PageRank, BFS, AWS Setup

TAs: Akanksha Tripathy, Yash Nakadi, Jeffrey Li

Breadth-First Search

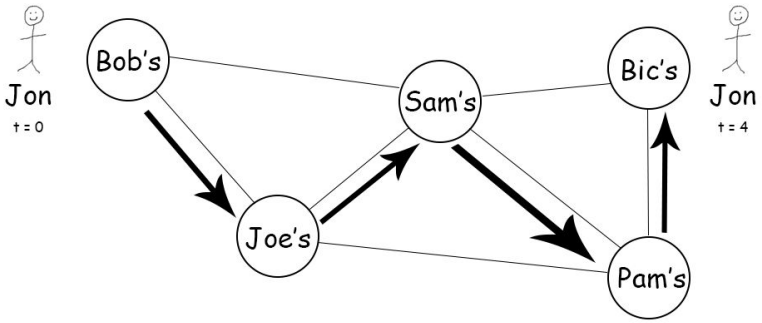
Graph Traversal: The core concept in any Algorithms class!

A software engineer's bread and butter.

Why is it important for a data scientist?

Datasets will often contain patterns and connections, for ex. A railway service dataset, with routes for multiple trains.

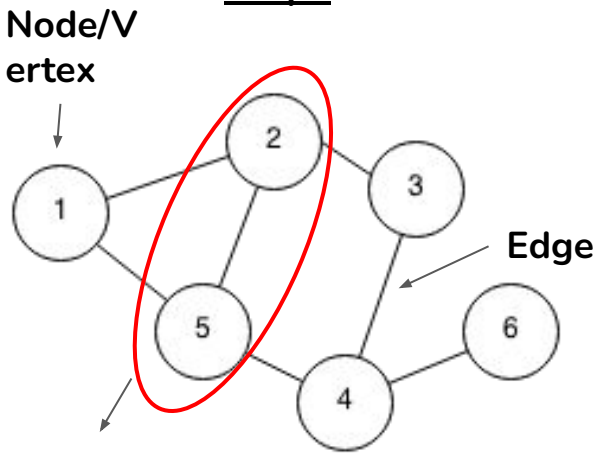
Graph traversal techniques become handy tools for wrangling such data.



Basic Graph Terminology

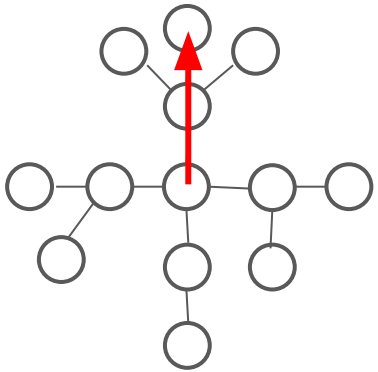
Before we search deeper, let's get a grip of the basic terminology that we'll use..

Graph



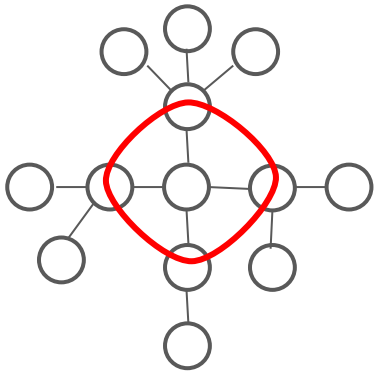
2 & 5 are **neighbors** of 1

Depth



Unidirectional edge

Breadth

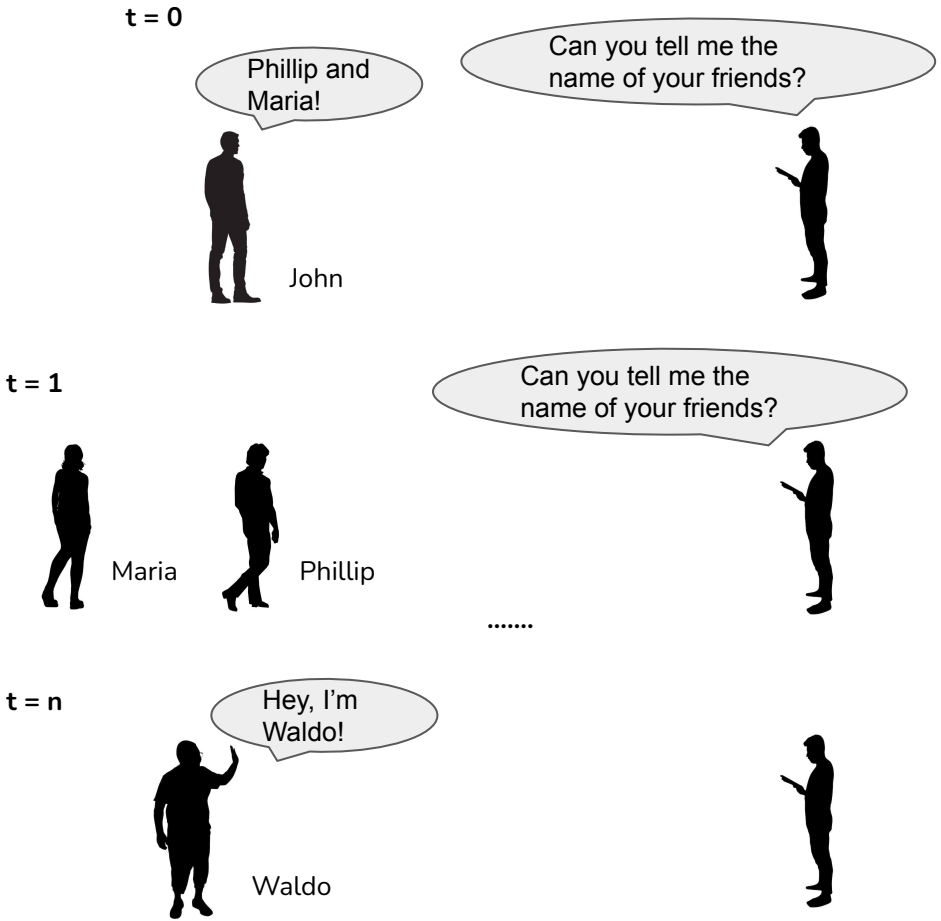


Bidirectional edge

Where’s Waldo?

Let’s have a look at a real life use case of BFS.

Person	Friends
John	Phillip, Maria
Phillip	John, Carla, Ethan
Maria	John
Carla	Phillip
Ethan	Phillip, Waldo
Waldo	Ethan



Implementation

Let's implement this idea on Python! Refer to this pseudocode and try to fill out the BFS code in the Recitation notebook:

```
create a queue Q

mark p as visited and put p into Q (p is the start node)

while Q is non-empty

    remove the head u of Q

      if u is the target, break out of the loop

      mark and enqueue all (unvisited) neighbours of u
```

Demo

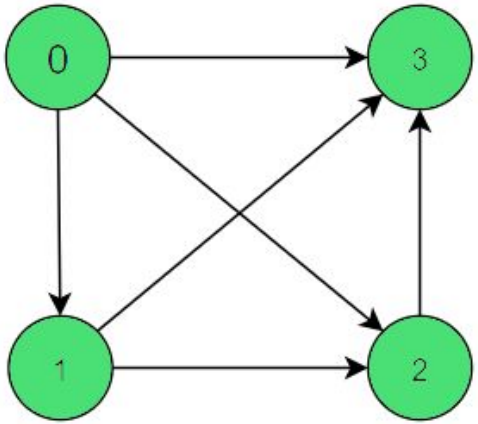
Something to Think About..

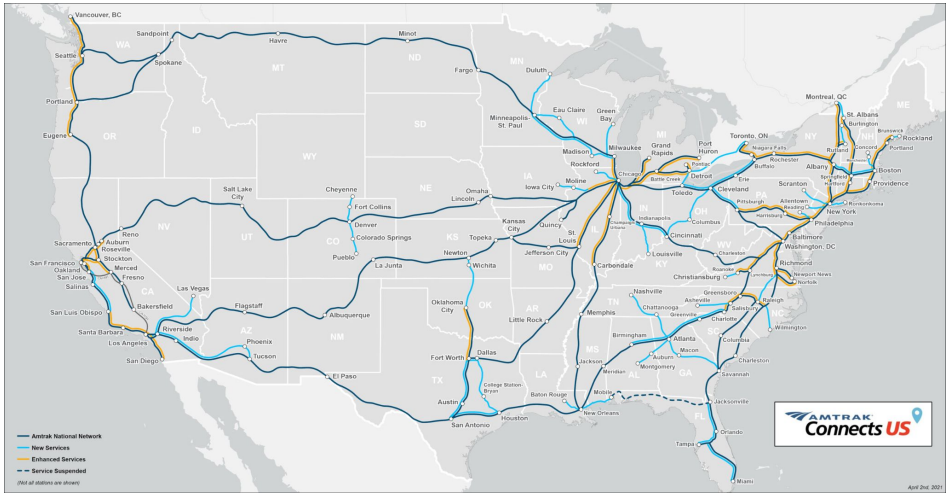
**** IMPORTANT FOR HOMEWORK 3 ****

Now that we know how to implement BFS on Python, can you figure out how to do this using SQL instead?

PageRank Prelude (Guiding Question)

Based on the graphs, which **nodes** do you think are the most important?



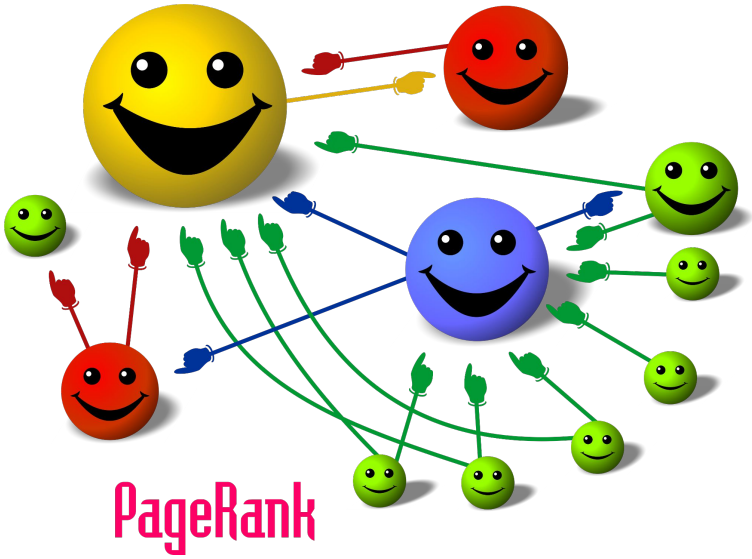


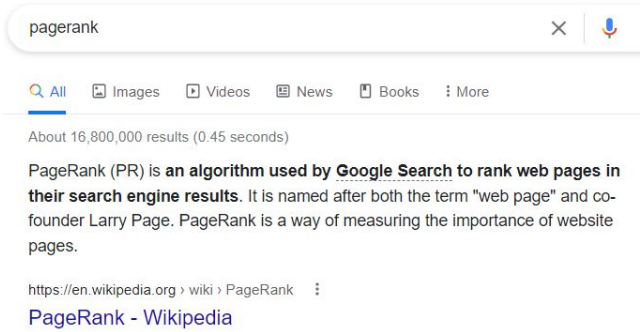
PageRank Intro

Algorithm used by Google to **rank websites** in search engine results.

Intuition: Pages with higher quality and higher number of incoming links are more important.

Assumption: More important websites are likely to receive more links from other websites.





PageRank Iterative Approach

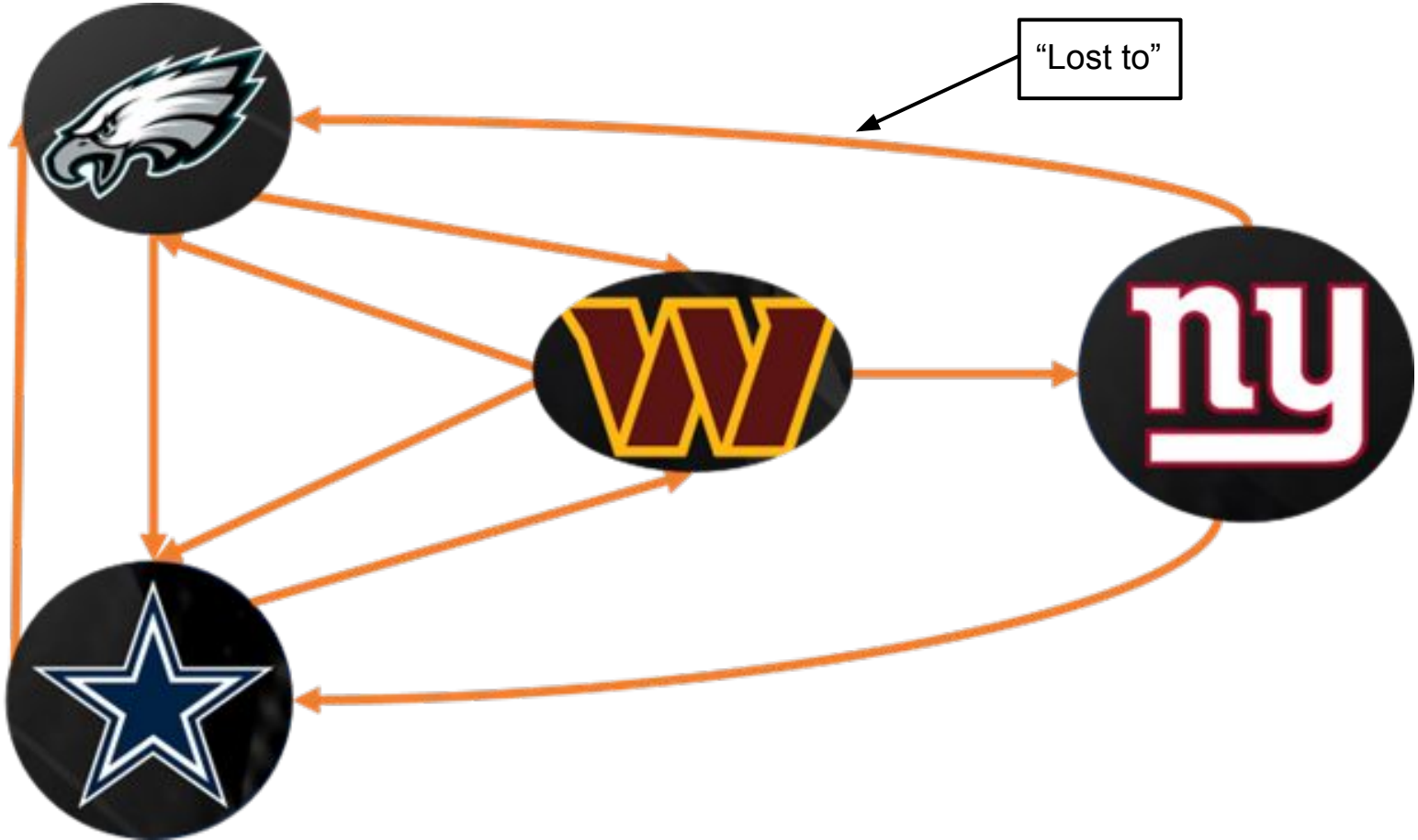
PageRank can take a value between 0 and 1

- The sum of all pages' ranks is 1
- **Fluid:** rank is distributed among the pages

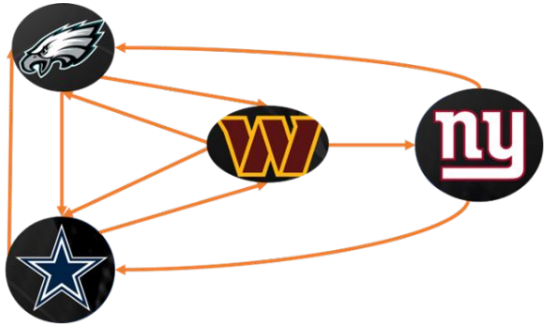
Iterative Approach

- Initialize pages with a predetermined rank value, usually **1 / n**, in which n = number of pages
- Propagate weights across outgoing edges
- Update the weights of pages based on the sum of incoming edges
- Iterate until convergence!

Example: 2022 NFC East Division Results



PageRank Iteration:



Stage 0 (Initialization)

Team	PageRank Weight
Eagles	
Cowboys	
Commanders	
Giants	

Stage 1

Team	PageRank Weight
Eagles	
Cowboys	
Commanders	
Giants	

Stage 2

Team	PageRank Weight
Eagles	
Cowboys	
Comman	
Giants	

Stage 3

Team	Weight
Eagles	
Cowboys	
Comman	
Giants	

Stage 4

Team	Weight
Eagles	
Cowboys	
Comman	
Giants	

PageRank Using Matrices

$$\begin{bmatrix} PageRank(p_1') \\ PageRank(p_2') \\ \dots \\ PageRank(p_m') \end{bmatrix} = M \begin{bmatrix} PageRank(p_1) \\ PageRank(p_2) \\ \dots \\ PageRank(p_m) \end{bmatrix}$$

Create an **m x m** weight transfer matrix M to capture links:

M(i, j) =

•**1 / nj** if page i is pointed to by page j and page j has nj outgoing links
•0 otherwise

Initialize all PageRanks to 1 (or 1/m), multiply by M repeatedly until all values converge

P’	=	0	0.5	0.33	0.5	* P
D’		0.5	0	0.33	0.5	D
W’		0.5	0.5	0	0	W
N’		0	0	0.33	0	N

P W