26. Computing the Rank of a Webpage

Google PageRank
More Practice with 2D Array OPs
More Practice with numpy

Functions and 2D Arrays

Assume

```
from random import uniform as randu
from numpy import *
```

Let's write a function **randuM(m,n)** that returns an m-by-n array of random numbers, each chosen from the uniform distribution on [0,1].

A Function that Returns an n-by-n Array of Random Numbers

```
def randuM(m,n):
    A = zeros((m,n))
    for i in range(m):
        for j in range(n):
            A[i,j] = randu(0,1)
    return A
```

Probability Arrays

A nxn probability array has the property that its entries are nonnegative and that the sum of the entries in each column is 1

.2	.6	.2
.7	.3	.3
.1	.1	.5

Probability Arrays

To generate a random probability array, generate a random matrix with nonnegative entries and then divide the numbers in each column by the sum of the numbers in that column

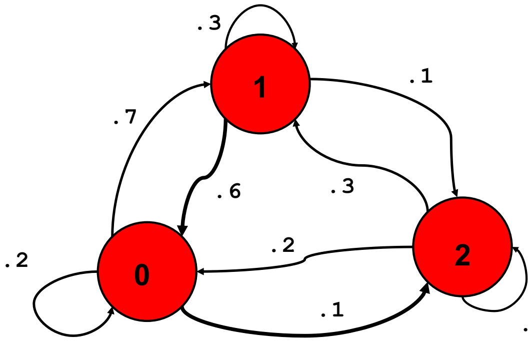
5	6	1
2	0	3
4	3	1

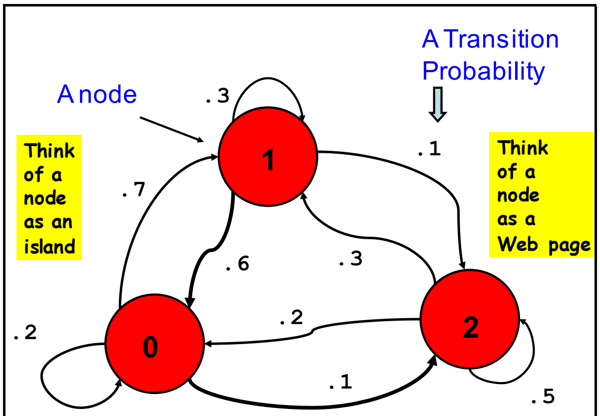
5/11	6/9	1/5
2/11	0/9	3/5
4/11	3/9	1/5

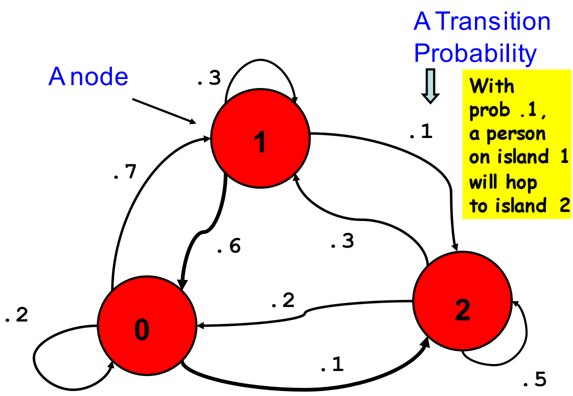
A Function that Returns a Random Probability Array

```
def probM(n):
    A = randuM(n,n)
    for j in range(n):
        # Normalize column j
        s = 0;
        for i in range(n):
            s += A[i,j]
        for i in range(n):
            A[i,j] = A[i,j]/s
    return A
```

Here is a Network





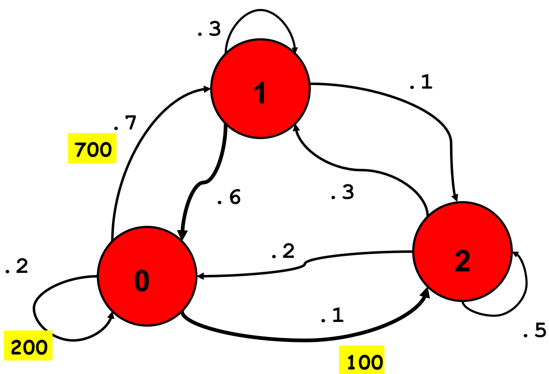


A Random Process

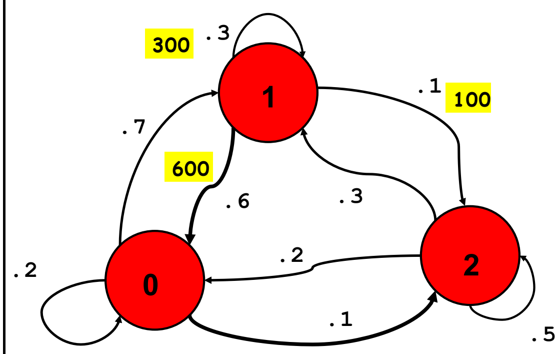
Suppose there are a 1000 people on each node.

At the sound of a whistle they hop to another node in accordance with the "outbound" probabilities.

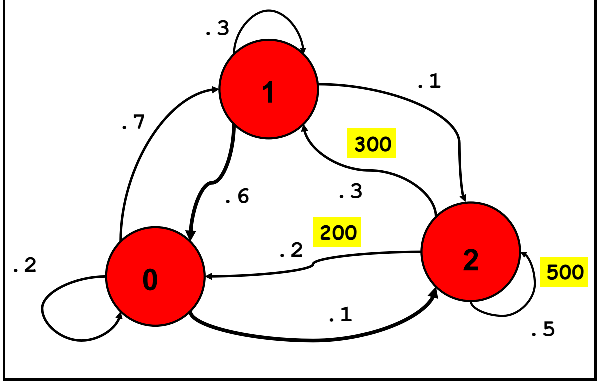
At Node 0



At Node 1



At Node 2



The Population Distribution

	Before	After
Node 0	1000	1000
Node 1	1000	1300
Node 2	1000	700

Repeat

	Before	After
Node 0	1000	1120
Node 1	1300	1300
Node 2	700	580

After 100 Iterations

	Before	After
Node 0	1142.85	1142.85
Node 1	1357.14	1357.14
Node 2	500.00	500.00

Appears to reach a Steady State

After 100 Iterations

	Before	After
Node 0	1142.85	1142.85
Node 1	1357.14	1357.14
Node 2	500.00	500.00

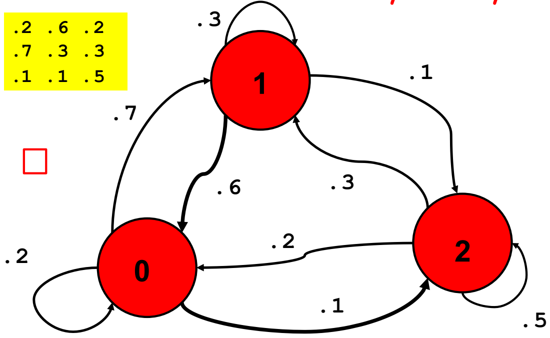
In terms of popularity: Island 1 > Island 0 > Island 2

After 100 Iterations

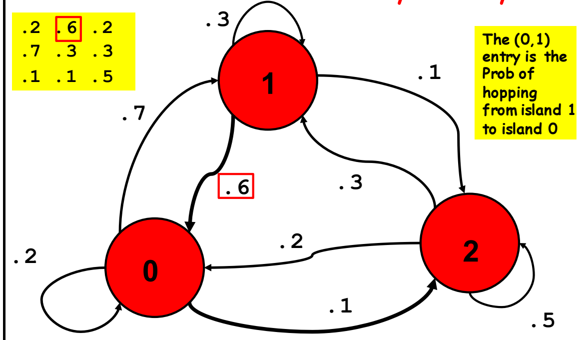
	Before	After
Node 0	1142.85	1142.85
Node 1	1357.14	1357.14
Node 2	500.00	500.00

[1142.85, 1357.14, 500.0] is the “stationary array”

Computing the Stationary Array Involves a Probability Array



Computing the Stationary Array Involves a Probability Array



Transition Probability Array

P:

.2	.6	.2
.7	.3	.3
.1	.1	.5

P[i,j] is the probability of hopping from node j to node i

Formula for Updating the Distribution Array

P =

.2	.6	.2
.7	.3	.3
.1	.1	.5

```
w[0] = .2*v[0] + .6*v[1] + .2*v[2]
w[1] = .7*v[0] + .3*v[1] + .3*v[2]
w[2] = .1*v[0] + .1*v[1] + .5*v[2]
```

V is the old distribution array,
w is the updated distribution array

Formula for Updating the Distribution Vector

P =

.2	.6	.2
.7	.3	.3
.1	.1	.5

```
w[0] = P[0,0]*v[0] + P[0,1]*v[1] + P[0,2]*v[2]
w[1] = P[1,0]*v[0] + P[1,1]*v[1] + P[1,2]*v[2]
w[2] = P[2,0]*v[0] + P[2,1]*v[1] + P[2,2]*v[2]
```

V is the old distribution vector,
w is the updated distribution vector

A Function that Computes the Update

```
def Update(P,v):
    n = len(x)
    w = zeros((n,1))
    for i in range(n):
        for j in range(n):
            w[i] += P[i,j]*v[j]
    return w
```

Back to PageRank

Background

Index all the pages on the Web from 0 to N-1. (N is around 50 billion.)

The PageRank algorithm orders these pages from "most important" to "least important".

It does this by analyzing links, not content.

Key Ideas

The Transition Probability Array

A Very Special Random Walk

The Connectivity Array

A Random Walk on the Web

Repeat:

You are on a webpage.
There are m outlinks.
Choose one at random.
Click on the link.

The Connectivity Array

G[i,j] is 1 if there is a link on page j to page i

$$G: \begin{matrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{matrix}$$

The Probability Array

a = 1/3

b = 1/2

c = 1/4

$$P: \begin{matrix} 0 & a & 0 & 0 & b & 0 & c & 0 \\ a & 0 & 0 & 0 & 0 & 0 & c & 1 \\ 0 & a & 0 & 0 & b & 0 & 0 & 0 \\ a & 0 & a & b & 0 & a & 0 & 0 \\ 0 & 0 & 0 & b & 0 & 0 & c & 0 \\ 0 & a & a & 0 & 0 & a & 0 & 0 \\ a & 0 & 0 & 0 & 0 & 0 & c & 0 \\ 0 & 0 & a & 0 & 0 & a & 0 & 0 \end{matrix}$$

PageRank From the Stationary Array

Stationary Array	PageRank
0.5723	3
0.8206	1
0.7876	2
0.2609	5
0.2064	7
0.8911	0
0.2429	6
0.4100	4

Webpage 5
Has pageRank
0