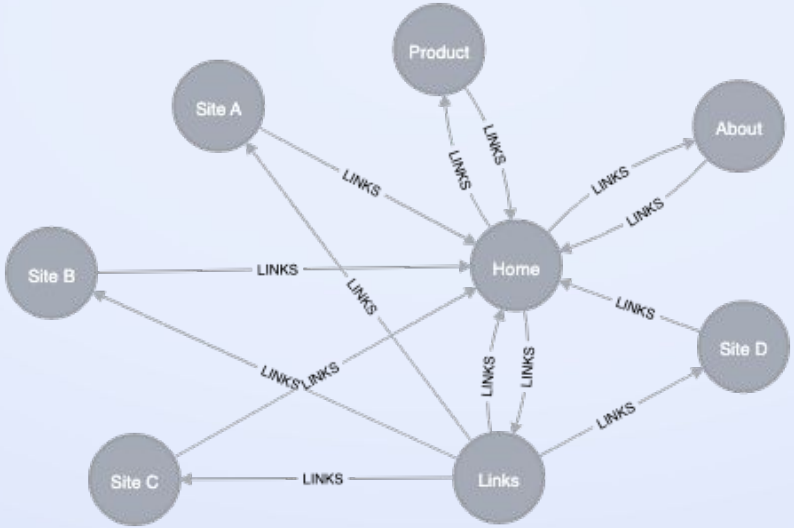
Google PageRank

Bhavini Kadiwala, Emily Tran, Jose Lopez, Selam Berekat

What is Google PageRank?

PageRank: An algorithm used by Google Search to rank pages

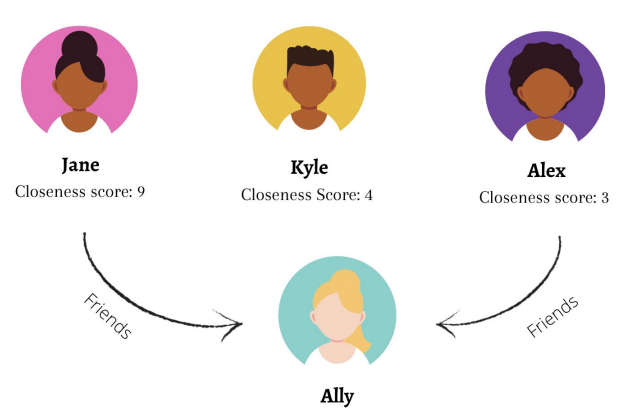
- Most relevant on top
- Importance of web page is determined by links



Classmate example:

Formula:
FS(Ally) = ∑ p ∈ F FS(p) * (IsFriends(p, Ally)/ NumFriends(p))

4 Potential Classmates:



Classmates and their FS:

Person	FriendScore(FS)
Emily	FS(Emily) = ∑ p ∈ F FS(p) * (IsFriends(p,Emily)/ NumFriends(p))
Jose	FS(Jose) = ∑ p ∈ F FS(p) * (IsFriends(p,Jose)/ NumFriends(p))
Salem	FS(Jose) = ∑ p ∈ F FS(p) * (IsFriends(p,Jose)/ NumFriends(p))

Matrix A:

	Emily	Jose	Salem
Emily	0	1	1/2
Jose	1/2	0	1/2
Salem	1/2	0	0

To get our next FriendScore, we take our current guess (FSi) * A matrix →FS(i+1).
We keep continuing this process until FS(i+1) stops changing and it becomes FS*, where **A * FS* = FS*.**

Formula for an eigenvector: **Ax = λx**.
This makes our **FS*** value an eigenvector of **A**.

Definition

$$g_{ij} = \begin{cases} 1, & \text{if hyperlink from pg i to pg j exists} \\ 0, & \text{else} \end{cases}$$

$$r_i = \sum_j g_{ij}, \quad c_j = \sum_i g_{ij}.$$

$$a_{ij} = \begin{cases} pg_{ij}/c_j + \delta, & c_j \neq 0 \\ 1/n, & c_j = 0 \end{cases}$$

$$\delta = (1-p)/n$$

$$\sum_i x_i = 1,$$

$$x = Ax \Longrightarrow (I - A)x = 0$$

$$A = pGD + ez^T$$

$$(I - pGD)x = \gamma e \qquad \gamma = z^T x$$

$$d_{jj} = \begin{cases} 1/c_j & : \quad c_j \neq 0 \\ 0 & : \quad c_j = 0, \end{cases}$$

$$z_j = \begin{cases} \delta & : \quad c_j \neq 0 \\ 1/n & : \quad c_j = 0. \end{cases}$$

1. Regular Page Rank

With G defined as the link-matrix, we have the transition probability matrix of the Markov Chain as:

$$A = pGD + ez^T$$

Then take the following steps to find the pagerank vector x:

$$x = Ax$$

$$(I - A)x = 0 \Longrightarrow (I - pGD)x = \gamma e$$

$$\gamma = z^T x.$$

The following MATLAB code calculates this particular algorithm:

```
i = [2 3 3 4 1 3 6 5];
j = [1 1 2 2 3 4 5 6];
G = sparse(i,j,1,n,n);
pagerank1(G)

function rank_vector = pagerank1(G)
    [m,n] = size(G);
    p = 0.85;
    c = sum(G,1);
    k = find(c~=0);
    D = sparse(k,k,1./c(k),n,n);
    e = ones(n,1);
    I = speye(n,n);
    x = (I - p*G*D)\e;
    rank_vector = x / sum(x);
end
```

2. Power Method

The power method begins by multiplying the transition matrix (which includes the random jump probabilities) by the “equal probability vector”:

$$x = e/n$$

The vector produced is then multiplied by the same transition matrix, and this is iterated until the resulting vector converges to several decimal places.
The vector that is obtained at the end of the iteration is the pagerank vector.

The following MATLAB code calculates this particular algorithm:

```
function rank_vector3 = pagerank_pwrmthd(G)
    [m,n] = size(G);
    p = 0.85;
    delta = (1-p)/n;
    c = sum(G,1);
    k = find(c~=0);
    D = sparse(k,k,1./c(k),n,n);
    e = ones(n,1);
    G = p*G*D;
    z = ((1-p)*(c~=0) + (c==0))/n;
    x = e/n;
    while true
        u = G*x + e*(z*x);
        if round(x,4) == round(u,4)
            break
        end
        x = u;
    end
    rank_vector3 = u;
end
```

3. Inverse Iteration

This method takes advantage of the fact that we know the first eigenvalue is equal to one. It then uses this to find the corresponding eigenvector in one iteration. This iteration involves the inversion of the transition probability matrix.

The following MATLAB code calculates this particular algorithm:

```
function rank_vector2 = pagerank_inviter(G)
    [m,n] = size(G);
    p = 0.85;
    delta = (1-p)/n;
    c = sum(G,1);
    k = find(c~=0);
    D = sparse(k,k,1./c(k),n,n);
    e = ones(n,1);
    I = speye(n,n);
    A = p*G*D + delta;
    x = (I - A)\e;
    rank_vector2 = x/sum(x);
end
```

Comparison of Execution Times

To compare which pagerank computes the algorithm the fastest, we used the "timeit" function in MATLAB and obtained the following results:

PageRank (Original): 8.7667e-05

Power Method: 1.5614e-04

Inverse Iteration: 0.0023

**The "timeit" function runs a function several times and obtains the median runtime in seconds.

The following code was used to compute these times:

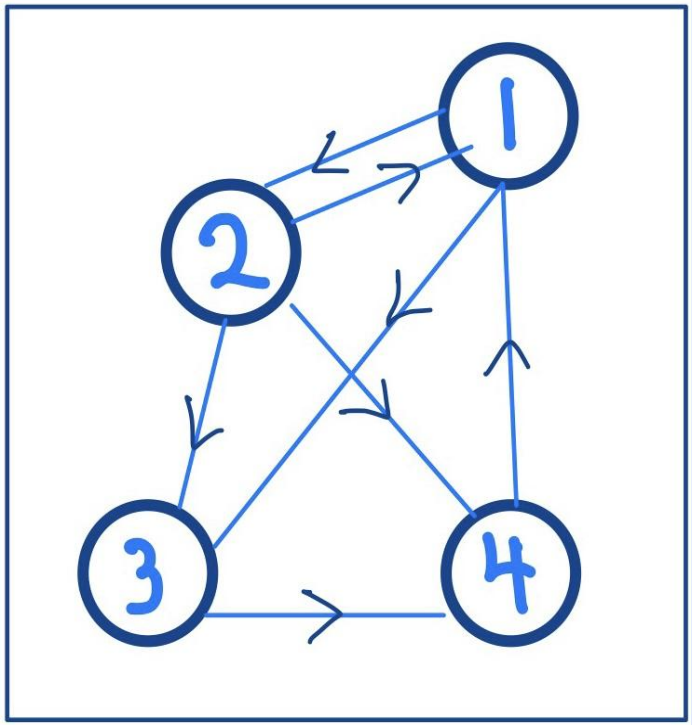
```
f = @() pagerank1(G);
timeit(f)

f = @() pagerank_inviter(G);
timeit(f)

f = @() pagerank_pwrmthd(G);
timeit(f)
```

Example on PageRank Algorithm

Line Graph



Adjacency Matrix (G)

0	1	0	1
1	0	0	0
1	1	0	0
0	1	1	0

Probability Matrix (T)

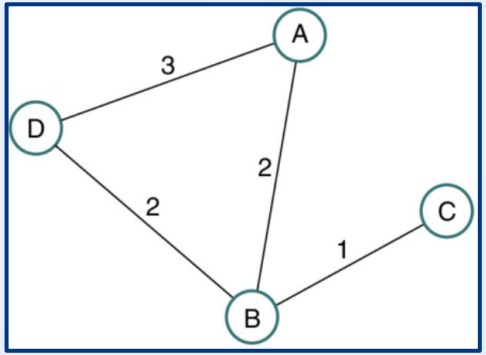
0	0.3333	0	1.0000
0.5000	0	0	0
0.5000	0.3333	0	0
0	0.3333	1.0000	0

Matrix Representation conti..

Irreducible Matrix (Q), Alpha = 0.85

0.0375	0.3208	0.0375	0.8875
0.4625	0.0375	0.0375	0.0375
0.4625	0.3208	0.0375	0.0375
0.0375	0.3208	0.8875	0.0375

Reducible Graph



$$T \cdot A \cdot T^{T} = \begin{bmatrix} X & Y \\ 0 & Z \end{bmatrix}$$

Ranking Score

0.3231
0.1748
0.2244
0.2777

Conclusion

Does Google still use this PageRank Implementation?

Yes! It is still an important internal tool they use although there are many added “quirks” to the PageRank algorithm.

There are over 200 Google ranking factors and the main ones include:

- Backlinks
- Internal Linking
- External Linking

Sources

https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/moler/lu.pdf

https://math.stackexchange.com/questions/936757/why-is-pagerank-an-eigenvector-problem

https://towardsdatascience.com/pagerank-algorithm-fully-explained-dc794184b4af

http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf

https://www.dhruvonmath.com/2019/03/20/pagerank/

http://pi.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture3/lecture3.html