第七章

正负矩阵

引导问题：PageRank

Page**Rank(网页级别）：2001年9月被授予美国专利，专利人是*****Google*****创始人之一拉里·佩奇（Larry *Page*）。因此，*Page*Rank里的*page*不是指网页，而是指佩奇，即这个等级方法是以佩奇的名字来命名的。**

pagerank 模型模拟的是一个用户在互联网上浏览到每个网页的概率。

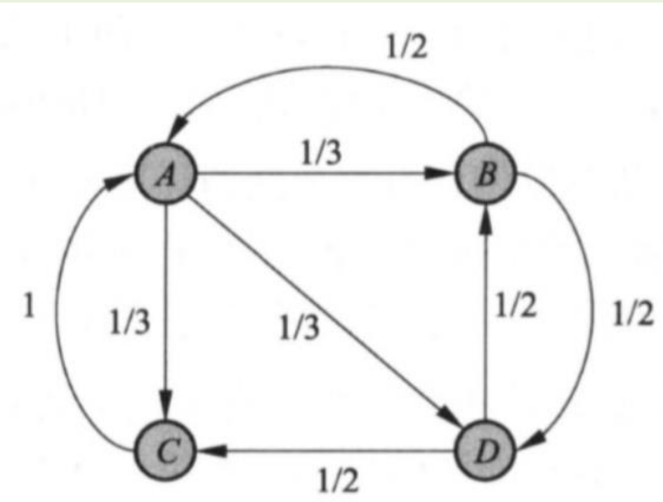
Pagerank基本思想：

数量假设：一个页面越被其他页面链接，说明他越重要

质量假设：越是被高质量页面链接，说明该页面越重要。

在一定条件下，极限情况访问每个结点的概率收敛到平稳分布，这时各个结点的平稳概率值就是其**PageRank**值，表示结点的重要度。

引导问题：PageRank



概率转移

$$M=\begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

A 

C 

B 

D 

$$R_{t+1}=MR_t$$

$$\lim_{t\to\infty} M^t R_0 = R$$

提纲

- 1. 矩阵不等式
- 2. 正的矩阵
- 3. 非负矩阵
- 4. 随机矩阵
- 5. PageRank

1.1 不等式

设$A=[a_{ij}]\in M_{m\times n}(R)$以及$B=[b_{ij}]\in M_{m\times n}(R)$，记

(1)$A\geq 0$，如果所有$a_{ij}\geq 0$；

以及$A>0$，如果所有$a_{ij}>0$

(2) $A\geq B$，如果$A-B\geq 0$；

以及$A>B$，如果$A-B>0$

如果$A\geq 0$，矩阵A是非负的矩阵；
如果$A>0$，矩阵A是正的矩阵。

【定义】$|A|=[|a_{ij}|]$，矩阵按元素逐个取绝对值

1.2 性质

设给定$A=[a_{ij}]\in M_n$以及$x=[x_i]\in F^n$

(1)$|Ax|\le|A||x|$

(2)假设A是非负的且有一行是正的，如果$|Ax|=A|x|$，那么存在一个实数$\theta\in[0,2\pi)$，使得$e^{-i\theta}x=|x|$

(3)假设x是正的，如果$|Ax|=A|x|$，那么$|A|=A$。

1.2 性质

设给定A，$B \in M_n$

(1) $|AB| \le |A||B|$

(2) $|A^m| \le |A|^m$

(3) 如果$0 \le A \le B$，那么$0 \le A^m \le B^m$

(4)如果$|A| \le |B|$，那么$\|A\| \le \|B\|$

(5) $\|A\| = \||A|\|$

1.2 性质

【定理】设$A=[a_{ij}]\in M_n$是非负的，那么就有

$$\min_{1\le i\le n}\sum_{j=1}^{n}a_{ij}\le\rho(A)\le\max_{1\le i\le n}\sum_{j=1}^{n}a_{ij}$$

以及

$$\min_{1\le j\le n}\sum_{i=1}^{n}a_{ij}\le\rho(A)\le\max_{1\le j\le n}\sum_{i=1}^{n}a_{ij}$$

非负矩阵的最大的行（列）和是谱半径上界;
非负矩阵的最小的行（列）和是谱半径下界。

1.2 性质

【定理】设$A=[a_{ij}]\in M_n$是非负的，那么对任何正的向量 $x=[x_i]\in R^n$ 有

$$\min_{1\le i\le n}\frac{1}{x_i}\sum_{j=1}^{n}a_{ij}x_j\le\rho(A)\le\max_{1\le i\le n}\frac{1}{x_i}\sum_{j=1}^{n}a_{ij}x_j$$

以及

$$\min_{1\le j\le n}x_j\sum_{i=1}^{n}\frac{a_{ij}}{x_i}\le\rho(A)\le\max_{1\le j\le n}x_j\sum_{i=1}^{n}\frac{a_{ij}}{x_i}$$

提纲

- 1. 矩阵不等式
- **2. 正的矩阵**
- 3. 非负矩阵
- 4. 随机矩阵
- 5. PageRank

2.1 基本性质

【定理】 如果 $A\in M_n$ 是正的，则存在正的向量 x 与 y，使得

$$Ax=\rho(A)x$$

以及

$$y^TA=\rho(A)y^T$$

2.1 基本性质

【定理】设$A\in M_n$是正的。如果λ是A的一个特征值，且$\lambda\neq\rho(A)$，那么

$$|\lambda|<\rho(A)$$

【定理】如果$A\in M_n$是正的，那么$\rho(A)$作为矩阵A的特征值的几何重数为1。

$\rho(A)$：Perron根；特征向量：Perron向量

2.2 Perron

【定理】设 $A\in M_n$是正的，那么

(1)$\rho(A)>0$

(2)$\rho(A)$是A的代数重数为1的单重特征根

(3)存在唯一的实向量$x=[x_i]$，使得$Ax=\rho(A)x$，以及$x_1+x_2+\ldots+x_n=1$，且这个向量是正的

(4)存在唯一的实向量 $y=[y_i]$，使得$y^TA=\rho(A)y^T$，以及$x_1y_1+x_2y_2+\ldots+x_ny_n=1$，且这个向量是正的

(5) $\lim\limits_{m\to\infty}(\rho(A)^{-1}A)^m=xy^T$

2.3 其他性质

【樊畿】设$A=[a_{ij}]\in M_n$。假设$B=[b_{ij}]\in M_n$是非负的，且对所有的$i\neq j$都有$b_{ij}\geq |a_{ij}|$。那么A的每个特征值z都在n个圆盘的并集之中

$$\bigcup_{i=1}^{n}\{z\in \mathbf{C}:|z-a_{ii}|\leq \rho(B)-b_{ii}\}$$

特别地，A是非奇异的，如果对所有$i=1,\ldots,n$都有$|a_{ii}|>\rho(B)-b_{ii}$。

提纲

- 1. 矩阵不等式
- 2. 正的矩阵
- **3. 非负矩阵**
- 4. 随机矩阵
- 5. PageRank

3.1 基本性质

【定理】如果$A\in M_n$是非负的，那么$\rho(A)$是A的一个特征值，且存在一个非负的非零向量x，使得 $Ax=\rho(A)\,x$。

3.1 基本性质

- 【定理】如果$A\in M_n$是非负的，实向量$x\in R^n$是非负的且是非零的。如果$a\in R$且$Ax\geq ax$，那么$\rho(A)\geq a$。

3.1 基本性质

【推论】如果 $\boldsymbol{A}\in \boldsymbol{M}_n$ 是非负的，那么

$$\rho(A)=\max_{\substack{\boldsymbol{x}\geq 0\\ \boldsymbol{x}\neq 0}}\min_{\substack{1\leq i\leq n\\ x_i\neq 0}}\frac{1}{x_i}\sum_{j=1}^{n}a_{ij}x_j$$

3.1 基本性质

【定理】 如果$A\in M_n$是非负的。假设存在一个正的向量x以及一个非负的实数λ，使得或者有$Ax=\lambda x$，或者有$x^TA=\lambda x^T$，那么就有$\lambda=\rho(A)$

证明：设$\mathbf{x}=[x_i]\in\mathbf{R}^n$且$A\mathbf{x}=\lambda\mathbf{x}$。设$D=\mathrm{diag}(x_1,\ldots,x_n)$，

$$B=D^{-1}AD$$

$$\boldsymbol{B}\boldsymbol{e}=D^{-1}AD\boldsymbol{e}=D^{-1}A\mathbf{x}=\lambda D^{-1}\mathbf{x}=\lambda\boldsymbol{e}$$

$$\Longrightarrow \quad \rho(B)=\lambda \quad \Longrightarrow \quad \rho(A)=\lambda$$

提纲

- 1. 矩阵不等式
- 2. 正的矩阵
- 3. 非负矩阵
- **4. 随机矩阵**
- 5. PageRank

4.1 什么是随机矩阵

- 具有性质 $Ae=e$，即所有行和都等于+1的非负矩阵称为（行）随机矩阵

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

4.1 什么是随机矩阵

具有性质$e^TA=e^T$，即所有列和都等于+1的非负矩阵称为（列）随机矩阵

$$
\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}
\begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix}
= \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}
$$

使得A^T也为随机矩阵的随机矩阵$A \in M_n$称为双随机的

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

4.2 性质

- +1是随机矩阵A的特征值，且$\rho(A)=1$
- M_n中全体随机矩阵的集合是一个紧集，而且是一个凸集。
- $n\times n$的随机矩阵至少有n个非零元素

4.2 性质

【Birkhoff定理】矩阵$A \in M_n$是双随机的，当且仅当存在置换矩阵P_1，P_2，…，$P_N \in M_n$以及正的纯量t_1，t_2，…，$t_N \in R$，使得

$t_1+t_2+\ldots+t_N=1$以及$A= t_1 P_1 +t_2 P_2 +\ldots+t_N P_N$

4.2 性质

【von Neumann定理】设$A,B\in M_n$，有序排列的奇异值是 $\sigma_1(A)\geq \sigma_2(A)\geq \ldots \geq \sigma_n(A)$ 以及 $\sigma_1(B)\geq \sigma_2(B)\geq \ldots \geq \sigma_n(B)$ 。那么

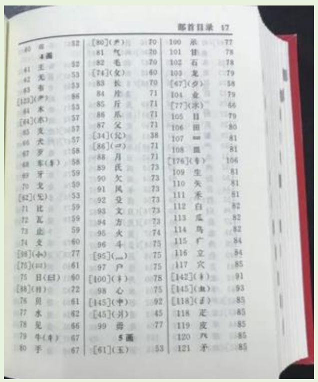
$$\operatorname{Re}\operatorname{tr}(AB)\leq \sum_{i=1}^{n}\sigma_i(A)\sigma_i(B)$$

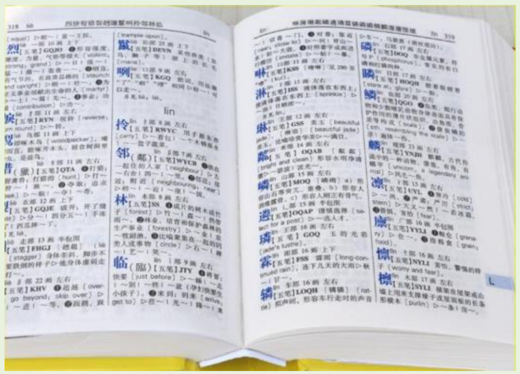
提 纲

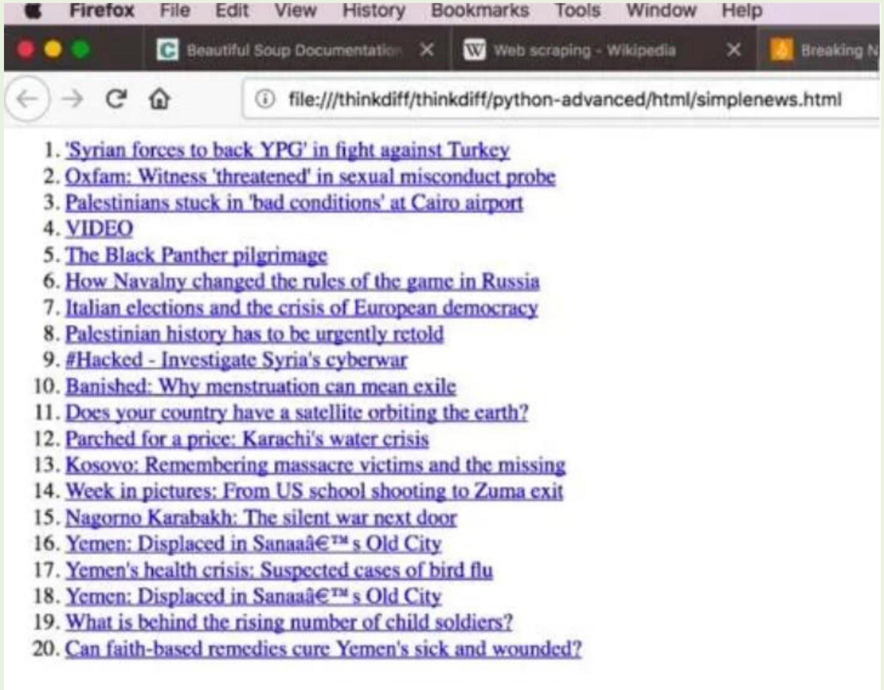
- 1. 矩阵不等式
- 2. 正的矩阵
- 3. 非负矩阵
- 4. 随机矩阵
- **5. PageRank**

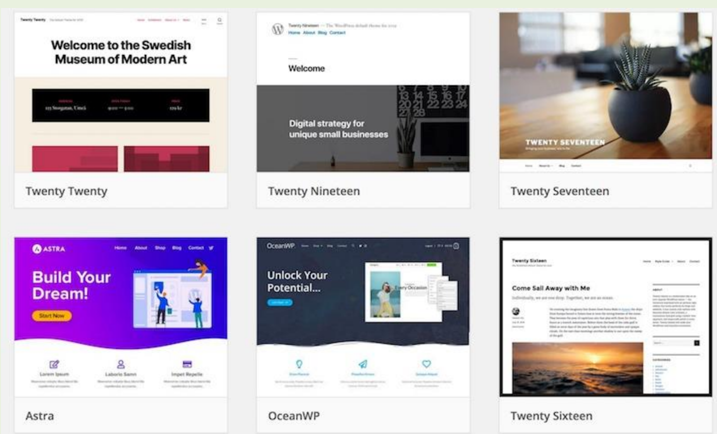
PageRank

- 传统搜索vs. 网页搜索









1. 搜索对象的数量较小

一本字典收录的字通常只有一两万个，
一家图书馆收录的不重复图书通常不超过几十万种，
一家商店的商品通常不超过几万种

2. 搜索对象具有良好的分类或排序

字典里的字按拼音排序
图书馆里的图书按主题分类
商店里的商品按品种或用途分类

3. 搜索结果的重复度较低

字典里的同音字通常不超过几十个
图书馆里的同名图书和商店里的同种商品通常也不超过几十种

互联网的鲜明特点却是以上三条无一满足。

互联网发展的早期，**Yahoo**曾试图为网页建立分类系统，
但随着网页数量的激增，这种做法很快就“挂一漏万”了。
而搜索结果的重复度更是以快得不能再快的速度走向失控。

在谷歌主导互联网搜索之前， 多数搜索引擎采用的排序方法， 是以被搜索词语在网页中的出现次数来决定排序——出现次数越多的网页排在越前面。

这个判据不能说毫无道理， 因为用户搜索一个词语，通常表明对该词语感兴趣。

既然如此， 那该词语在网页中的出现次数越多， 就越有可能表示该网页是用户所需要的。

1996 年初，谷歌公司的创始人，当时还是**Stanford University** 研究生的佩奇 (**Larry Page**) 和布林 (**Sergey Brin**) 开始了对网页排序问题的研究。

这两位小伙子之所以研究网页排序问题，一来是导师的建议 (佩奇后来称该建议为“我有生以来得到过的最好建议”)，二来则是因为他们对这一问题背后的数学产生了兴趣。

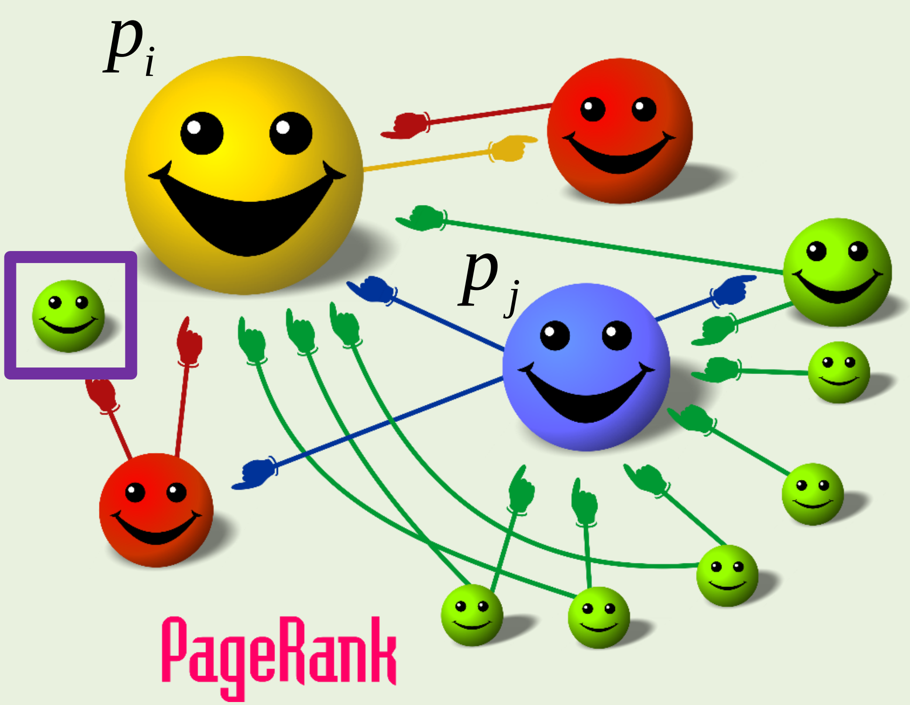
一个网页排序的思路，那就是通过研究网页间的相互链接来确定排序。

具体地说，一个网页被其它网页链接得越多，它的排序就应该越靠前。

依照这个思路，网页排序问题就跟整个互联网的链接结构产生了关系，正是这一关系使它成为了一个不折不扣的数学问题。

PageRank

- **核心问题：如何对网页排序**



$p_i(n)$：用户第n次浏览时访问网页W_i的概率

N_i：网页W_i有N_i个对外链接

$$p_{j\to i}=\begin{cases}1 & \text{网页}W_j\text{有链接指向}W_i\\ 0 & \text{网页}W_j\text{无链接指向}W_i\end{cases}$$

$$\vdots \qquad \vdots$$

$$p_i(n+1)=\sum_j p_j(n)\,p_{j\to i}/N_j$$

$$\vdots \qquad \vdots$$

PageRank

$$\boldsymbol{p}_{n+1} = H\boldsymbol{p}_n$$

$$H_{ij} = p_{j\to i} / N_j$$

$$\boldsymbol{p}_n = H^n \boldsymbol{p}_0$$

p_0为虚拟读者初次浏览时访问各网页的几率分布 (在佩奇和布林的原始论文中， 这一几率分布被假定为是均匀分布)。

如果这三个问题的答案都是肯定的，那么网页排序问题就算解决了。 反之， 哪怕只有一个问题的答案是否定的， 网页排序问题也就不能算是得到了满意解决。
很遗憾， 是后一种， 而且是其中最糟糕的情形， 即三个问题的答案全都是否定的。

① $\lim \boldsymbol{p}_n$ 是否存在？ ✗

② 如果极限存在，是否与$\boldsymbol{p}_0$无关？ ✗

③ 使用极限向量进行排序是否合理？ ✗

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

在只包含两个相互链接网页的迷你型互联网上，

$$H=\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

如果 $p_0=(1, 0)^T$，

极限不存在 (几率分布在 $(1, 0)^T$ 和 $(0, 1)^T$ 之间振荡)。

另外任何一个“悬挂网页”都能象黑洞一样，把其它网页的几率“吸收”到自己身上 (因为虚拟用户一旦进入那样的网页，就会由于没有对外链接而永远停留在那里)，这显然是不合理的。

对于真实用户来说，自行访问的网页显然与各人的兴趣有关，但对于在平均意义上代表真实用户的虚拟用户来说，可以假定它将会在整个互联网上随机选取一个网页进行访问。

用数学语言来说，这相当于是把 **H** 的列向量中所有的零向量都换成 **e/N** (其中 **e** 是所有分量都为 **1** 的列向量，**N** 为互联网上的网页总数)。如果我们引进一个描述“悬挂网页”的指标向量 (**indicator vector**) **a**，它的第 **i** 个分量的取值视 W_i 是否为“悬挂网页”而定——如果是“悬挂网页”，取值为 **1**，否则为 **0**

修正后的矩阵的每一列的矩阵元之和都是 **1**，是一个随机矩阵。删除了“悬挂网页”。从而可以给上述第三个问题带来肯定回答 (当然，这一回答没有绝对标准，可以不断改进)。不过，这一修正解决不了前两个问题。

为了解决那两个问题，进一步引进修正。

PageRank

$$\boldsymbol{a}(j)=\begin{cases}1 & \text{网页}\mathbf{W}_j\text{是悬挂网页}\\ 0 & \text{其它}\end{cases}$$

- 修正方法：

$$H \Longrightarrow D=H+\boldsymbol{e}\boldsymbol{a}^{\mathrm{T}}/S$$

$$\begin{bmatrix} 0 & \cdots & 0 & \cdots & \vdots \\ \vdots & \ddots & 0 & \cdots & \vdots \\ 1/N_{i1} & \cdots & 0 & \cdots & \vdots \\ \vdots & \cdots & 0 & \ddots & \vdots \\ 1/N_{S1} & \cdots & 0 & \cdots & 0 \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & \cdots & 1/S & \cdots & \vdots \\ \vdots & \ddots & 1/S & \cdots & \vdots \\ 1/N_{i1} & \cdots & 1/S & \cdots & \vdots \\ \vdots & \cdots & 1/S & \ddots & \vdots \\ 1/N_{S1} & \cdots & 1/S & \cdots & 0 \end{bmatrix}$$

假定虚拟用户在每一步都有一个小于 1 的几率 α 访问当前网页所提供的链接， 同时却也有一个几率 1-α 不受那些链接所限， 随机访问互联网上的任何一个网站。增加了网页访问的随机性和个性。
谷歌矩阵不仅是一个随机矩阵， 而且由于第二项的加盟， 它有了一个新的特点， 即所有矩阵元都为正 。
如果我们用 p 表示 p_n 的极限， 则 p 给出的就是整个互联网的网页排序——它的每一个分量就是相应网页的访问几率， 几率越大， 排序就越靠前。
而且 “佩奇排序” 还有一个重要特点， 那就是它只与互联网的结构有关， 而与用户具体搜索的东西无关。这意味着排序计算可以单独进行， 而无需在用户键入搜索指令后才临时进行。 谷歌搜索的速度之所以快捷，在很大程度上得益于此。

PageRank

- 谷歌矩阵（**Google matrix**）：

$$D = H + \boldsymbol{e}\boldsymbol{a}^{\mathrm{T}} / S \quad \Rightarrow \quad G = \alpha S + (1-\alpha)\boldsymbol{e}\boldsymbol{e}^{\mathrm{T}} / N$$

$$\boldsymbol{p}_n = G^n \boldsymbol{p}_0$$

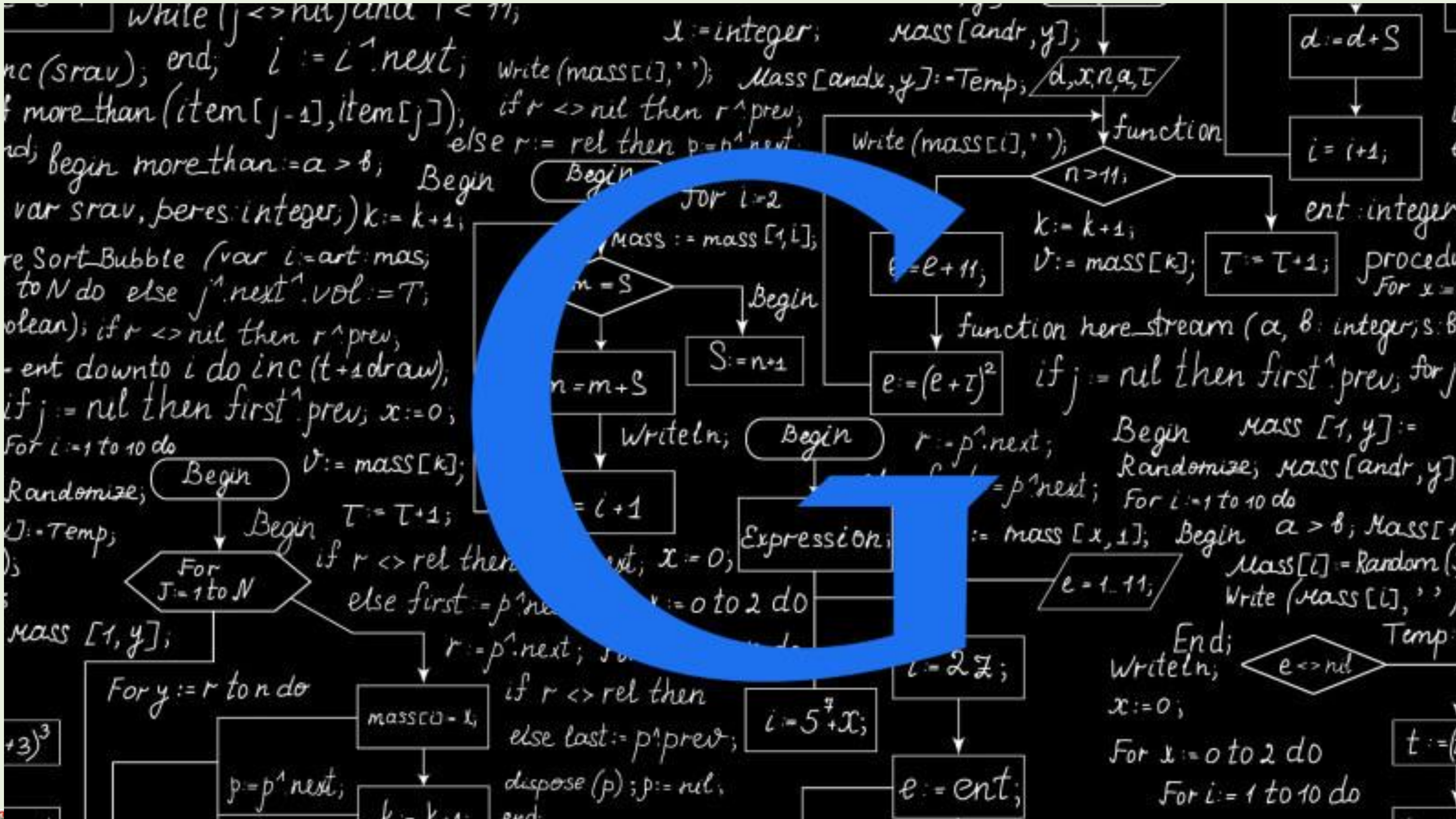
$$\boldsymbol{p} = \lim_{n\to\infty} \boldsymbol{p}_n$$

PageRank算法是由拉里·佩奇(**Larry Page**)在斯坦福大学读博士期间开发的。他对创建一种更准确的方法来在搜索结果中对网站进行排名感兴趣，并提出了使用页面之间的链接来确定重要性的想法。他第一次在自己的搜索引擎**Backrub**中使用这种算法，后来变成了谷歌。

该算法立即获得了成功，并迅速成为谷歌搜索引擎的关键组成部分。如今，**PageRank**被其他搜索引擎使用，被认为是搜索领域最重要的算法之一。

与佩奇和布林研究排序算法几乎同时， 有另外几人也相互独立地沿着类似的思路从事着研究。 他们中有一位是当时在美国新泽西州工作的中国人， 他的算法后来也成就了一家公司——一家中国公司。 此人的名字叫做李彦宏 (**Robin Li**)， 他所成就的那家公司就是百度

PageRank


x := integer;
Write (mass[i], ' ');
if r <> nil then r^prev;
Begin
function
n > 11;
k := k + 1;
Sort_Bubble (var i : art mas;
function here_stream (a, b : integer;
if j = nil then first^prev;
Writeln;
Begin
Randomize;
For i := 1 to 10 do
Expression;
else first := p^next;
For J := 1 to N
For y := r to n do
if r <> rel then
else last := p^prev;
dispose (p); p := nil;
For i := 1 to 10 do