ASSIGNMENT 1

1. It requires a certain amount of theory and mathematical knowledge.
2. Let's go over the theory right now.
3. Start early

Google PageRank Algorithm (simplified)

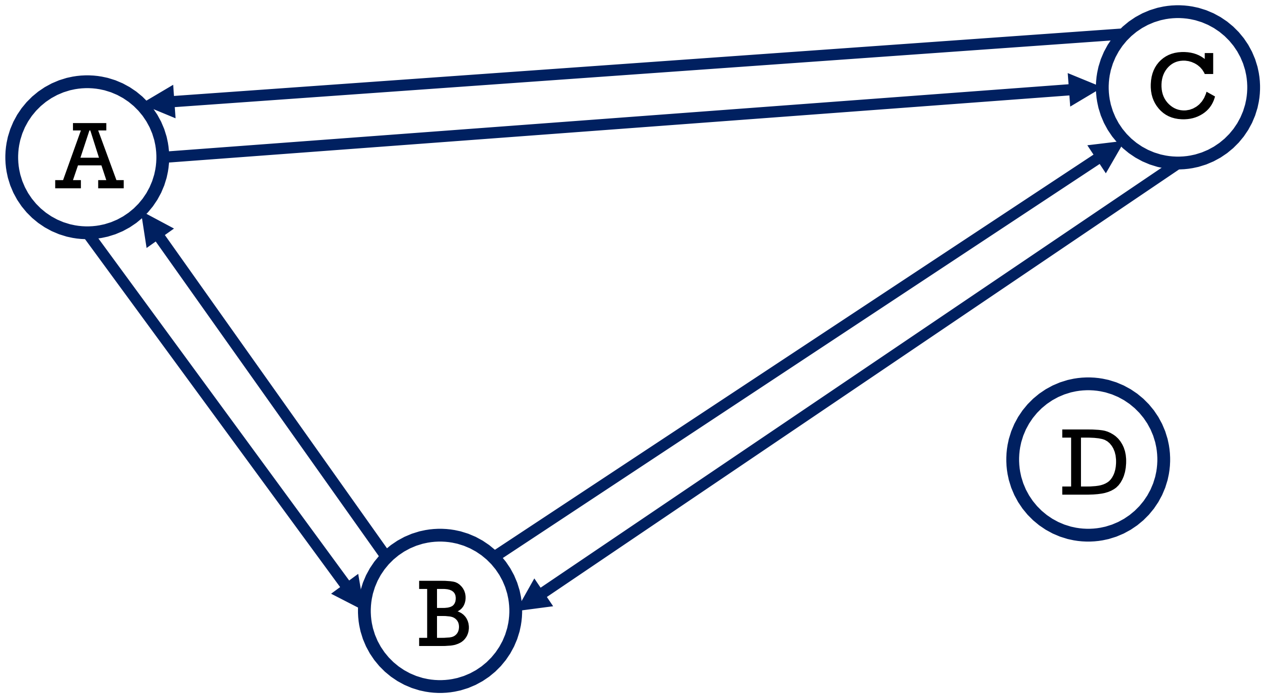
- Great use of the **matrix** and **linear algebra**
- **PROBLEM – Lots of web pages on web, how to rank them?**
- More links to a page, the higher the rank
- Find out how web pages link to each other (connectivity matrix)
- Find out chance to access web pages relative to each other (importance matrix)
- Include non linked pages (stochastic matrix)

Google PageRank Algorithm (simplified)

- Add user randomness into our stochastic matrix
  - User click links with 85% chance
  - User teleports to sites with 15% chance
- This becomes an n x n **transition matrix**
- Multiply it with a column vector n x 1 repeatedly until the column vector stops changing
- Compare the result with all other sites and get the ranking!

1. Start with a web

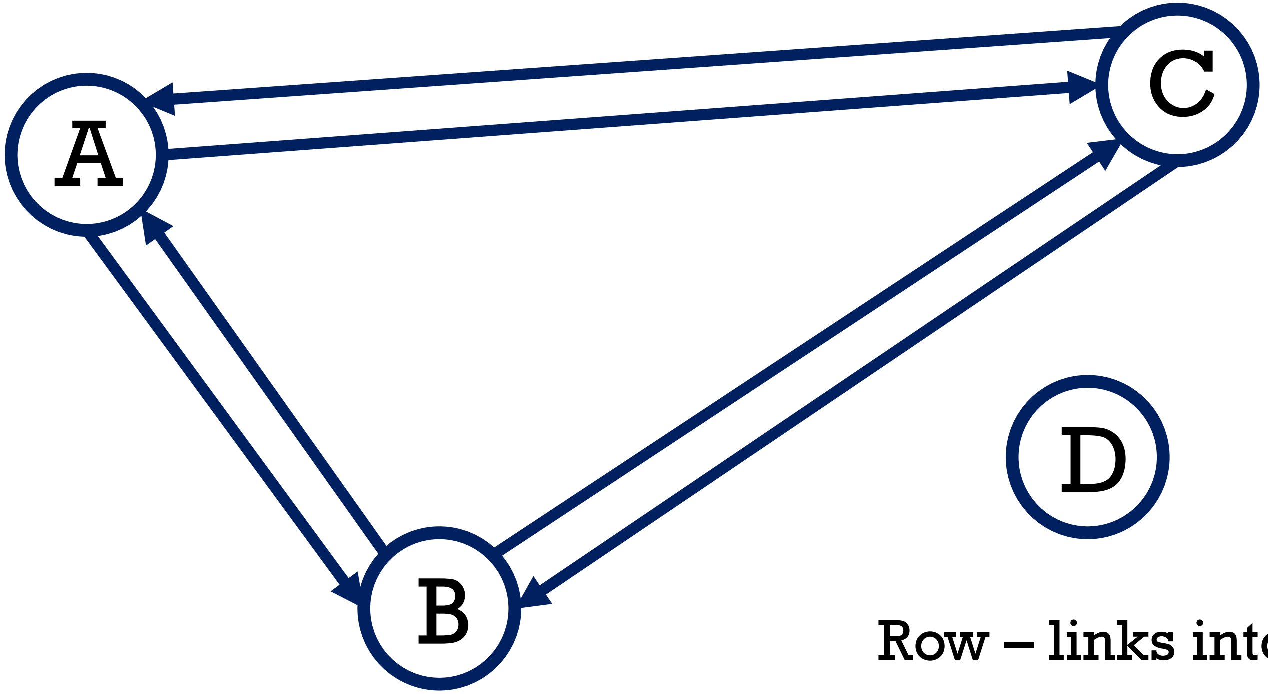
Let W be a set of webpages of size n



W =
{
Apple,
Bell,
Cisco,
Dropbox
}

Sizeof(W) = 4

2. Our connectivity matrix G

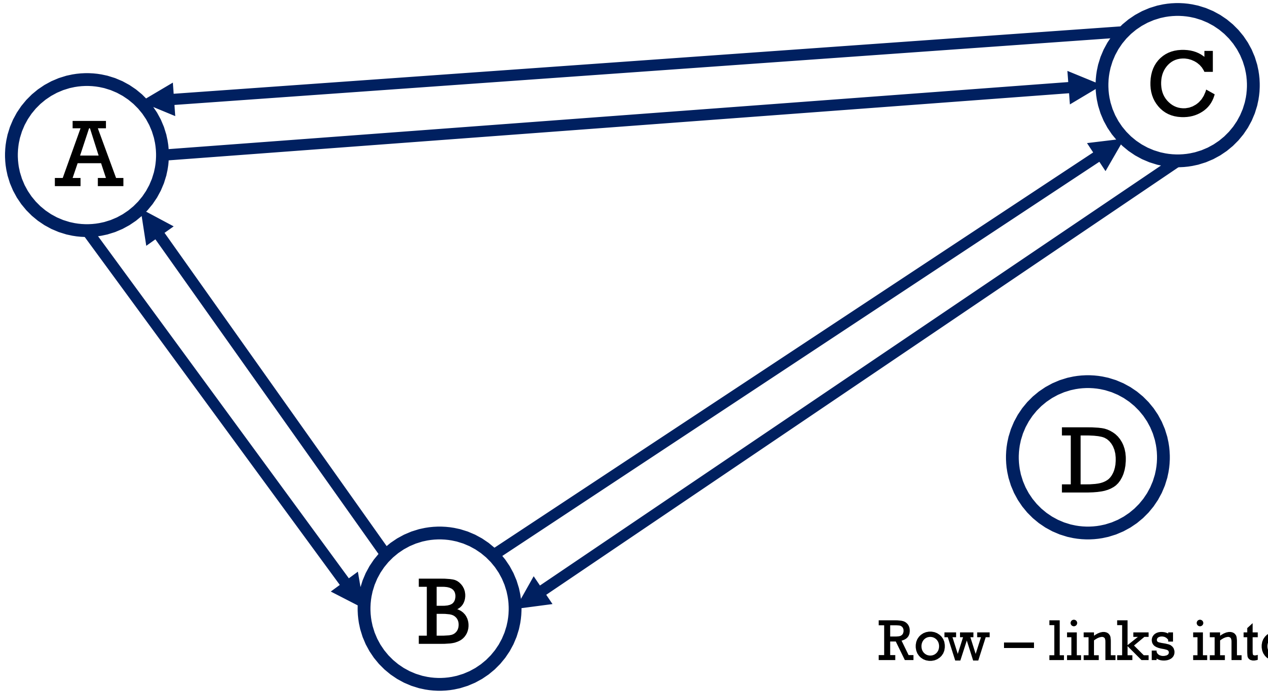


Column – links out from

Row – links into

	A	B	C	D
A				
B				
C				
D				

2. Our connectivity matrix G



Column – links out from

Row – links into

	A	B	C	D
A	0	1	1	0
B	1	0	1	0
C	1	1	0	0
D	0	0	0	0

3. Degree

	A	B	C	D
A	0	1	1	0
B	1	0	1	0
C	1	1	0	0
D	0	0	0	0

In-degree $\mathbf{r}_i$ is the number of 1s in row i.

Out-degree $\mathbf{c}_j$ is the number of 1s in column j.

4. Importance matrix S

We can modify our connectivity matrix to show us this ``importance'' if we **divide each value in each column by the sum of each column.**

Think of this as a "normalized" version of the previous matrix where values are between [0,1]

	A	B	C	D
A	0	**0.5**	**0.5**	0
B	**0.5**	0	**0.5**	0
C	**0.5**	**0.5**	0	0
D	0	0	0	0

5. Importance matrix **S** (what about site D?)

We can modify our connectivity matrix to show us this ``importance'' if we divide each value in each column by the sum of each column.

Total 4 web pages.

D equal random chance to go to all. 1 / 4 = 0.25

	A	B	C	**D**
A	0	0.5	0.5	**0.25**
B	0.5	0	0.5	**0.25**
C	0.5	0.5	0	**0.25**
D	0	0	0	**0.25**

6. Stochastic matrix S = Probability matrix S

- Called a "left stochastic matrix" because
  - All columns add to 1
  - All elements are [0, 1]

	A	B	C	D
A	0	0.5	0.5	**0.25**
B	0.5	0	0.5	**0.25**
C	0.5	0.5	0	**0.25**
D	0	0	0	**0.25**
	=1	=1	=1	=1

7. Introduce concept of randomness

We need to introduce the notion of a random walk

We need to multiply our probability matrix by a **random walk probability factor**

For our assignment, we will designate this variable **p**, and set **p = 0.85**.

```
double p{0.85};
```

7. Introduce concept of randomness + teleport

p = 0.85 //probability we'll follow the previous matrix

1-0.85 = 0.15 //probability we won't follow the matrix

0.15 chance we'll **teleport** to another site

- Don't follow link, enter address in address bar

8. Create our transition matrix M

Equal chance to go to any page with **teleportation.**
Q is an n x n matrix in which each element is 1/n
We have 4 web pages so 1 / 4 = 0.25

Q =

0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25

8. Create our transition matrix M

M = (probability click links) + (probability teleport)

M = 0.85 * **S** + (1 - 0.85) * **Q**

$$\mathbf{M} = 0.85 * \begin{bmatrix} 0 & 0.5 & 0.5 & 0.25 \\ 0.5 & 0 & 0.5 & 0.25 \\ 0.5 & 0.5 & 0 & 0.25 \\ 0 & 0 & 0 & 0.25 \end{bmatrix} + 0.15 * \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

8. Create our transition matrix M

$$\mathbf{M} = \begin{bmatrix} 0.0375 & 0.4625 & 0.4625 & 0.25 \\ 0.4625 & 0.0375 & 0.4625 & 0.25 \\ 0.4625 & 0.4625 & 0.0375 & 0.25 \\ 0.0375 & 0.0375 & 0.0375 & 0.25 \end{bmatrix}$$

9. Create a column matrix **rank** of size n x 1

Column matrix **rank**

1.0
1.0
1.0
1.0

10. The Markov Process

Multiply the transition matrix **M** by our matrix **rank**, and then multiply **M** by the result and then keep doing this until the rank stops changing (**result converges**), e.g., M * rank = rank2.

M * rank = **rank2**

$$\begin{bmatrix} 0.0375 & 0.4625 & 0.4625 & 0.25 \\ 0.4625 & 0.0375 & 0.4625 & 0.25 \\ 0.4625 & 0.4625 & 0.0375 & 0.25 \\ 0.0375 & 0.0375 & 0.0375 & 0.25 \end{bmatrix} * \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \end{bmatrix} = \begin{bmatrix} 1.2125 \\ 1.2125 \\ 1.2125 \\ 0.3625 \end{bmatrix}$$

10. The Markov Process

Multiply the transition matrix **M** by our matrix **rank**, and then multiply **M** by the result and then keep doing this until the rank stops changing (**result converges**), e.g., M * rank2 = rank3.

M * **rank2** = **rank3** //repeat until results converge

$$\begin{bmatrix} 0.0375 & 0.4625 & 0.4625 & 0.25 \\ 0.4625 & 0.0375 & 0.4625 & 0.25 \\ 0.4625 & 0.4625 & 0.0375 & 0.25 \\ 0.0375 & 0.0375 & 0.0375 & 0.25 \end{bmatrix} * \begin{bmatrix} 1.2125 \\ 1.2125 \\ 1.2125 \\ 0.3625 \end{bmatrix} = \begin{bmatrix} ??? \\ ??? \\ ??? \\ ??? \end{bmatrix}$$

10. The Markov Process

Multiply the transition matrix **M** by our matrix **rank**, and then multiply **M** by the result and then keep doing this until the rank stops changing (**result converges**), e.g., M * rankX = rankY. In this case, we get:

1.2698
1.2698
1.2698
0.1905

11. And finally

Divide each element in rank by the sum of the values in rank (scale rank so its elements sum to 1):

```
rank = 1.2698 / 3.999 = 0.3175 A
       1.2698 / 3.999 = 0.3175 B
       1.2698 / 3.999 = 0.3175 C
       0.1905 / 3.999 = 0.0476 D
```

And that’s that!

- The result makes intuitive sense
- Each of pages A, B, and C has a rank of about 32%, and page D ranks fourth with about 5%
- Keeping in mind that we haven't considered how a user's query will affect the rank, **you now understand how Google's PageRank* works**.

*a simplified version

PRO TIP: Break the assignment down

- Fully understand the assignment
- Focus on creating the matrix before any of the algorithm steps
- Create a class to represent a 2D matrix of any size
  - Probably with 2D vectors
- Add functionality to manipulate matrices:
  - Change the value of individual matrix cells
  - multiply a float to the entire matrix
  - add two n x n matrices together
  - multiply differently sized matrices together

PRO TIP: Break the assignment down

- Once your matrix class fully works with all its math operations, then start on the algorithm
- Break each algorithm step into smaller components
- Possibly multiple function calls per algorithm step
  - Instantiate default n x n matrix
  - Populate matrix with initial values
  - Importance matrix:
    - Sum up matrix column
    - Divide entry in column by sum
  - etc