# CS630 Graduate Algorithms
# November 25, 2024

# Dora Erdos and Jeffrey Considine

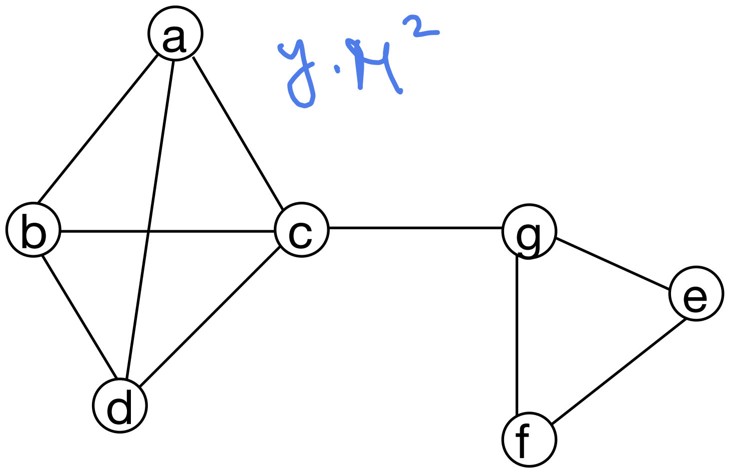

PageRank
Random walks on graphs


Slides by Dora Erdos

# Graph adjacency matrix

$y = [1,1,0,0,0,0,0]$

$y \cdot M^2$




M[i,j]=1 if there is a directed edge from i to j

x = [1,0,0,0,0,0,0] represents starting the walk at vertex a

Then:

xM = [0,1,1,1,0,0,0] $O(n^2)$

$xM^2$ = [3,2,2,2,0,0,1]

$xM^3$ = [6,7,8,7,1,1,2]

n×n matrices

$$A \cdot B[i,j] = \sum_{k=1}^{n} A[i,k] \cdot B[k,j]$$

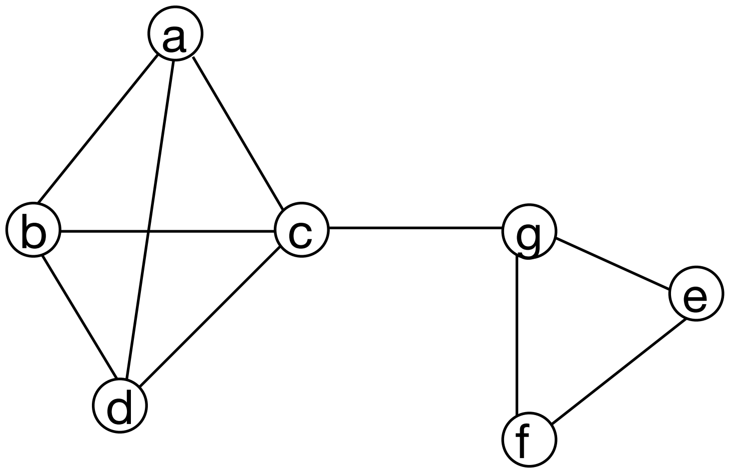
$O(n^3)$

⇒ multiply index k in row with index k in column j

$$M = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \end{array} \begin{array}{c} \begin{array}{ccccccc} a & b & c & d & e & f & g \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array} \quad M^2 = \begin{bmatrix} 3 & 2 & 2 & 2 & 0 & 0 & 1 \\ 2 & 3 & 2 & 2 & 0 & 0 & 1 \\ 2 & 2 & 4 & 2 & 1 & 1 & 0 \\ 2 & 2 & 2 & 3 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 2 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 3 \end{bmatrix} \quad M^3 = \begin{bmatrix} 6 & 7 & 8 & 7 & 1 & 1 & 2 \\ 7 & 6 & 8 & 7 & 1 & 1 & 2 \\ 8 & 8 & 6 & 8 & 1 & 1 & 6 \\ 7 & 7 & 8 & 6 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 & 3 & 2 & 4 \\ 2 & 2 & 6 & 2 & 4 & 4 & 2 \end{bmatrix}$$

# Graph adjacency matrix

M[i,j]=1 if there is a directed edge from i to j




Let $x = [x_1, x_2, \ldots, x_n]$ represent the starting vertices of a walk.

xi can be 0/1

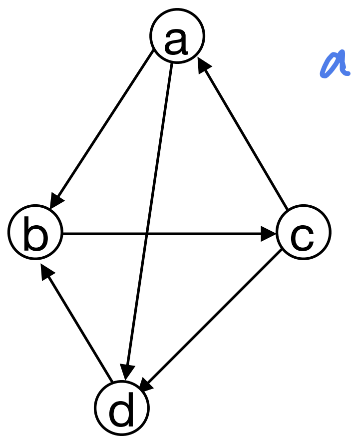
xi can be a probability → could be uniform
↳ we don't have any additional info
↳ not uniform
↳ we know smg more about the data or app.

$xM^k[j]$ = the number of ways we can reach j in k steps if we start based on x.

$$M = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \quad M^2 = \begin{bmatrix} 3 & 2 & 2 & 2 & 0 & 0 & 1 \\ 2 & 3 & 2 & 2 & 0 & 0 & 1 \\ 2 & 2 & 4 & 2 & 1 & 1 & 0 \\ 2 & 2 & 2 & 3 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 2 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 3 \end{bmatrix} \quad M^3 = \begin{bmatrix} 6 & 7 & 8 & 7 & 1 & 1 & 2 \\ 7 & 6 & 8 & 7 & 1 & 1 & 2 \\ 8 & 8 & 6 & 8 & 1 & 1 & 6 \\ 7 & 7 & 8 & 6 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 & 3 & 2 & 4 \\ 2 & 2 & 6 & 2 & 4 & 4 & 2 \end{bmatrix}$$

# Graph adjacency matrix

D[i,j]=1 iff there is a directed edge from i to j.




asymmetric

$$D = \begin{bmatrix} 0 & \boxed{1} & 0 & 1 \\ \boxed{0} & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad D^2 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad D^3 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 2 & 0 & 1 \\ 0 & 1 & 2 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$x = [1,1,0,0]$

$xM^2$ = # of ways to get to each vertex in 2 steps if we start from a or b.

- $D^2$[i,j] contains the number of length-2 directed path from i to j.
- $D^k$[i,j] contains the length-k paths.
- The sum of column i in $D^k$ is the total number of ways we can get to i in k steps.

Let x = [$x_1$,$x_2$,…,$x_n$] be a vector that represents the probability that we start in each node

Then $xD^k$ tells us where the walk is after k steps if we start based on x.

↳ how likely we are to be in a certain location after k steps

# “status” as centrality measure

- vertices have a status (importance) score
- vertices have ‘high’ status if they are connected to vertices with ‘high’ status (this is a circular definition ;) )

Centrality $x_i$ of a node $i$ is the sum of the centrality of its neighbors, scaled by some constant.

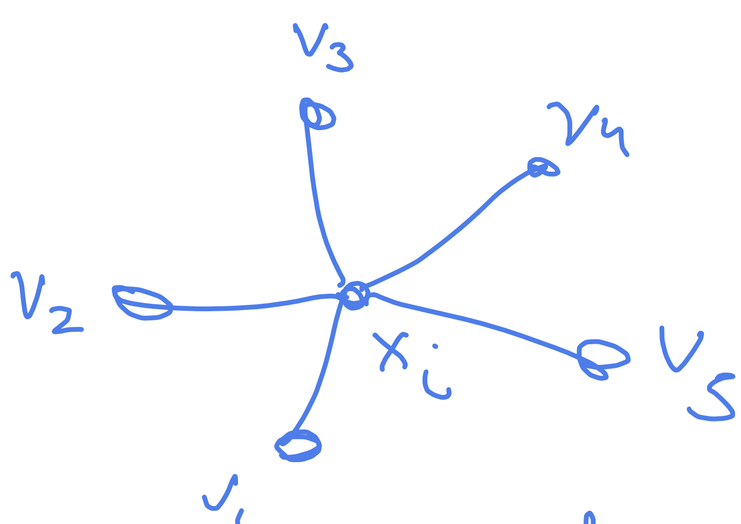
A = adjancy mtx → edge $(i,j)$

$A[i,j] = 1$

$$x_i = \frac{1}{\lambda} \sum_{(i,j) \in E} x_j = \sum_{A[i,j]=1} x_j$$

↳ every neighbor of $x_i$

$\lambda$ is a constant that we chose in a clever way




insight:
- we start with same initial values of status for each $x_i$
- iteratively each $x_i$ gets updated based on their neighbors current status

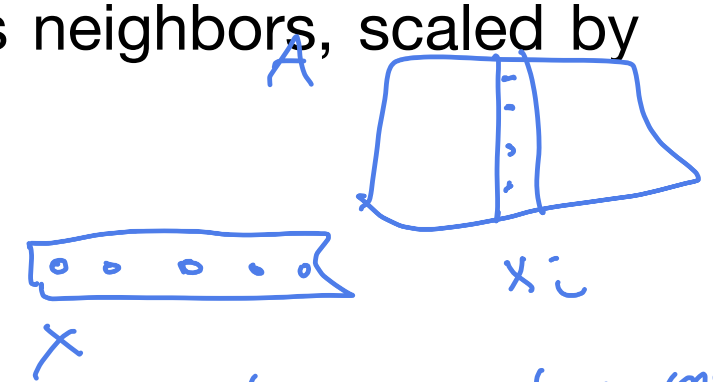
⇝ $x_i$ is also influencing $x_j$

turns out that after many iterations this stabilizes

→ $x_i$ won't change anymore

# Eigenvector centrality

Centrality $x_i$ of a node $i$ is the sum of the centrality of its neighbors, scaled by some constant.




$$x_i = \frac{1}{\lambda} \sum_{(i,j)\in E} x_j = \frac{1}{\lambda} \underbrace{\sum_{A[i,j]==1} x_j}$$

index i of the vector – mtx multiplication
xA

Arrange the centrality values in a vector x = [$x_1$,$x_2$,….,$x_n$].

$$x = \frac{1}{\lambda} xA$$

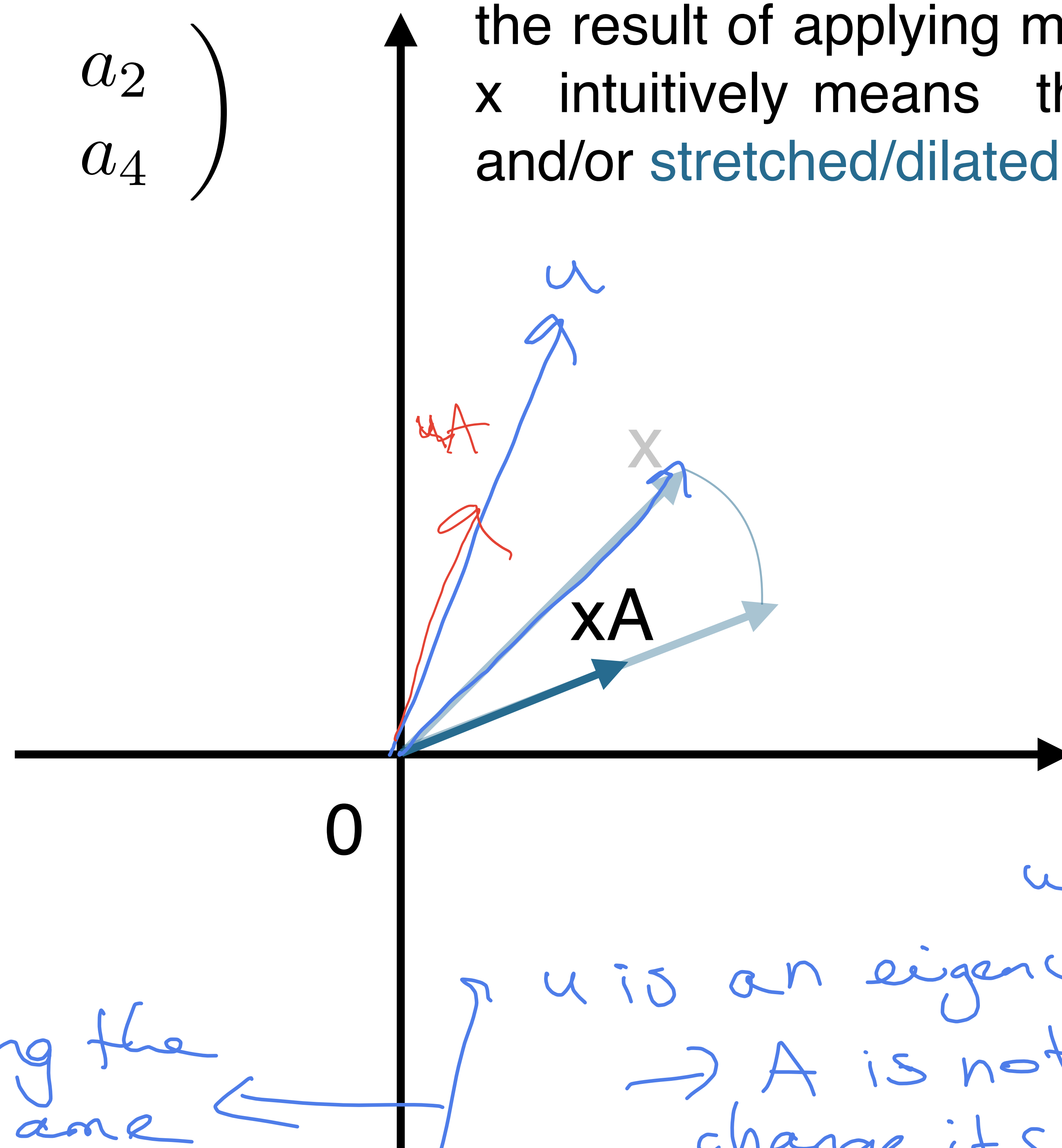
Rearranging we get $\underbrace{x\lambda = xA}$

equation that defines eigenvalues and eigenvectors

x[i] is called the eigenvector centrality of node i.
if we choose $\lambda$ to be the largest eigenvalue, than all x[i] are positive. (more on this later)

$$A = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$$

$n \times n$




the result of applying matrix A to a vector x intuitively means that x gets rotated and/or stretched/dilated.

$\lambda u = uA$

$\Uparrow$

changing the length same as multiplying by a constant $\lambda$ $\rightarrow \lambda u$

$uA$

u is an eigenvector

$\rightarrow$ A is not going to change its angle only its length

# Eigenvalue decomposition

Assume $A$ is an $n$ x $n$ square matrix
$u$ is an n-dim vector

$$v = uA = \sum_{j=1}^{n} u_j A[*, j]$$

You can think of $v$ as a linear combination of the columns of $A$.

$u$ is an eigenvector if applying $A$ to it only changes its magnitude not its direction.

$$\lambda u = uA$$

If $A$ has rank $n$, then there are $n$ orthonormal eigenvectors and eigenvalues.

vectors x and y are orthonormal if

$x \cdot x^T = 1$

$x \cdot y^T = 0$

Eigenvalue decomposition of A

$$A = U \Lambda U^{-1}$$

U contains the eigenvectors as its columns
$\Lambda$ is a diagonal matrix with the eigenvectors in its diagonal

$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

## Eigenvalue decomposition and powers of a matrix

eigenvalue decomposition $A = V\Lambda V^T$

- V is orthonormal $VV^T = V^TV = I = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix}$
- $\Lambda$ is diagonal

$$A = V\Lambda V^T$$

$$\Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \lambda_n \end{bmatrix}$$

$$AV = V\Lambda \underbrace{V^T V}_{I}$$

$$AV = V\Lambda$$

compute $A^k$:

$$A = V\Lambda V^T$$

$$A^2 = V\Lambda \underbrace{V^T V}_{I} \Lambda V^T = V\Lambda^2 V^T$$

$$A^k = V\lambda V^T V\lambda V^T \ldots V\Lambda V^T = V\Lambda^k V^T$$

$$\Lambda^k = \begin{bmatrix} \lambda_1^k & & & 0 \\ & \lambda_2^k & & \\ & & \ddots & \\ 0 & & & \lambda_n^k \end{bmatrix}$$

Not sure how to use it yet, but there is definitely some kind of relationship between $A^k$ and eigenvector centrality.

# Path lengths

- degree centrality - length 1 path
- adjacency matrix = # of length-1 paths $A$
- # of length-k paths $A^k$
- # of paths of length at most k $\sum_{i=1}^{k} A^i$

start based on
$x = [x_1, x_2, \ldots, x_n]$
$\Rightarrow$ length-k path
from x:
$xA^k$

- infinite paths?
    - define centrality as the relative fraction of paths passing through a vertex
    - equivalent to asking: starting from a random vertex and taking some number of steps along the edges, how likely are we to end up in vertex v?

next up:
How to compute $xA^k$

# Walking on the graph — power method

$\hookrightarrow$ powers of a matrix

Let $x^{(0)} = [x_1, x_2, \ldots, x_n]$ correspond to some initial state of the vertices.

- The $^{(0)}$ in the notation corresponds to the 0th iteration, not an exponent.

$x^{(0)} = x$

$x^{(t)}$ = iteration $t$ → taking $t$ steps if we start from $x$

Now take a step

$$x^{(1)} = x^{(0)} A$$ = 1 step on the graph from $x$

$x_i^{(1)}$ is the weight-scaled notion of being in node i.

Another step, then many more

$k$ iterations

$$x^{(1)}$$

$$x^{(2)} = x^{(1)} A = x^{(0)} A^2$$ = 2 steps starting from $x$

$$\vdots$$

$$x^{(k)} = \underbrace{x^{(k-1)} A}_{\text{one more step}} = x^{(0)} A^k$$ → intuition: $x^{(k)}$ = where we end up after $k$ steps

$x^{(k-1)}$ = $k-1$ steps

computing $A^2$ is a mtx mult $O(n^3)$

$x^{(1)}$ = vector $\quad x^{(1)} \cdot A = O(n^2)$

$A^k = \underbrace{A^{k-1} \cdot A}_{O(n^3)}$

$$x^{(2)} = x^{(0)} \cdot A^2 = \underbrace{x^{(0)} \cdot A}_{x^{(1)}} \cdot A = x^{(t)} A$$

$\underbrace{x^{(k-1)} \cdot A}_{O(n^2)}$

$x^{(k)}$ = one more step after $x^{(k-1)}$

# Walking on the graph — power method

Let $x^{(0)} = [x_1, x_2, \ldots, x_n]$ correspond to some initial state of the vertices.

- The $^{(0)}$ in the notation corresponds to the 0th iteration, not an exponent.

In summary we have

$$x^{(1)} = x^{(0)} A$$

$$x^{(k)} = x^{(0)} A^k$$

$\|v\|_2$ = length of vector v

2 norm

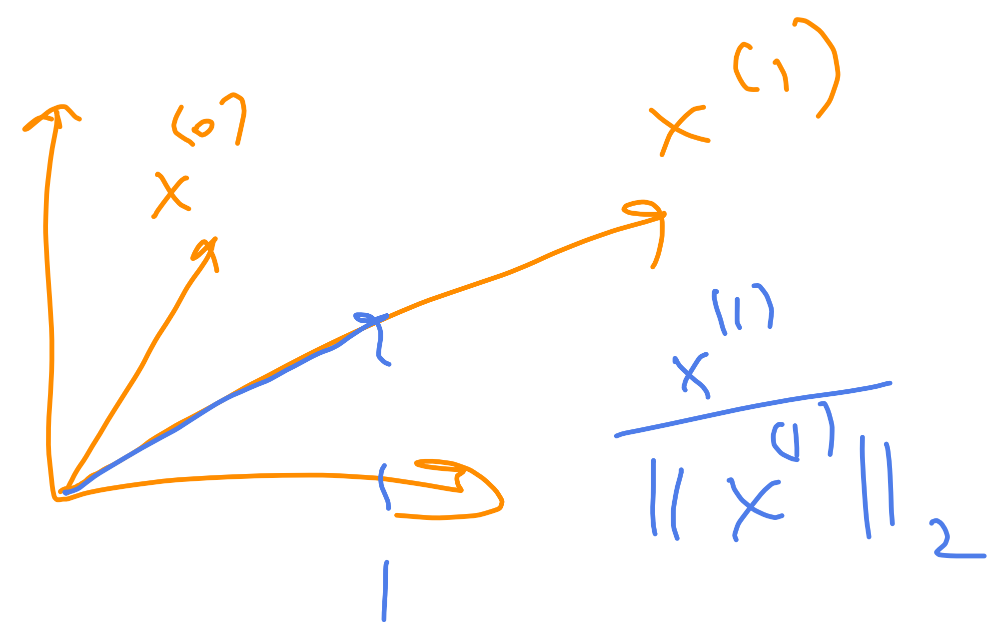
If x can be any kind of weight, then these would be really large numbers. So let's normalize in each iteration.

$\frac{v}{\|v\|_2}$ = point in same direction as v but has length 1

$$\text{Assume } \|x^{(0)}\|_2 == 1 \text{ (or take } x^{(0)} = \frac{x^{(0)}}{\|x^{(0)}\|_2}\text{)}$$

$$x^{(1)} = x^{(0)} A$$

$$x^{(1)} = \frac{x^{(1)}}{\|x^{(1)}\|_2}$$




After k iterations

$$x^{(k)} = x^{(k-1)} A = x^{(0)} A^k$$

$$x^{(k)} = \frac{x^{(k)}}{\|x^{(k)}\|_2}$$

# Walking on the graph — power method

$$x^{(0)} = \frac{x^{(0)}}{\|x^{(0)}\|_2}$$

$$x^{(k)} = x^{(k-1)}A = x^{(0)}A^k$$

$$x^{(k)} = \frac{x^{(k)}}{\|x^{(k)}\|_2} = \frac{x^{(0)}A^k}{\|x^{(0)}A^k\|_2}$$

power method:
each iteration
1. multiply by A : $x^{(k-1)} \cdot A$
2. divide by its length: $x^{(k)} \leftarrow \frac{x^{(k)}}{\|x^{(k)}\|_2}$

Power method: instead of computing $x^{(k)}$ directly from $A^k$ we perform the k iterations to get there.

Now we can think of $x_i^{(k)}$ as some kind of probability of being in node i after k steps (since x sums to 1).

- this is in fact not quite the proper probability distribution, but very similar

# Search engines on the web

In the mid 90s there were dozens of search engines. Then, after 1998 Google emerged as the engine and the others mostly disappeared.

How did search engines work?

- submit a query , e.g. “personal 737 jet”
- first step: basic text processing is done to find the pages containing the term
  - thousands of pages are retrieved
- second step: show the results to the user
  - Google turned out to be the best at finding the most relevant ones
  - they used PageRank to decide relevance
-

Brin, S.; Page, L. (1998). "The anatomy of a large-scale hypertextual Web search engine" (PDF). *Computer Networks and ISDN Systems*. **30** (1–7): 107–117.

# PageRank centrality — random walks

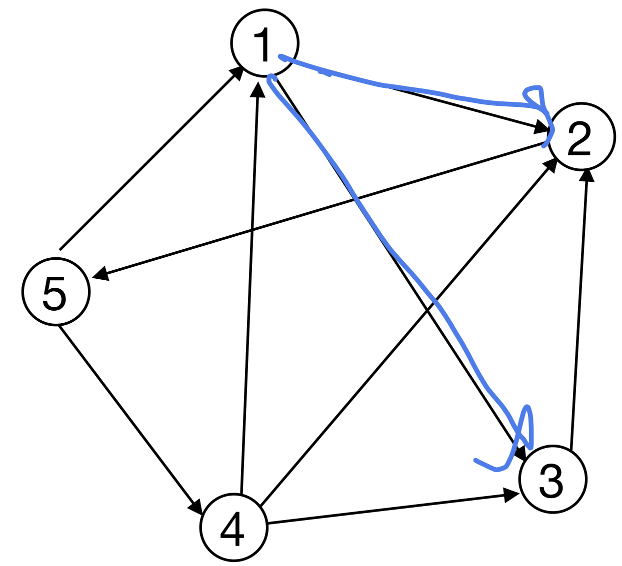
Setting:
- on the Web there are sites and  hyperlinks directing from one page to an other.
- remember the status score?
  - high status nodes (sites) are the ones pointed (linked) by high status nodes.
  - here links are endorsements by the author
- we are still on a quest to assign the “status” score to nodes

Random surfer:
- we start on a random page
- at each step we follow one of the outgoing links at random
- sometimes we get bored (e.g. quit browsing), and start over at some random page
- if we were to walk for a looooong time, then our starting page doesn’t matter anymore
- we should find the probability that at any given time we are in any given page
- it turns out that after many steps this probability is independent of the walk itself
- this probability distribution (called stationary distribution) is Pagerank
  - is related to Markov Chains

# Random walks on graphs — PageRank

→ PageRank one number per node

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




→ random walks is the start

The transition matrix P represents the probability of traversing from one node to its neighbor.

→ current location after t steps (probability wise)

The (row) vector $q^{(t)}$ contains the probability that we are in each node after t steps.

Then $q^{(t+1)} = q^{(t)}P$ → take one more step after t

$$P = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix} \quad q^{(t)} \cdot P$$

go uniform rnd to one of the 3 edges out of node 4

$$\begin{cases} q_1^{(t+1)} = \frac{1}{3}q_4^{(t)} + \frac{1}{2}q_5^{(t)} \\ q_2^{(t+1)} = \frac{1}{2}q_1^{(t)} + q_3^{(t)} + \frac{1}{3}q_4^{(t)} \\ q_3^{(t+1)} = \frac{1}{2}q_1^{(t)} + \frac{1}{3}q_4^{(t)} \\ q_4^{(t+1)} = \frac{1}{2}q_5^{(t)} \\ q_5^{(t+1)} = q_2^{(t)} \end{cases}$$

← (subscript = index in vector $q^{(t)}$)

general formula:

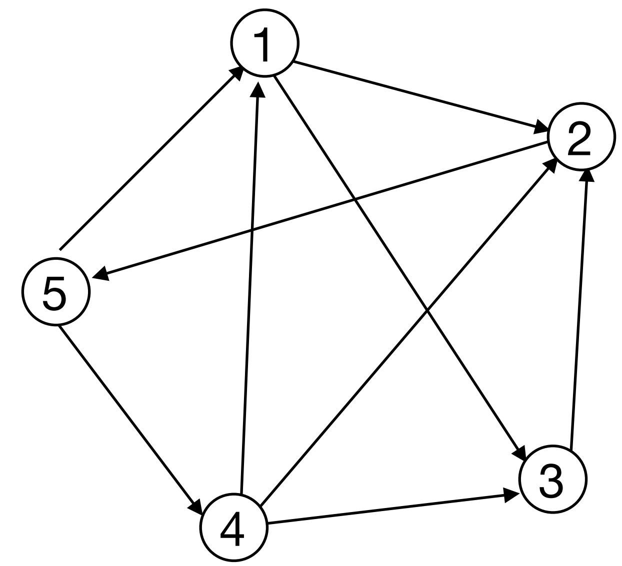
$$q^{(t+1)} = q^{(t-1)}P = q^{(0)}P^t$$

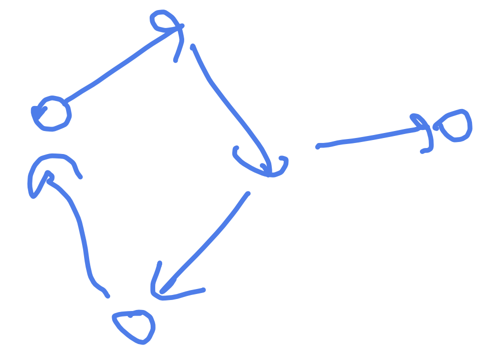
$$q^{(t+1)} = q^{(0)}P^t$$

$$q^{(t+1)} = \left[ q_1^{(t+1)}, q_2^{(t+1)}, q_3^{(t+1)}, q_4^{(t+1)}, q_5^{(t+1)} \right]$$

# Random walks on graphs — PageRank

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



Incorporating the random restart (often referred as random jump):
In each iteration with some probability we jump to any of the nodes uniformly at random.

$$P = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix} \qquad J = \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix}$$

$$P' = \alpha P + (1-\alpha) J \text{ where } \alpha \in [0,1]$$

general formula:

$$q^{(t+1)} = q^{(0)} (P')^t$$

# Random walks on graphs — PageRank

$$P = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{in} \\ P_{21} & \dots & & p_{2n} \\ \vdots & \ddots & & \vdots \\ P_{n1} & P_{n2} & \dots & P_{nn} \end{bmatrix} \qquad J = \begin{bmatrix} \frac{1}{n} & \frac{1}{n} & \dots & \frac{1}{n} \\ \frac{1}{n} & \dots & & \frac{1}{n} \\ \vdots & \ddots & & \vdots \\ \frac{1}{n} & \frac{1}{n} & \dots & \frac{1}{n} \end{bmatrix}$$

$P' = \alpha P + (1 - \alpha) J$ where $\alpha \in [0, 1]$

$q^{(t+1)} = q^{(0)} (P')^t$ → after t+1 steps

We get Pagerank as $t \to \infty$ → where am I on average at any time

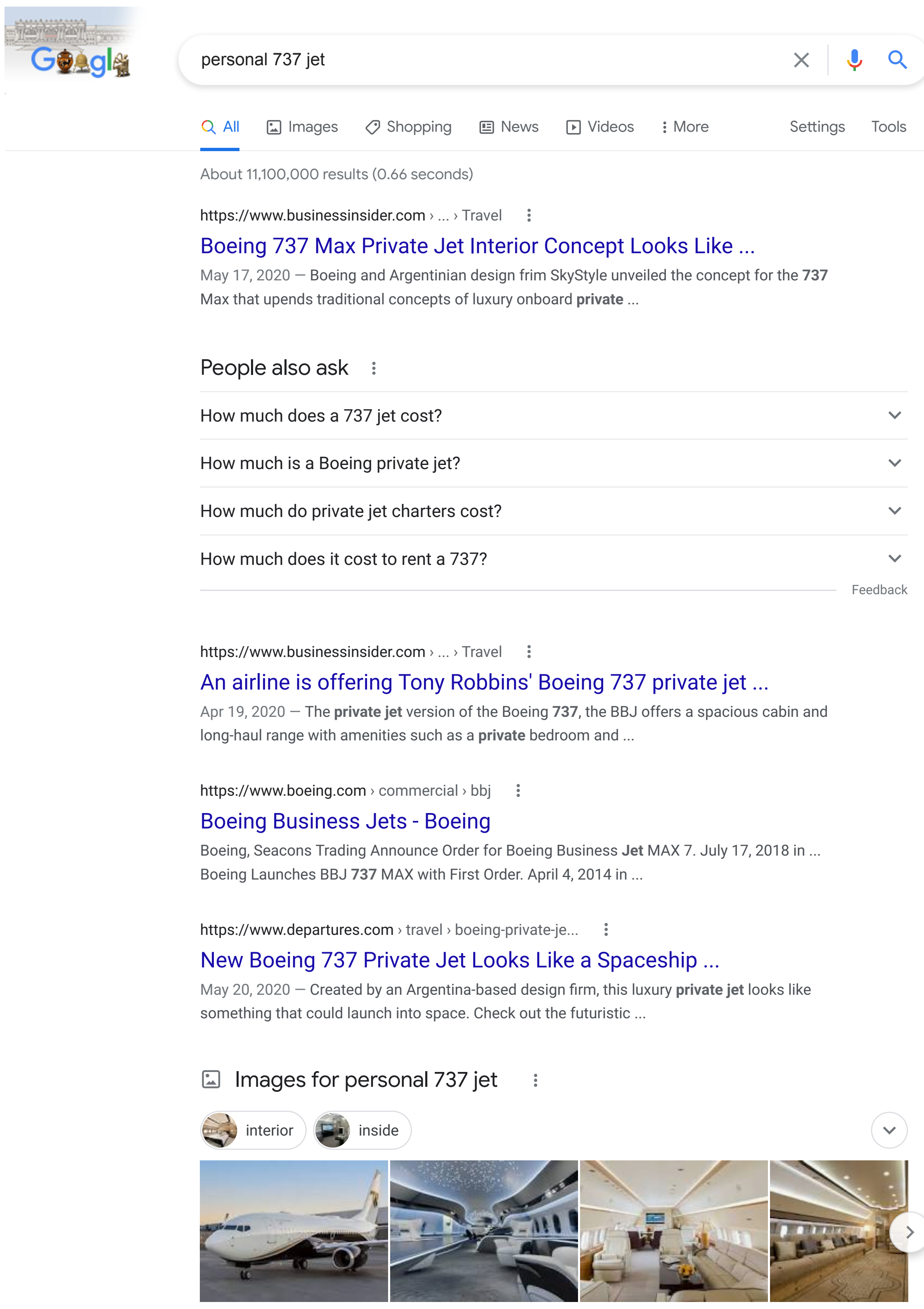
$\alpha$ controls the rate of convergence

variant: personalized PageRank

- instead if jumping to a uniform random location, we jump back to some specific starting node.

# Pagerank and the power method

Remember? we said that computing $(P')^t$ is inefficient, hence we use an iterative approach.




To compute $(P')^t$

$$(P')^t = V\Lambda^t V^T$$

For this we need to compute the eigenvalue decomposition of P'

- Remember, that takes computing the inverse matrix (say, for the largest eigenvalue)
- using Gaussian elimination that takes $O(\frac{2}{3}n^3)$

$$(P' - \lambda I)^{-1} v = 0$$

There are 11 million hits! lots of flops…

$$O(\frac{2}{3}n^3) \approx 11^{18} \cdot 10^{18} \approx 10^{36}$$

That takes $10^{27}$ seconds, is measured in months….

conclusion: use the power method instead….

# Markov Chains

- A Markov chain describes a discrete time stochastic process over a set of states

according to a transition probability matrix

$$S = \{s_1, s_2, \ldots s_n\}$$

  - $P_{ij}$ = probability of moving to state $j$ when at state $i$
    - $\sum_j P_{ij} = 1$ (stochastic matrix)

$$P = \{P_{ij}\}$$

- Memorylessness property: The next state of the chain depends only at the current state and not on the past of the process (first order MC)
  - higher order MCs are also possible

# Random Walk

- Random walks on graphs correspond to Markov Chains
  - The set of states S is the set of nodes of the graph G
  - The transition probability matrix P is the probability that we follow an edge from one node to another

# State probability

- The vector $q^0 = (q^0_1, q^0_2, \dots, q^0_n)$ that stores the probability of being at state $i$ at the start of the random walk
- $q^1 = q^0P$ is the probability of being in each node after one random step
- $q^t = q^{t-1}P$ is the probability of being at node $i$ after $t$ steps
- Can we generalize this?

# Stationary Distribution

- $q^t = q^{t-1}P$ is the probability of being at node $i$ after $t$ steps
- The stationary distribution is the probability of being in any given node = the fraction of time spent in each node
- as $t \longrightarrow \infty$ this doesn't change by taking another step: $q=qP$

prob. distribution: · we have all possible states (here: nodes)
· what is the prob of each specific location

1. $q = qP$ → $q$ is the eigenvector corresponding to $\lambda = 1$

$q = qP$

→ probability of being at location $i$ is fixed (in the limit)

→ ~~taking~~ taking one more step won't affect it

# Stationary Distribution

- A stationary distribution for a MC with transition matrix P, is a probability distribution $\pi$, such that $\pi = \pi P$

$\pi = [\pi_1, \pi_2, \ldots, \pi_n]$ = probability of being in states/nodes $1 \ldots n$

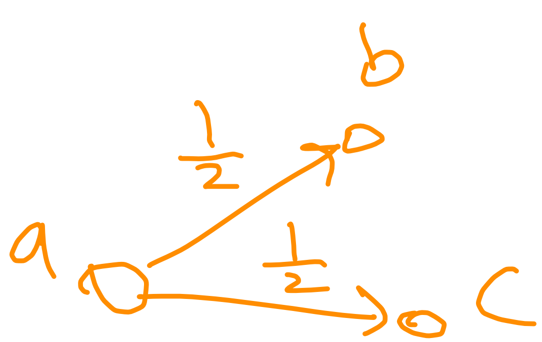
- $\pi$ is the eigenvector corresponding to eigenvalue 1
  - since P is stochastic 1 is its largest eigenvalue

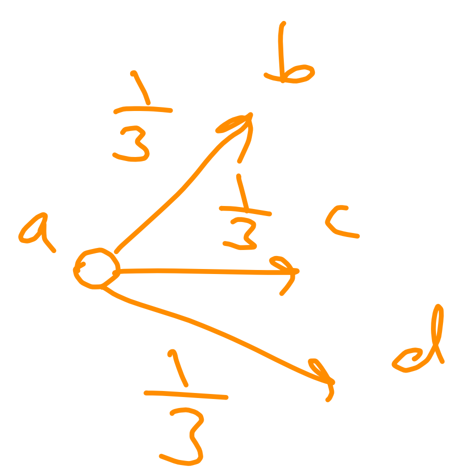
↳ a mtx corresponding to rnd values & rows sum to 1

- The stationary distribution exists if the MC is irreducible and aperiodic.
  - The underlying graph is strong connected and not bipartite

Does this apply to PageRank?

↳ if it has a stationary distr. that means that the relevance scores of each page are fixed




$$\frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1$$

# Undirected graphs — stationary distribution not very informative

$\lambda v = vA$

summary:

$q = qP$

pagerank score of each website

pagerank transition mtx

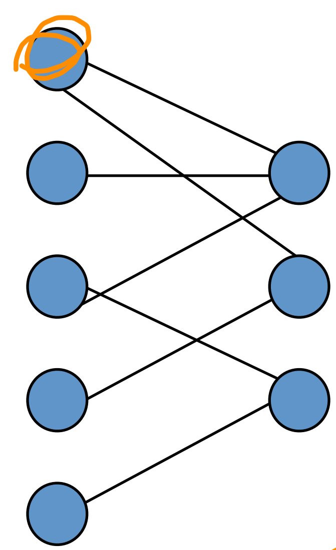
- connected undirected graph is always irreducible
- why not bipartite?
  - every other step the walker is not in a node with probability 1 (periodic)

strong connected:

↳ path from any node to any other

avoid:

⇒ rnd jump in pagerank takes care of this



in even iterations: always on the left

odd iters: always on the right

→ probabilities alternate between 0 and not 0 in each iteration

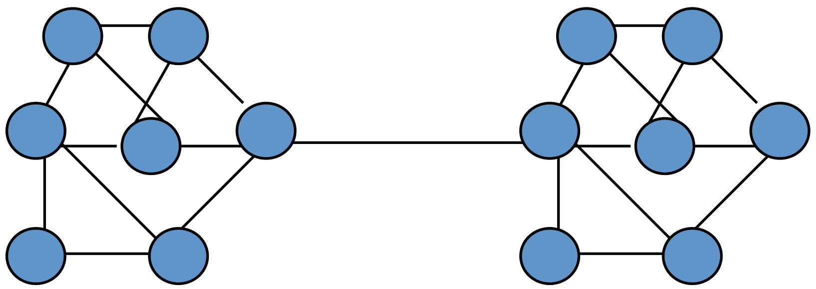
not stable ←

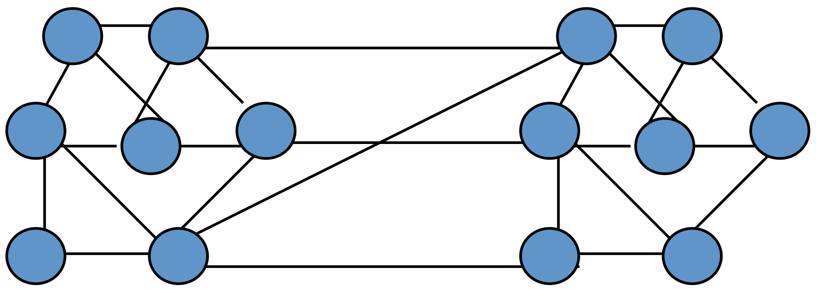
# Undirected graphs — stationary distribution not very informative

- connected undirected graph is always irreducible
- why not bipartite?
- In a connected and non-bipartite graph the stationary distribution is proportional to the degrees of the nodes

# Undirected graphs — stationary distribution not very informative

- connected undirected graph is always irreducible
- why not bipartite?
- In a connected and non-bipartite graph the stationary distribution is proportional to the degrees of the nodes
- mixing time: the rate of convergence





high

low