $\neq$ Internet, WWW is rather a protocol and set of resources that is layered on top of the pre-existing internet.)

1945 Memex, V.Bush. one of many hypertext forerunners

1989 Berners-Lee, CERN, global hyperspace idea

1990 WorldWideweb.app on NeXT computer

1991 CERN server/client released in summer of '91, http protocol and html coded pages, also linemode browser (lynx)

1991–1994 growth, mainly in Europe. First U.S. website at Stanford Linear Accelerator Center (1992). Spring '93 Mosaic client (NCSA), added in-line graphics, also produced its own version of httpd server software. CERN still maintained a list of "all webservers in world".

Early '94: crawlers like "jumpstation", and "WWW Worm" (McBryan 1994, 110,000 pages and 1500 queries per day in Mar/Apr '94).

Nov 97: 2M-100M docs (expected 1B by 2000). Altavista handled 20M queries/day. 2000: expected 100's of million/day .

(Actual 2004 Google was 4.2B pages. In 2005 Yahoo and Google each claimed to have indexed upwards of 15B pages, then stopped posting their claimed counts.)

If 10 times the number of pages meant every query brings up 10 times as many results to sort through, then search engine methodology doesn't scale with size of web. Perhaps it only worked because the amount of material on the web was still so *small*? But it turned out there's a set of heuristics for ordering the search results, so that the desired page is frequently ranked in the top ten, and it doesn't matter that there are many thousands of other pages retrieved.

The Page Rank methodology stems from a long history of citation analysis, where a "link" is as well some signal of recommendation (or popularity). It can computed as a property of the entire graph, independent of any particular query, and hence is efficient for serving a large volume of queries. The Markov process it embodies was also not new, but was applied in a particularly powerful way, again demonstrating the unexpected power of simple algorithms and ample computing power applied to massive datasets.

Brin/Page used "Page Rank" to measure the importance of page, and help to order the search results. The page rank r_j of page j is determined self-consistently by the equation

$$r_j = \frac{p}{n} + (1-p) \sum_{i|i\to j} \frac{r_i}{d_i} \ , \tag{1}$$

where p is a number between 0 and 1 (originally taken to be .15), the sum on i is over pages i pointing to j, and d_i is the outgoing degree of page i. Intuitively, we see that pages with high "Page rank" r_i that have low d_i, i.e., don't point to too many other pages, convey the most page rank to page j. Formally, the above describes the steady state of a diffusion process, where r_i can be interpreted as the probability that one lands at page j by navigating according to the following algorithm: with probability $1-p$ one goes at random to any of the pages j pointed to by page i, and with probability p one instead jumps at random to any page anywhere on the web (so one doesn't get caught in a single subcomponent).

This can also be stated as an eigenvector problem. Recall the incidence matrix A is defined by $A_{ij} = 1$ if i points to j and otherwise $A_{ij} = 0$; A matrix B giving the transition probability from page i to page j can be constructed in terms of the incidence matrix A as

$$B_{ij} = \frac{p}{n} O_{ij} + (1-p)\frac{1}{d_i} A_{ij} \tag{2}$$

where n = total # of pages, d_i is the outdegree of node i, and $O_{ij} = 1(\forall i,j)$ is a matrix of all ones (i.e., $O = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \dots & & & \\ 1 & 1 & \dots & 1 \end{pmatrix}$). The matrix eigenvector relation

$$\vec{r} = B^T \vec{r} \tag{3}$$

is then seen to be equivalent to eqn. (1), where B^T is the transpose of B as defined in (2), and we have assumed that $\vec{r}$ is normalized as a probability, so that $\sum_i r_i \, O_{ij} = \sum_i r_i = 1$.

By the Perron-Frobenius theorem [details to be added], the matrix B has a unique principal eigenvector, corresponding to largest eigenvalue, and its components are all positive. (Since $\sum_j A_{ij}/d_i = 1$, we find $\sum_j B_{ij} = p + (1-p) = 1$ and B is normalized such that its principal eigenvalue is 1.) Thus eqn. (3) always has a solution.

To calculate the rank of all the pages, the crawler first visits as many pages as possible, and calculates the link structure of the web graph. Calculating eigenvectors of enormous matrices can be painful, but in the case of the principal eigenvector there's a simple method. Recall that any $N \times N$ matrix M has N eigenvectors $\vec{v}_{(i)}$ that satisfy

$$M\vec{v}_{(i)} = \lambda_i \vec{v}_{(i)} \ . \tag{4}$$

These eigenvectors form a basis set, meaning that any other N-dimensional vector $\vec{w}$ can be expressed as a linear combination $\vec{w} = \sum_{i=1}^{N} \alpha_i \vec{v}_{(i)}$, with α_i constants.

Now take the largest eigenvalue to be λ_1, and consider applying M a total of n times

$$M^n \vec{w} = \sum_{i=1}^{N} \alpha_i \lambda_i^n \vec{v}_{(i)}$$

where we have used eqn (4). As n gets large we see that the term with the largest eigenvalue will dominate,

$$\frac{1}{\lambda_1^n} M^n \vec{w} = \alpha_1 \vec{v}_{(1)} + \sum_{i=2}^{N} \alpha_i \Big(\frac{\lambda_i}{\lambda_1}\Big)^n \vec{v}_{(i)} \approx \alpha_1 \vec{v}_{(1)} \ ,$$

since $\lambda_i/\lambda_1 < 1 \ \forall i \neq 1$. That means we can determine the principal eigenvector simply by applying a matrix M sufficiently many times to any vector which has non-zero dot product with the principal eigenvector (i.e., non-vanishing α_1 in the above) — this procedure effectively projects to the eigenvector of interest.

Some simple 2×2 examples show how this works. The matrix $M = \frac{1}{3}\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ has right eigenvectors $\vec{v}_1 = \binom{1}{1}$ and $\vec{v}_2 = \binom{1}{-1}$, with eigenvalues $\lambda_1 = 1$ and $\lambda_2 = 1/3$. Consider acting on $\vec{w} = \vec{v}_1 + \vec{v}_2 = \binom{2}{0}$ with M many times:

$$M\vec{w} = \frac{1}{3}\begin{pmatrix} 4 \\ 2 \end{pmatrix}, \quad M^2\vec{w} = \frac{1}{3^2}\begin{pmatrix} 10 \\ 8 \end{pmatrix}, \quad M^3\vec{w} = \frac{1}{3^3}\begin{pmatrix} 28 \\ 26 \end{pmatrix}, \quad \ldots$$
$$M^n\vec{w} = \frac{1}{3^n}\begin{pmatrix} 3^n + 1 \\ 3^n - 1 \end{pmatrix} = \vec{v}_1 + \frac{1}{3^n}\vec{v}_2 \ .$$

We see that as n grows larger, multiplying by M^n acts to project out the unique principle eigenvector $\vec{v}_1$ corresponding to the largest eigenvalue, and the components of $\vec{v}_1 = \binom{1}{1}$ are all non-negative. This will not be the case for matrices that are either not irreducible or have negative entries. For example, $M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ is not irreducible and has two eigenvectors $\binom{1}{0}$ and $\binom{0}{1}$ both corresponding to eigenvalue $\lambda = 1$. And the matrix $M = \begin{pmatrix} 2 & -1 \\ -3 & 0 \end{pmatrix}$ has principle eigenvector $\vec{v}_1 = \binom{1}{-1}$ with $\lambda = 3$, but $\vec{v}_1$ has negative components. (The other eigenvector $v_2 = \binom{1}{3}$ has eigenvalue $\lambda = -1$.)

Now consider the case of $n = 4$ web pages linked as in the figure below. We can go back to eqn. (1) and calculate r_i iteratively as follows: begin with $r_i^{(0)} = 1/n$, the vector of equal probability of being at any site. Its dot product with the principal eigenvector is guaranteed to be non-vanishing, since the principal eigenvector has only positive components. We calculate iteratively

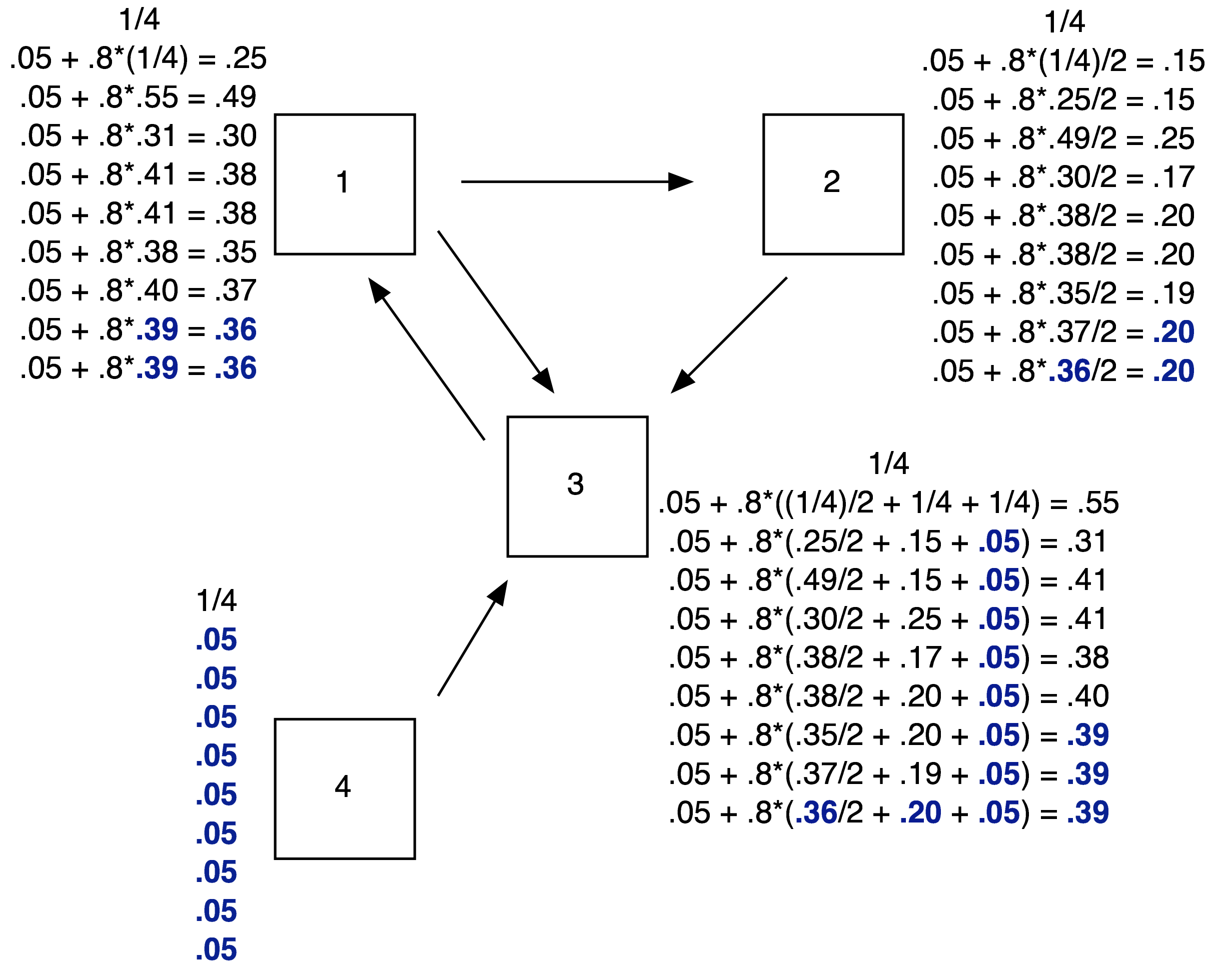
$$r_j^{(1)} = \frac{p}{n} + (1-p)\sum_{i|i\to j} \frac{r_i^{(0)}}{d_i}$$

$$r_j^{(2)} = \frac{p}{n} + (1-p)\sum_{i|i\to j} \frac{r_i^{(1)}}{d_i}$$

$$\vdots$$

$$r_j^{(n)} = \frac{p}{n} + (1-p)\sum_{i|i\to j} \frac{r_i^{(n-1)}}{d_i}\ .$$

The figure shows the result of these iterations for $p = .2$ (and $n = 4$):



In the first iteration, the rank $r_1 = 1/4$ is multiplied by .8 and then shared between pages 2 and 3, giving .1 to each. Since that is the only page pointing to 2, we see that $r_2^{(1)} = .05 + .1 = .15$. For the four components of $\vec{r}$ corresponding to pages $(1, 2, 3, 4)$, we see that starting from the vector (1/4, 1/4, 1/4, 1/4), after 9 iterations the probabilities converge to approximately (.36, .20, .39, .05). Pages 1 and 3 are roughly the same, page 2 is roughly half as probable as those two, and page 4 is reachable only by random jumps hence has the lowest probability. Page 3 has slightly higher probability than page 1 due to to the additional incoming link from 4.

This graph is also small and simple enough to work with the transition matrix directly. First we write the adjacency matrix A_{ij} for this graph (where $A_{ij} = 1$ iff page i points to page j), $A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$. Since the graph has a single connected component, for additional simplicity we'll just take $p = 0$, so from any page there's only a probability of jumping to one of the pages to which it points. Then the matrix B_{ij} is equal to A_{ij}/d_i (where d_i is the outgoing degree of page i), and satisfies $B_{ij} = A_{ij}/d_i = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$.

If we again start from an equal probability distribution, in which all four pages have $r_i^{(0)} = 1/4$, and iterate the equation $r_j^{(n)} = \sum_{i|i \to j} r_i^{(n-1)}/d_i$ two times (i.e., for $n = 1, 2$) to approximate the solution for the page rank, this is equivalent to the effect of acting with the matrix B^T with $p = 0$ twice on $\vec{r}^{(0)}$:

$$(B^T)^2 \vec{r} = \begin{pmatrix} 1/2 & 1 & 0 & 1 \\ 0 & 0 & 1/2 & 0 \\ 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix} = \begin{pmatrix} r_1/2 + r_2 + r_4 \\ r_3/2 \\ r_1/2 + r_3/2 \\ 0 \end{pmatrix}$$

so $(B^T)^2 \vec{r}^{(0)} = (5/8, 1/8, 1/4, 0)$.

The exact page rank for the four pages in the figure for $p = 0$ can be guessed by staring long enough at them, or equivalently by directly calculating the principal eigenvector $\vec{r} = (r_1, r_2, r_3, r_4)$ of the matrix, with components normalized as probabilities, $\sum_{i=1}^{4} r_i = 1$. The components of the eigenvalue equation $B^T \vec{r} = \lambda \vec{r}$ are

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1/2 & 0 & 0 & 0 \\ 1/2 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix} = \begin{pmatrix} r_3 \\ r_1/2 \\ r_1/2 + r_2 + r_4 \\ 0 \end{pmatrix} = \lambda \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix}$$

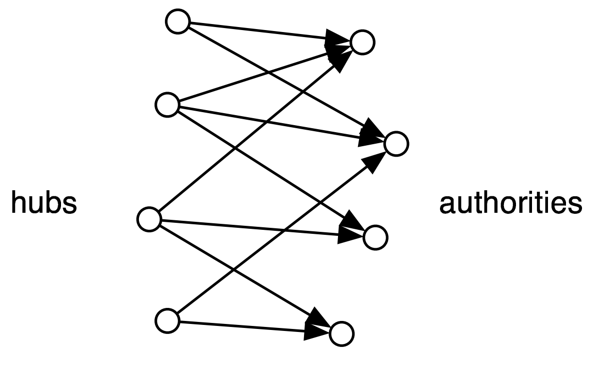
implying the four equations

$$r_3 = \lambda r_1 \quad r_1/2 = \lambda r_2 \quad r_1/2 + r_2 + r_4 = \lambda r_3 \quad 0 = \lambda r_4 \ .$$

The principal eigenvector, with $\lambda = 1$, hence satisfies $r_3 = r_1 = 2r_2$ and $r_4 = 0$, hence the eigenvector, normalized as a probability is $\vec{r} = (2/5, 1/5, 2/5, 0)$. (The other three eigenvalues are $\lambda = 0, (-1 \pm i)/2$, but only the largest one above is relevant here.) We see that page 4 has zero probability for $p = 0$, since there's no path to it. Pages 1 and 3 have the same steady state probability, since page 3 conveys all of its page rank to page 1, which gives half directly back to page 3, and the other half flows back to page 3 via page 2. Page 3 only has the highest page rank when we take p non-zero as shown in the figure, so that it receives extra probability from page 4.

(Incomplete notes here...) In addition to link analysis, other heuristics such as location information are used to order the search results. If the query term is a single word, then does it occur in the title of the page (metadata), or in the anchor text pointing to the page, or in the URL itself, or if it's in the text, is it in a large or small font, how many times does it occur, how near to the beginning of the page does it occur? For multiword queries, how close together do the query terms appear in the page (proximity increases relevance). (Note that sometimes the anchor text describing a page <a href="... g.jpg">giraffe</a> gives a more accurate description of page than the page itself, though can also lead to deliberate collaborative manipulation, see "google bombing")

Page rank works on the basis of link analysis: "important" web pages are self-consistently defined as ones pointed to by other "important" pages. But there is potentially more structure, for example pages with large in-degree or large out-degree might play some special role. This possibility was exploited by the HITS ("hypertext induced topic search") algorithm (Kleinberg, 1998), proposed at roughly the same time as the Page rank algorithm. Sites with large in-degree acquire some implicit sense of authority, by virtue of being pointed to by so many other sites. They are termed "authorities", and the sites with large out-degree that point to the most important authorities are termed "hubs", as schematically depicted below:



In this framework, every page has both an authority weight a_i and a hub weight h_i, defined to satisfy

$$a_j = \sum_{i|i\to j} h_i \ , \qquad h_i = \sum_{j|i\to j} a_j \ ,$$

i.e., the authority weight of a site j is given by the sum of the hub weights of sites i that point to the site, and the hub weight of a site i is given by the sum of the authority weights of the sites j to which it points. In terms of the incidence matrix A, these can be written

$$\vec{a} = A^T \vec{h} \ , \qquad \vec{h} = A\vec{a} \ .$$

Now we could imagine starting with some trial forms of the hub and authorities weights, $h_i^{(0)}$ and $a_i^{(0)}$, and iterating the above equation:

$$a_j^{(1)} = \sum_{i|i\to j} h_i^{(0)} \ , \qquad h_i^{(1)} = \sum_{j|i\to j} a_j^{(0)}$$

to provide more refined guesses $h_i^{(1)}$ and $a_i^{(1)}$, and then continuing. In matrix form, the j^{th} such iteration can be written

$$\vec{a}_{(j)} = A^T \vec{h}_{(j-1)} \ , \qquad \vec{h}_{(j)} = A\vec{a}_{(j-1)} \ .$$

Note that the result of two iterations can be written $\vec{a}_{(j)} = A^T A\, \vec{a}_{(j-2)}$, $\vec{h}_{(j)} = AA^T\, \vec{h}_{(j-2)}$, so the result of $2n$ such iterations is

$$\vec{a}_{(2n)} = (A^T A)^n \vec{a}_{(0)} \ , \qquad \vec{h}_{(2n)} = (AA^T)^n \vec{h}_{(0)} \ .$$

The matrices $A^T A$ and AA^T are symmetric with non-negative entries that sum to 1 along any row, so for a suitably chosen subset of nodes such that they're irreducible, the Perron-Frobenius theorem will apply. In that case, the above iterative procedure will converge, and we see that the authority weights a_i will be given by the components of the principal eigenvector of $A^T A$, and the hub weights h_i by the components of the principal eigenvector of AA^T.

In the original proposal, the algorithm was applied as follows: first retrieve all pages from the web that match the query terms. Then expand this set by including as well all pages that link to and are linked from the pages in the original set. (This might involve a cut-off for the number of pages any given page can bring in so that the set does not grow unmanageably large.) Then the above algorithm can be used to determine the authority weights for this set and the pages with the highest authority weights can be returned first in response to the query. Note that while this procedure might produce more refined results that the Page rank algorithm, it is computationally less efficient since it requires calculating a set of weights dynamically in response to each query. The Page rank algorithm, on the other hand, calculates the weights once and for all for the entire directed graph of the web, and they can then be used as a static feature to order the search results returned in response to any query.