Top Ten Algorithms Class 3

John Burkardt
Department of Scientific Computing
Florida State University

.....

http://people.sc.fsu.edu/~jburkardt/classes/tta_2015/class03.pdf

14 September 2015

Our Current Algorithm List

1. Bernoulli number calculation
2. Euclid's greatest common factor algorithm
3. Pancake flipping algorithm for genome relations
4. Search engine indexing

Chaolun Wang: Prototeins

Proteins can be described as a linear string of molecules.

However, a linear protein will automatically fold up into a tangled structure, because positive and negatively charged molecules attract each other.

The shape of the protein is often a key to understanding its function in the body.

To understand existing proteins and design new ones, it is necessary to simulate the process by which the linear protein folds.

- Brian Hayes, "Prototeins", American Scientist.
- Dill, MacCallum, "The Protein-Folding Problem, 50 Years On", Science, 23 November 2012.
- "Folding at home", https://folding.stanford.edu/

Brett-Michael Green: The Judge is an Idiot

By a terrible mistake, I have been asked to judge a competition in Nanology... but I know nothing about Nanology. There is a room with 100 Nanology scholars, and I need to award first, second and third prizes. There is some room for argument, but if I award really bad people, I will get in trouble.

I get an inspiration, and ask every scholar to point to the two other scholars in the room that they most respect. Unfortunately, most scholars only know a few people, and so everyone is pointing to people nearby them in the room, so this brilliant idea gives me lots of information, but I don't immediately see how to use it.

Is there a way to fake good judgment, that is, to make a good guess as to who are the most respected scholars of Nanology?

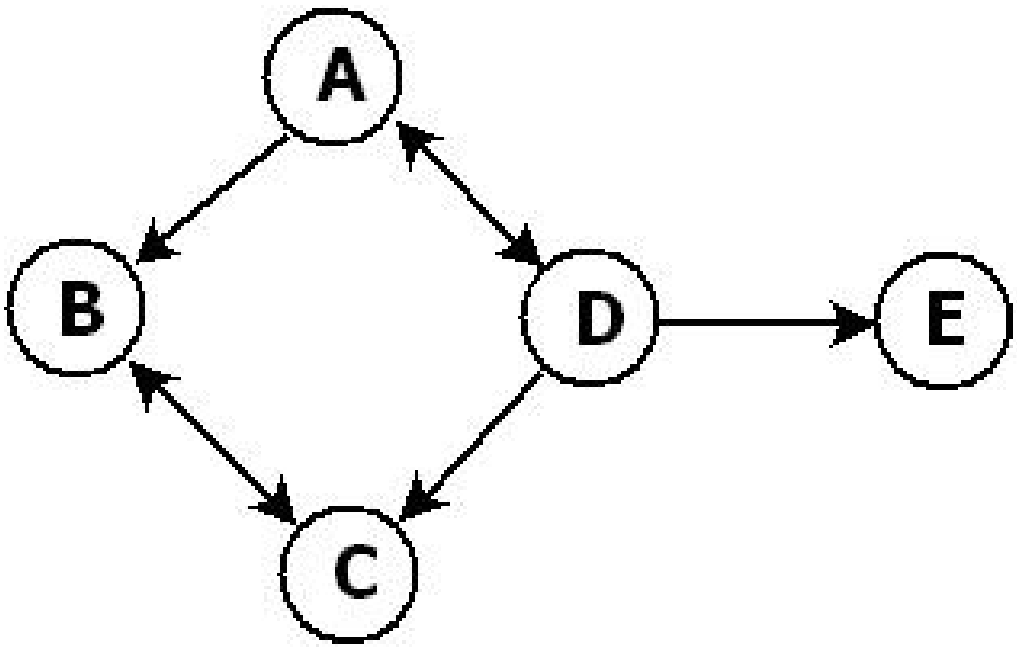
An answer to this question will help us to decide how to rank the pages that match a search engine query.

A Directed Graph

A directed graph or "digraph" describes one-way connections between pairs of objects.

The objects are "nodes" or "vertices"; We usually label them 1, 2, 3... or A, B, C...

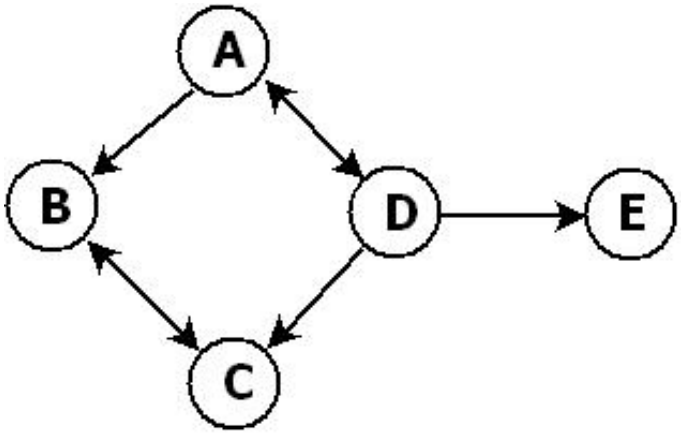
The connections are "edges", "links", or "arcs". We describe them by the nodes they connect, *(from,to)*, as in (1,7) or (D,E).



Representing a Directed Graph

If we need to do computations, we can represent a digraph by:

- an edge list: (A,B), (B,C), (C,B), (D,A), (D,C), (D,E);
- incidence list: (A:B), (B:C), (C:B), (D:A,C,E), (E:);
- an adjacency matrix A(i,j) = 1 if you can get from node I to node J;

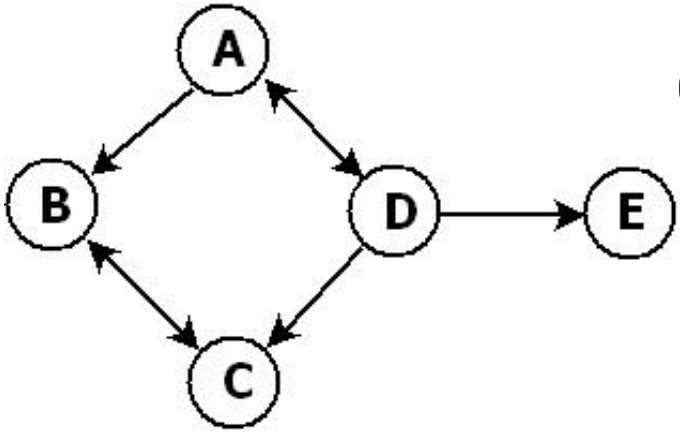


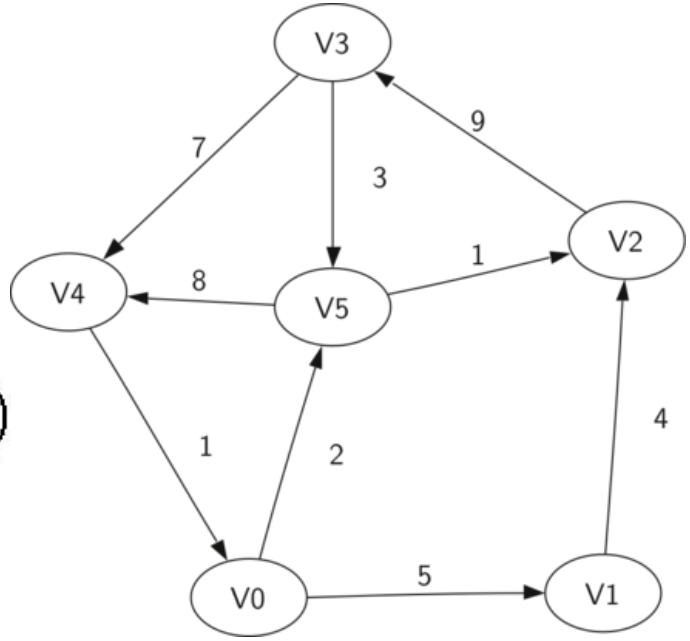
$$A = \begin{bmatrix} . & 1 & . & 1 & . \\ . & . & 1 & . & . \\ . & 1 & . & . & . \\ 1 & . & 1 & . & 1 \\ . & . & . & . & . \end{bmatrix}$$

Getting from A to B

The adjacency matrix only tells us whether two nodes A and B are directly connected. But of course, some nodes may be connected if we are allowed to use several links in a row.

- a path from A to B is a sequence of links that connect A to B.
- a circuit is a path from A to A.
- the length of a path or circuit is the number of links.
- a digraph is strongly connected if there is a path from A to B for every pair of nodes A and B.



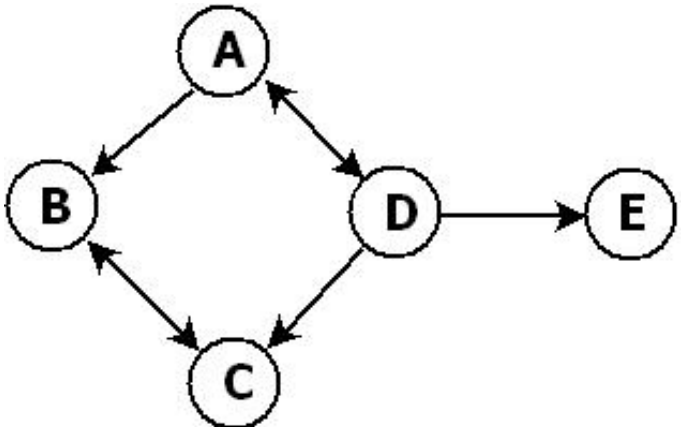


The Path Counting Algorithm

The adjacency matrix essentially counts the number of paths of length 1 from node I to node J. (It's 1 or 0.) How many paths of length 2 are there?

When I was told all you did was compute A^2, I didn't believe it.

$$A = \begin{bmatrix} . & 1 & . & 1 & . \\ . & . & 1 & . & . \\ . & 1 & . & . & . \\ 1 & . & 1 & . & 1 \\ . & . & . & . & . \end{bmatrix} \quad A^2 = \begin{bmatrix} 1 & 0 & 2 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad A^3 = \begin{bmatrix} 0 & 3 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 3 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



A(I,J) - can I get to J from I?

Consider a board game with n consecutive squares. The player starts on square 1, rolls a single die and then moves. This means that square 1 has a one-way connection to squares 2, 3, 4, 5, 6, and 7, and so on for each square. Once the player has moved to a new square k, we see that it is connected to squares $k+1$ through $k+6$. We can model that relationship with an adjacency matrix whose first few rows look like this:

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

T(I,J) - will I get to I from J?

Think of the edges as probabilities that we will transition from one edge to another. Normalize the probabilities, and transpose the adjacency matrix to get the transition matrix T. $T(i,j)$ is the probability that we will move to node i from node j.

$$T = \begin{bmatrix}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\frac{1}{6} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\frac{1}{6} & \frac{1}{6} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \cdot & \cdot & \cdot & \cdot & \cdot \\
\frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \cdot & \cdot & \cdot & \cdot \\
\frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \cdot & \cdot & \cdot \\
\cdot & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \cdot & \cdot \\
\cdot & \cdot & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{bmatrix}$$

Board Games

Although I can't show the complete matrix here, you should convince yourself that every column of this matrix should sum up to 1, because if I am at node J now, then after one roll of the die, there is a 100% probability that I will be somewhere.

A matrix like this, of nonnegative values, each of whose columns sum to 1, has the column stochastic property.

The transition matrix will allow us to "compute" movements in the game probabilistically.

If we start at square 1, we will symbolize this fact with a vector x of length 100, which is all 0, except that $x[1] = 1$. Then making a single move can be "computed" by computing $x \leftarrow T * x$.

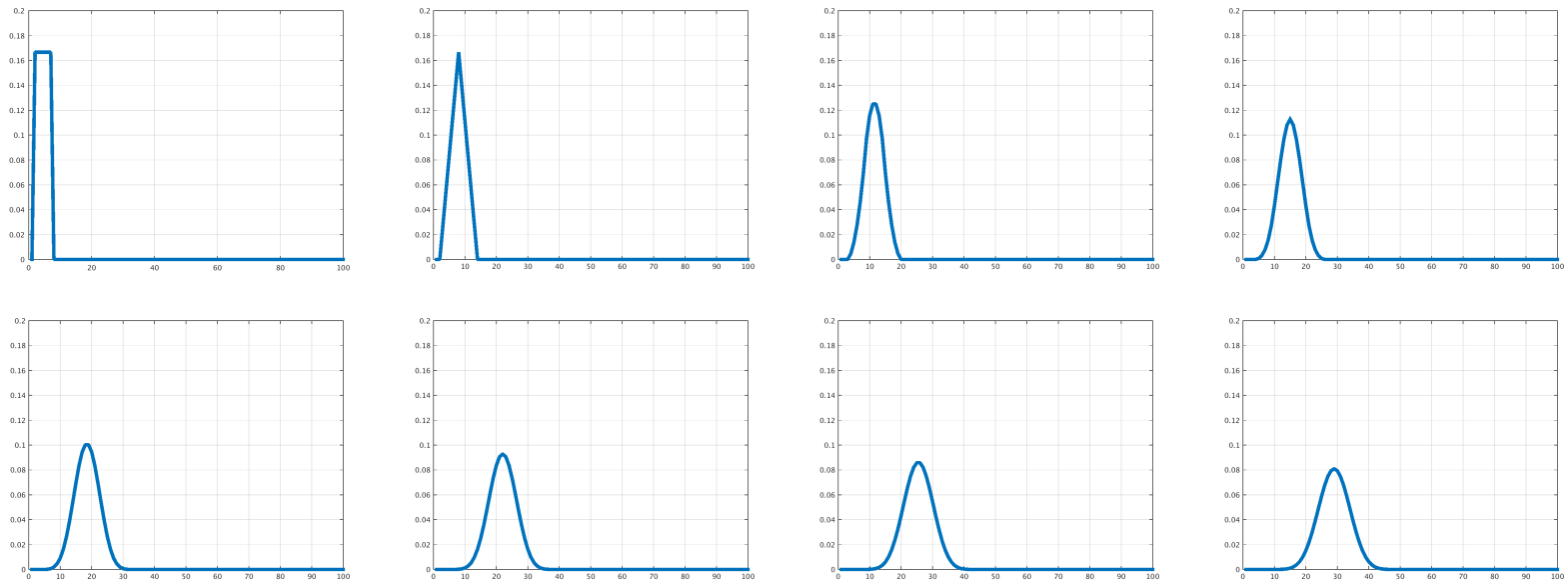
Board Games

Here is a table of the first ten entries in our vector, as we take successive moves:

x	Tx	T^2x	T^3x	T^4x
1	0.000	0.000	0.000	0.000
0	0.167	0.000	0.000	0.000
0	0.167	0.028	0.000	0.000
0	0.167	0.056	0.005	0.000
0	0.167	0.083	0.013	0.001
0	0.167	0.111	0.028	0.003
0	0.167	0.139	0.046	0.008
0	0.000	0.167	0.065	0.015
0	0.000	0.139	0.097	0.027
0	0.000	0.111	0.116	0.043

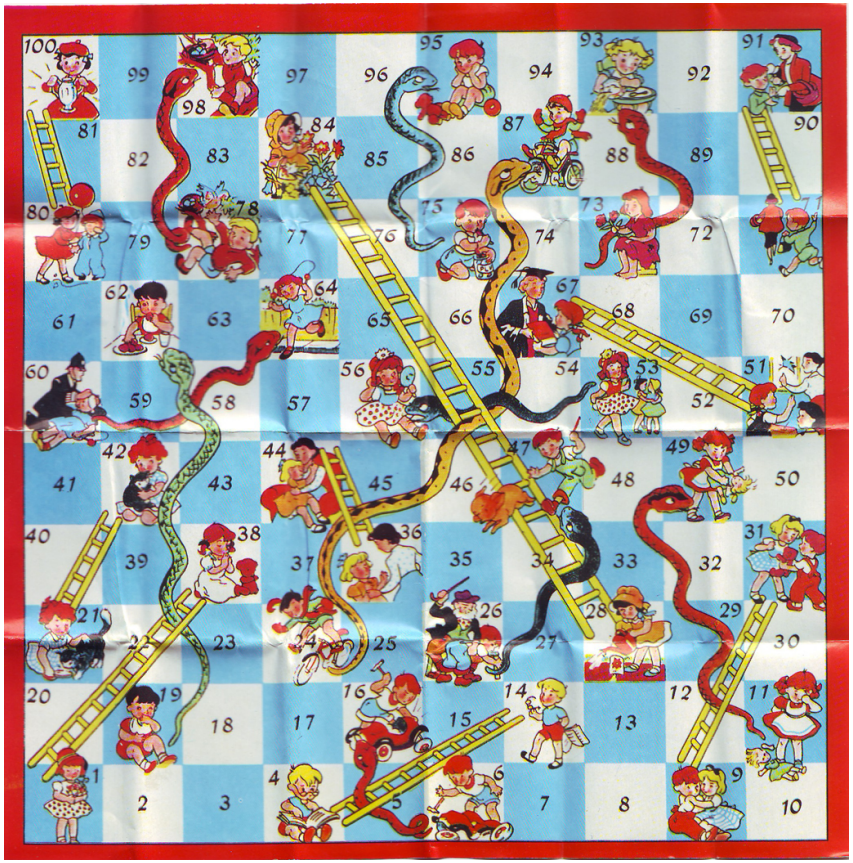
Board Games - Probability Distribution For First 8 Moves

Each time we replace x by $T * x$, we are getting the probabilities of the next state.



More Interesting Simulations

This is a very simple example of Monte Carlo analysis. See Nick Berry's blog articles "Markov Chains, Monte Carlo, and Chutes and Ladders" or "Mathematical Analysis of the Game of Candyland" to see how to determine the average length of a game.



A Steady Circuit

Board games come to an end. However, when we look at a digraph, one of the most interesting questions we can ask is: *"Can this digraph sustain a steady flow?"*

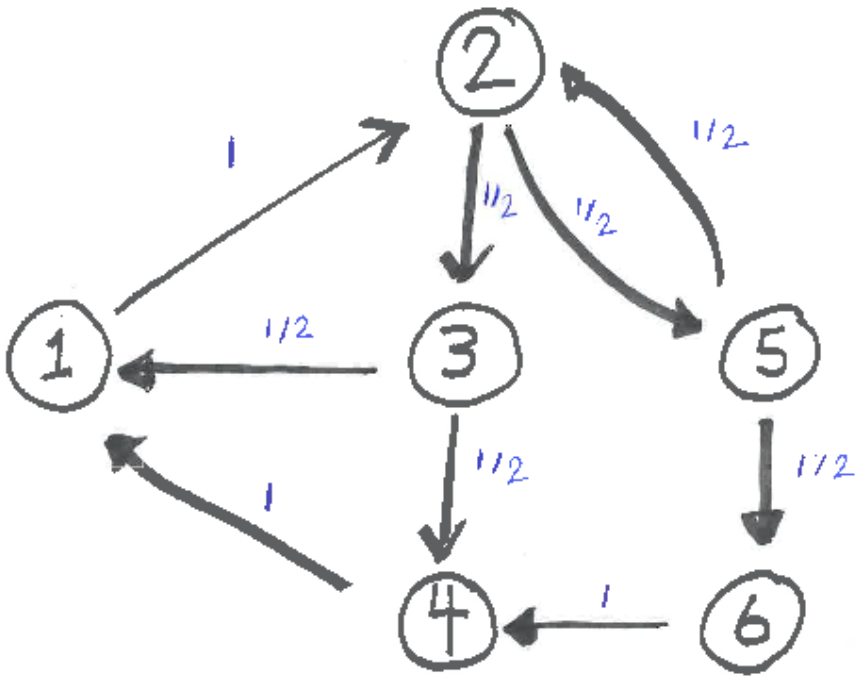
A steady flow would be an assignment of an x value to every node, so that $T * x$ changes nothing. Mathematically, $T * x = x$. This is an eigenvalue problem, where x, the steady flow, is an eigenvector, and the corresponding eigenvalue is 1.

The board game we just discussed can't sustain a steady flow, because the game ends when we reach square 100. But if we let the board wrap around, we'd have a very simple steady flow, by putting a 1 at every location. After rolling the die, each 1 would split up into 6 equal parts of 16 and advance 1, 2, 3, 4, 5 or 6 squares, but we would end up with the same configuration at the end.

An example

Here is a digraph that can sustain a steady flow.

It is strongly connected. (From 1 we can reach 2. From 2 we can reach 3 ... from 6 we can reach 1)



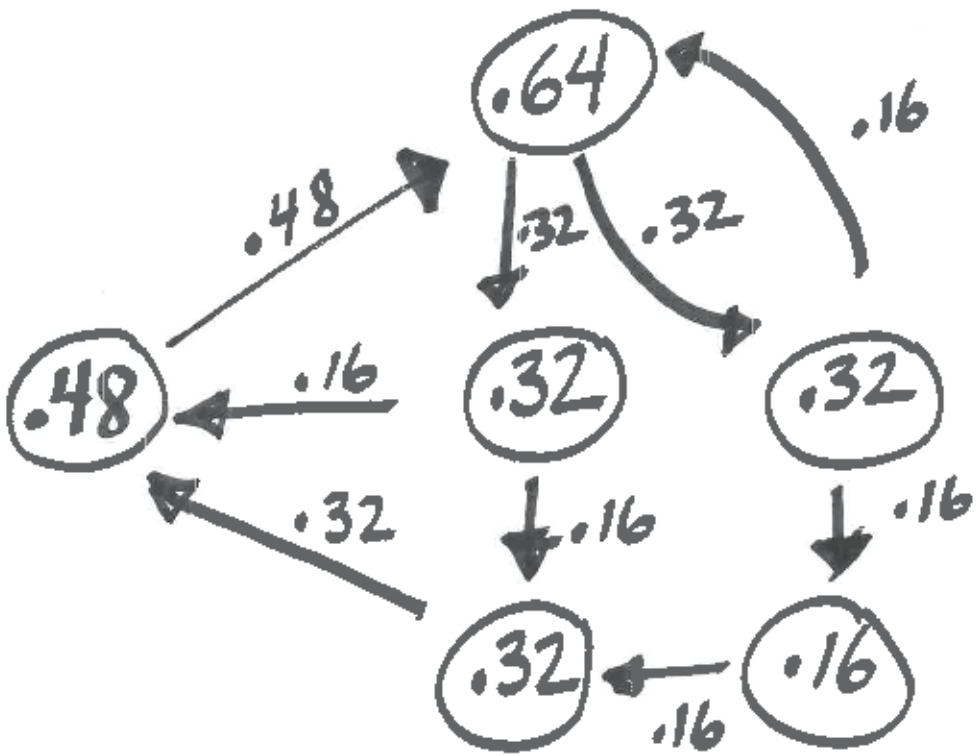
An example: Adjacency, Transition, Eigenvector

$$A = \begin{bmatrix} . & 1 & . & . & . & . \\ . & . & 1 & . & 1 & . \\ 1 & . & . & 1 & . & . \\ 1 & . & . & . & . & . \\ . & 1 & . & . & . & 1 \\ . & . & . & 1 & . & . \end{bmatrix} \quad T = \begin{bmatrix} 0.0 & 0.0 & 0.5 & 1.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 \\ 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.5. & 0.0 \end{bmatrix}$$

x	$T * x$
0.4867	0.4867
0.6489	0.6489
0.3244	0.3244
0.3244	0.3244
0.3244	0.3244
0.1622	0.1622

An example: The flow in action

Here is the digraph, with the amounts shown at the nodes. A node with one outgoing link simply transfers everything to the next node. A node with two outgoing links gives half to each of its neighbors.



Perron-Frobenius Theorem

Suppose a directed graph is strongly connected.

- Then the transition matrix T has a simple eigenvalue of 1;
- All other eigenvalues are strictly smaller;
- There is a corresponding strictly positive eigenvector x.

So...to find a flow on the directed graph, we solve the eigenvalue problem $T * x = x$.

Mathematically, this sounds easy now, but there are some important issues!

- What happens if the matrix T is enormous?;
- If T is really big, we can't call lapack or eigs or arpack;
- The graph might not be strongly connected.

Next Week (Student volunteer?)

I have enough money to hire one student, but the university insists that I have to do this in a fair way. I don't know how many people are standing in line outside my office, but I'm going to see each of them, one at a time. There is a chair in my office in which one person can sit. When a candidate walks into my office, I have two choices:

- tell the candidate, "I'm sorry, you don't get the job."
- tell the seated person, "Please leave, and let the new guy sit there."

Once the line has ended, the person occupying the chair gets the job.

How can I carry out this process in a fair way?

See Nick Berry, "Selecting a random person (fairly) from an unknown number of applicants", June 2013.

This is an example of a data stream algorithm.

Next Week (Student volunteer?)

Pretend you are the Google search engine, and that you want to publish the most popular search topic of the week. So on Monday you start "noticing" all the search words, and on Sunday at midnight you announce the winner.

Since there are trillions of possible topics, and billions of searches, can you think of an algorithm that will get the right answer, without requiring a huge amount of memory and time?

This is another example of a data stream algorithm, from the area of "big data".

See Brian Hayes, "The Britney Spears Problem", American Scientist, July-August 2008.