PageRank

Josh Mandzak & Jared Staman

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Test Questions

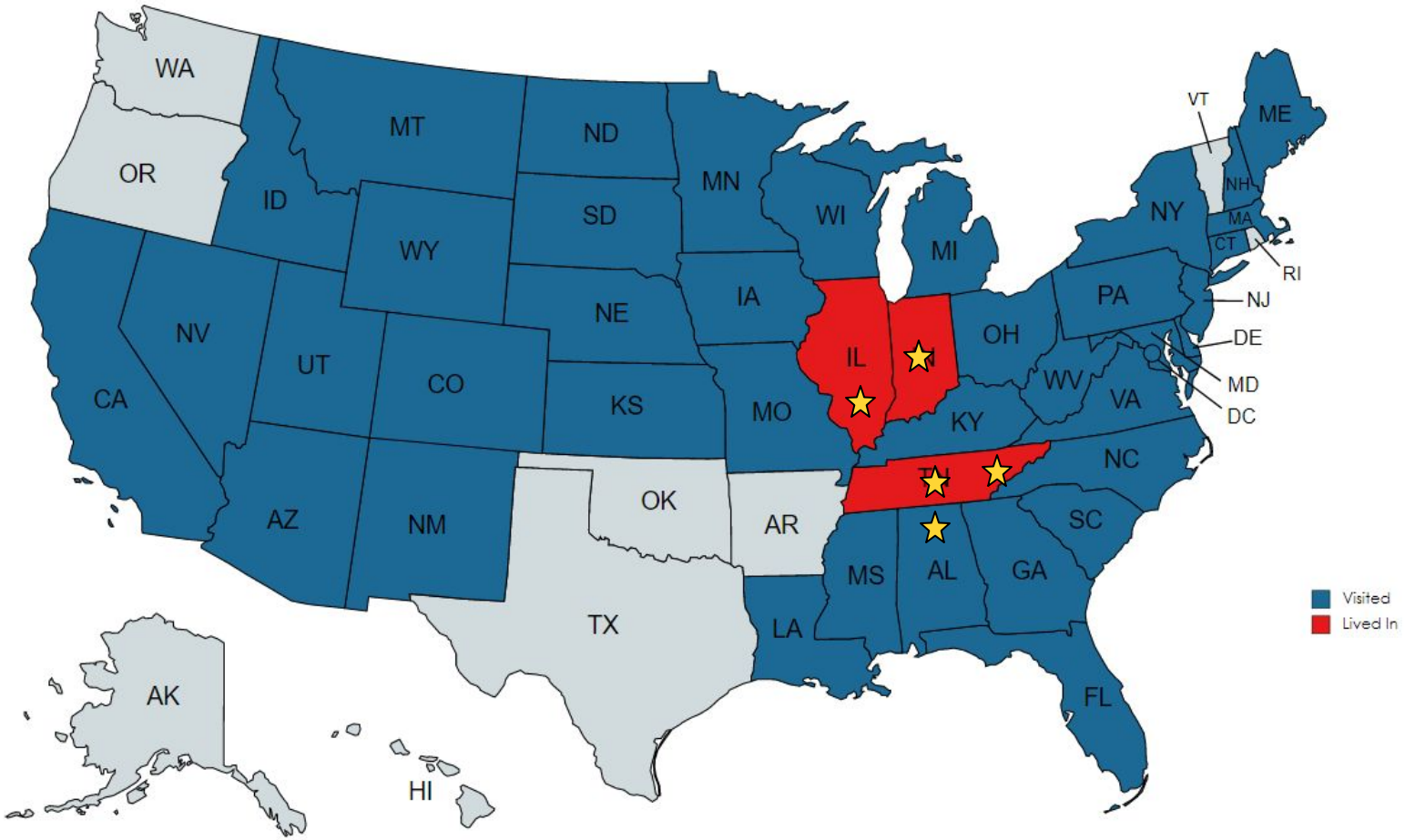
1. What was the name of Larry Page and Sergey Brin's original search engine that used backlinks?

2. What are the two stopping criteria for the iterative update process?

3. What were the nodes with the highest PageRank value for both of the guessing game examples?

Jared Staman

- Master’s student, Computer Science, graduating this Spring
- B.S. in CS at UTK May 2022
- CS 102 TA
- Software Engineer job at Lockheed Martin in Huntsville, AL

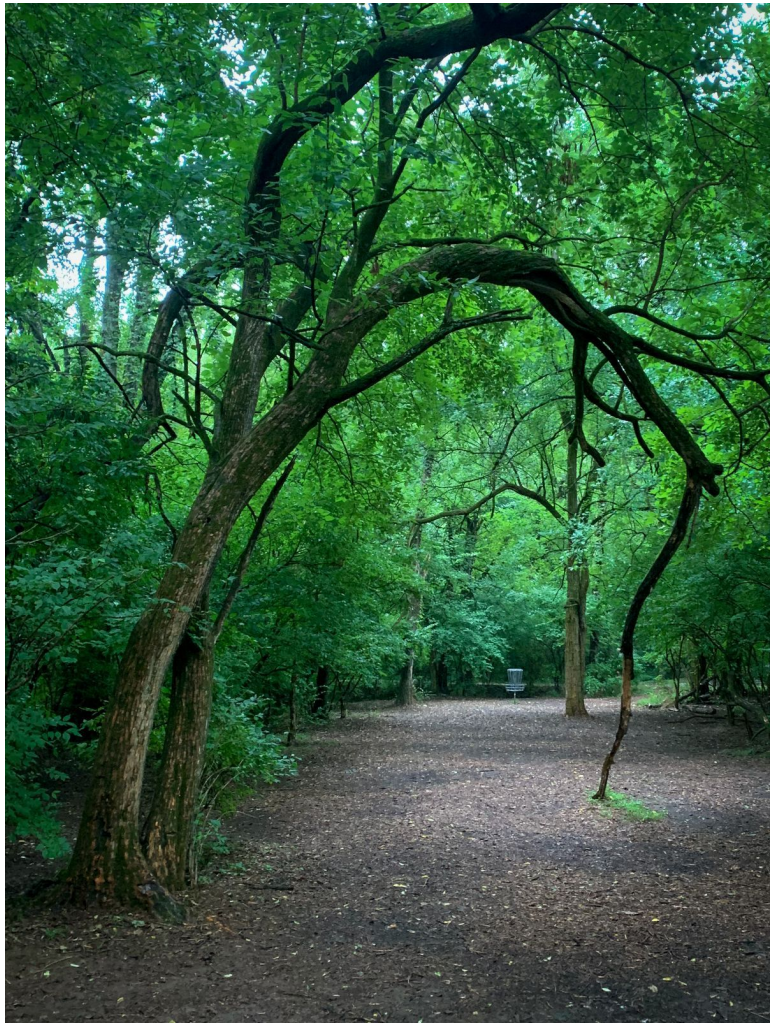


Lou Malnati's® PIZZERIA

THE UNIVERSITY OF TENNESSEE KNOXVILLE

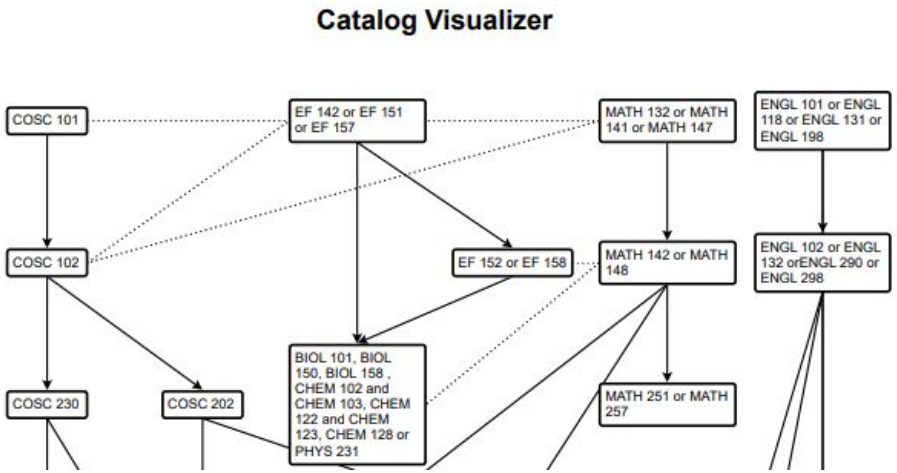
Hobbies

- Card Games (Pinochle, Poker)
- Disc Golf, Basketball, Golf
- Binging TV shows



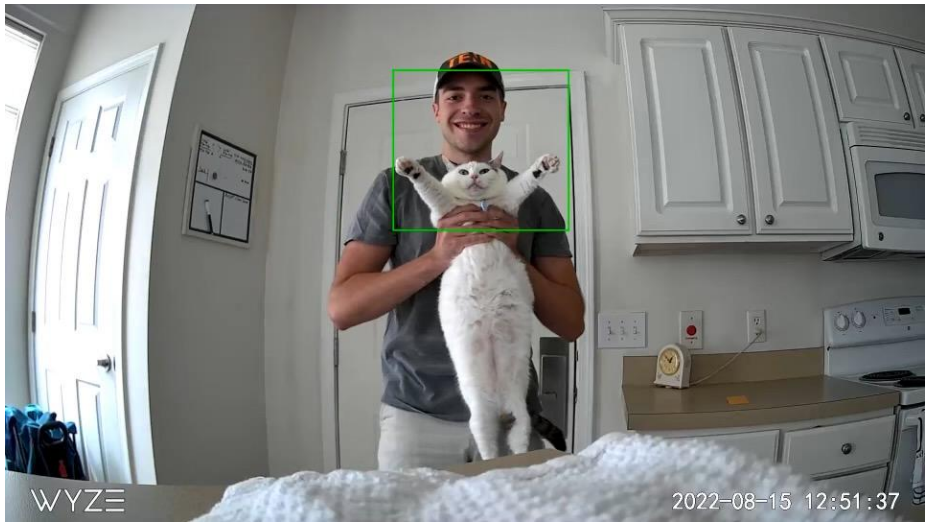
Josh Mandzak

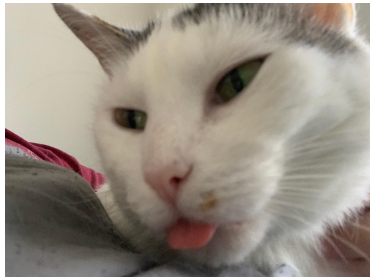
- Received BS in CS at UTK Spring 2022
- Expected MS in CS at UTK Spring 2023
- Full Time GTA for Dr. Berry
- Moving to Cary, NC for Garmin after graduation



Personal Life

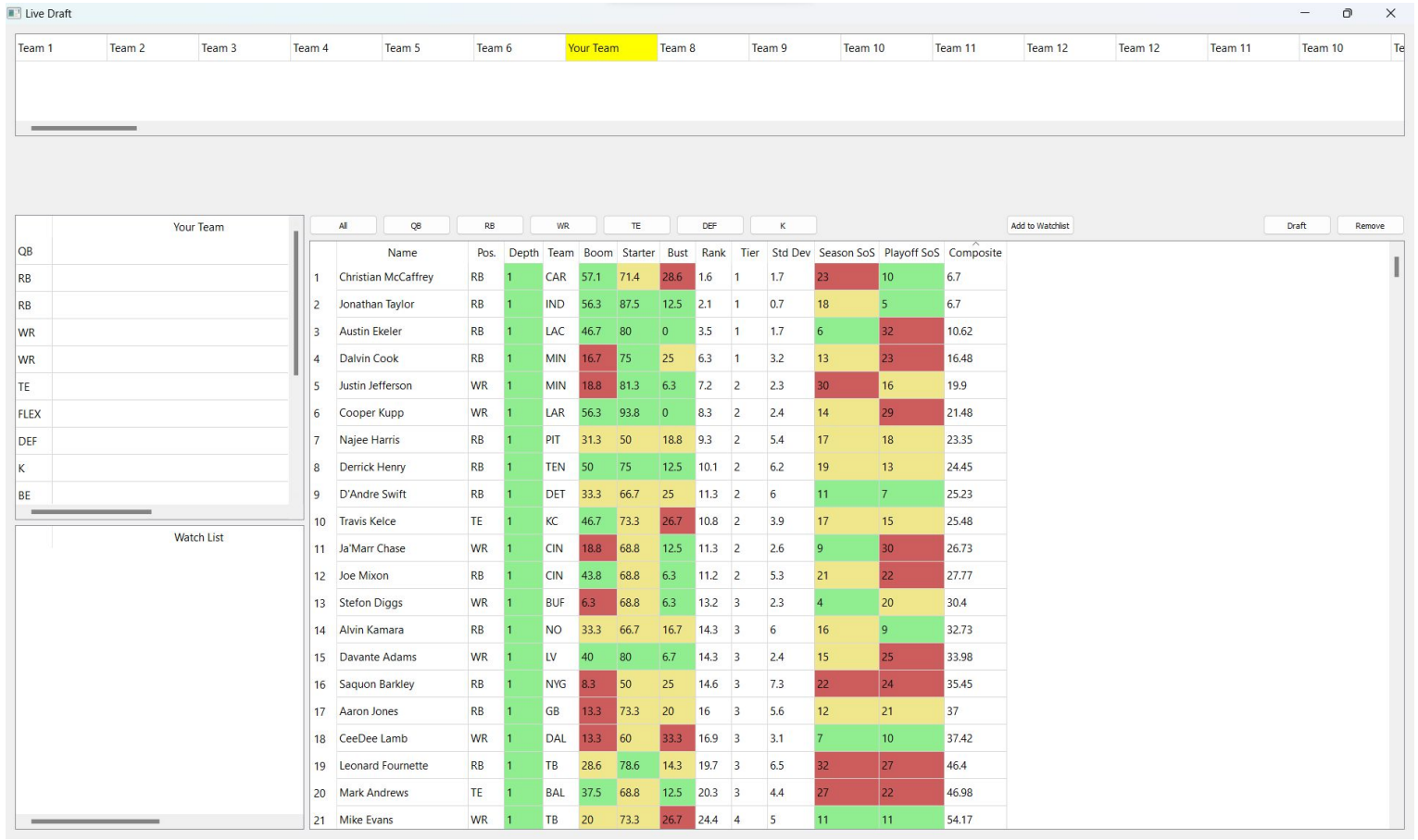
- From Morristown, TN
- Engaged
- Owner of world's dumbest cat (unofficial)
- Football Fan
  - Fantasy Football Champion





https://www.tripadvisor.com/LocationPhotoDirectLink-g55217-d4279107-i153164118-Taco_John_s-Morristown_Tennessee.html
http://townmapsusa.com/d/map-of-morristown-tennessee-tn/morristown_tn

Coding Meets Personal Life



	Name	Pos.	Depth	Team	Boom	Starter	Bust	Rank	Tier	Std Dev	Season SoS	Playoff SoS	Composite
1	Christian McCaffrey	RB	1	CAR	57.1	71.4	28.6	1.6	1	1.7	23	10	6.7
2	Jonathan Taylor	RB	1	IND	56.3	87.5	12.5	2.1	1	0.7	18	5	6.7
3	Austin Ekeler	RB	1	LAC	46.7	80	0	3.5	1	1.7	6	32	10.62
4	Dalvin Cook	RB	1	MIN	16.7	75	25	6.3	1	3.2	13	23	16.48
5	Justin Jefferson	WR	1	MIN	18.8	81.3	6.3	7.2	2	2.3	30	16	19.9
6	Cooper Kupp	WR	1	LAR	56.3	93.8	0	8.3	2	2.4	14	29	21.48
7	Najee Harris	RB	1	PIT	31.3	50	18.8	9.3	2	5.4	17	18	23.35
8	Derrick Henry	RB	1	TEN	50	75	12.5	10.1	2	6.2	19	13	24.45
9	D'Andre Swift	RB	1	DET	33.3	66.7	25	11.3	2	6	11	7	25.23
10	Travis Kelce	TE	1	KC	46.7	73.3	26.7	10.8	2	3.9	17	15	25.48
11	Ja'Marr Chase	WR	1	CIN	18.8	68.8	12.5	11.3	2	2.6	9	30	26.73
12	Joe Mixon	RB	1	CIN	43.8	68.8	6.3	11.2	2	5.3	21	22	27.77
13	Stefon Diggs	WR	1	BUF	6.3	68.8	6.3	13.2	3	2.3	4	20	30.4
14	Alvin Kamara	RB	1	NO	33.3	66.7	16.7	14.3	3	6	16	9	32.73
15	Davante Adams	WR	1	LV	40	80	6.7	14.3	3	2.4	15	25	33.98
16	Saquon Barkley	RB	1	NYG	8.3	50	25	14.6	3	7.3	22	24	35.45
17	Aaron Jones	RB	1	GB	13.3	73.3	20	16	3	5.6	12	21	37
18	CeeDee Lamb	WR	1	DAL	13.3	60	33.3	16.9	3	3.1	7	10	37.42
19	Leonard Fournette	RB	1	TB	28.6	78.6	14.3	19.7	3	6.5	32	27	46.4
20	Mark Andrews	TE	1	BAL	37.5	68.8	12.5	20.3	3	4.4	27	22	46.98
21	Mike Evans	WR	1	TB	20	73.3	26.7	24.4	4	5	11	11	54.17

FFgenius

This repository will contain machine learning models meant to help rank players to help with your fantasy football draft

Important Files

`models.py` runs a basic comparison between random forest, multiple linear regression, and neural nets for each position tested. Simply run the program and it will generate matrices for each position showing important evaluation scores for each model.

`nn_tuning.py` runs a randomized or grid search CV on a position of players (based on what is commented and uncommented out) in order to find the ideal hyperparameters for each model by position.

`random_forest_tuning.py` essentially does the same thing as `nn_tuning.py`, but with random forest regressors.

`relationships.py` shows a sorted heatmap for each position showing the relationship between all of the independent variables and the dependent variable (average points per game)

`parse.py` is run to create the filtered CSVs used for the model tuning. This program takes the general files and creates optimized CSVs that can be used more easily to streamline the ML model creation process.

The `stats` directory contains all of the files used by the machine learning models.

README.md

NFLsurvivor (WIP)

A repo to help you optimize your survivor pool strategy

Outline

- Brief Overview, Important Terms
- History of PageRank
- Algorithm
  - Algorithm Guessing Game
- Applications
- Implementation
- Open Issues
- Discussion

Overview

Terminology:

- Random Surfer
  - Someone who surfs web by clicking links
- Markov Chains
  - “a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event”
- Backlink
  - URL link from one page leading to another
- Search Engine
  - tool for finding websites on the web

History

Citation Analysis

- “Examination of the frequency, patterns, and graphs of citations in documents”
- Eugene Garfield established Institution for Scientific Information (ISI) in 1960
- *Citation Analysis as a Tool in Journal Evaluation* published 1972

Michael A. Langston

University of Tennessee
Verified email at tennessee.edu - Homepage
Big Data Analytics Graph Theoretical Algorithms Life Science Applications

Cited by		VIEW ALL
	All	Since 2018
Citations	7209	1875
h-index	46	22
i10-index	121	65

Josh Mandzak

About 20 results (**0.09** sec)

Add articles you wrote.

Selected: 0

Track citations to your articles

CREATE YOUR PROFILE Learn more

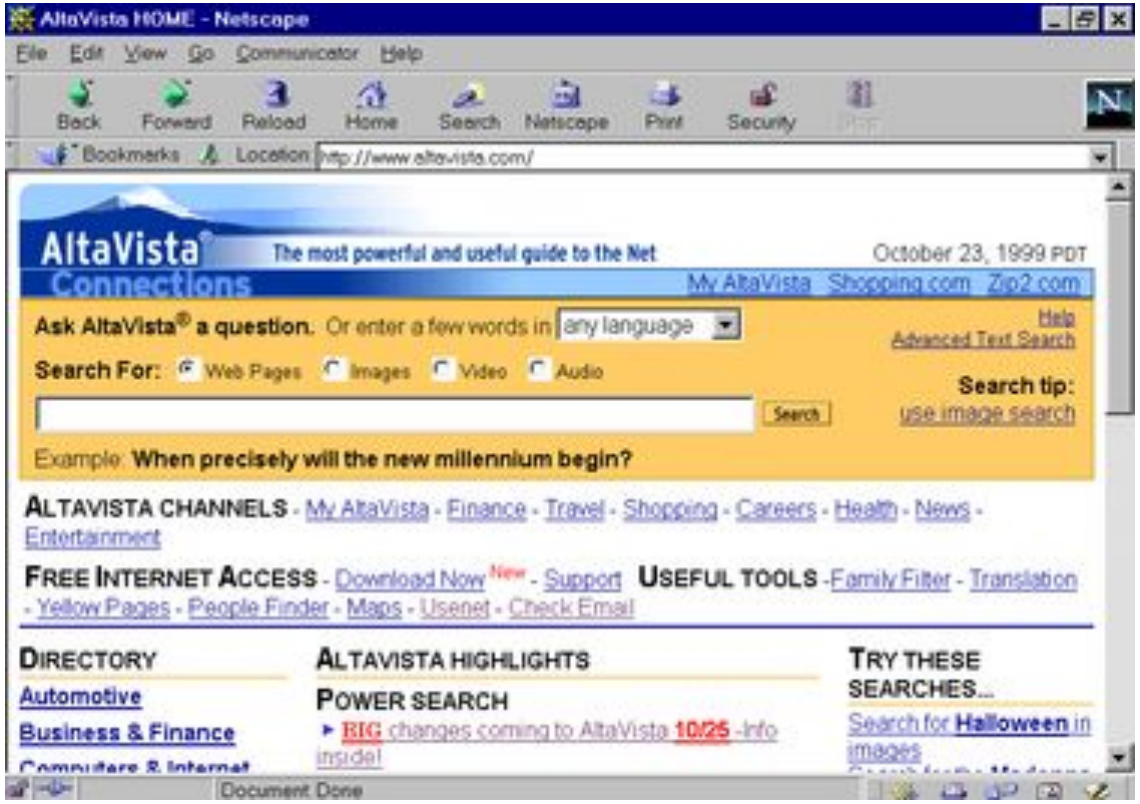
https://en.wikipedia.org/wiki/Citation_analysis
https://scholar.google.com/citations?user=PXCKvVgAAAAJ&hl=en&oi=ao
Citation Analysis as a Tool in Journal Evaluation (upenn.edu)

History

Search Engines

- First search engine “Archie” created 1990
- Stanford students create “Architext” 1993
- Yahoo! created 1994
- AltaVista created 1995

- Large problems included scale and search accuracy



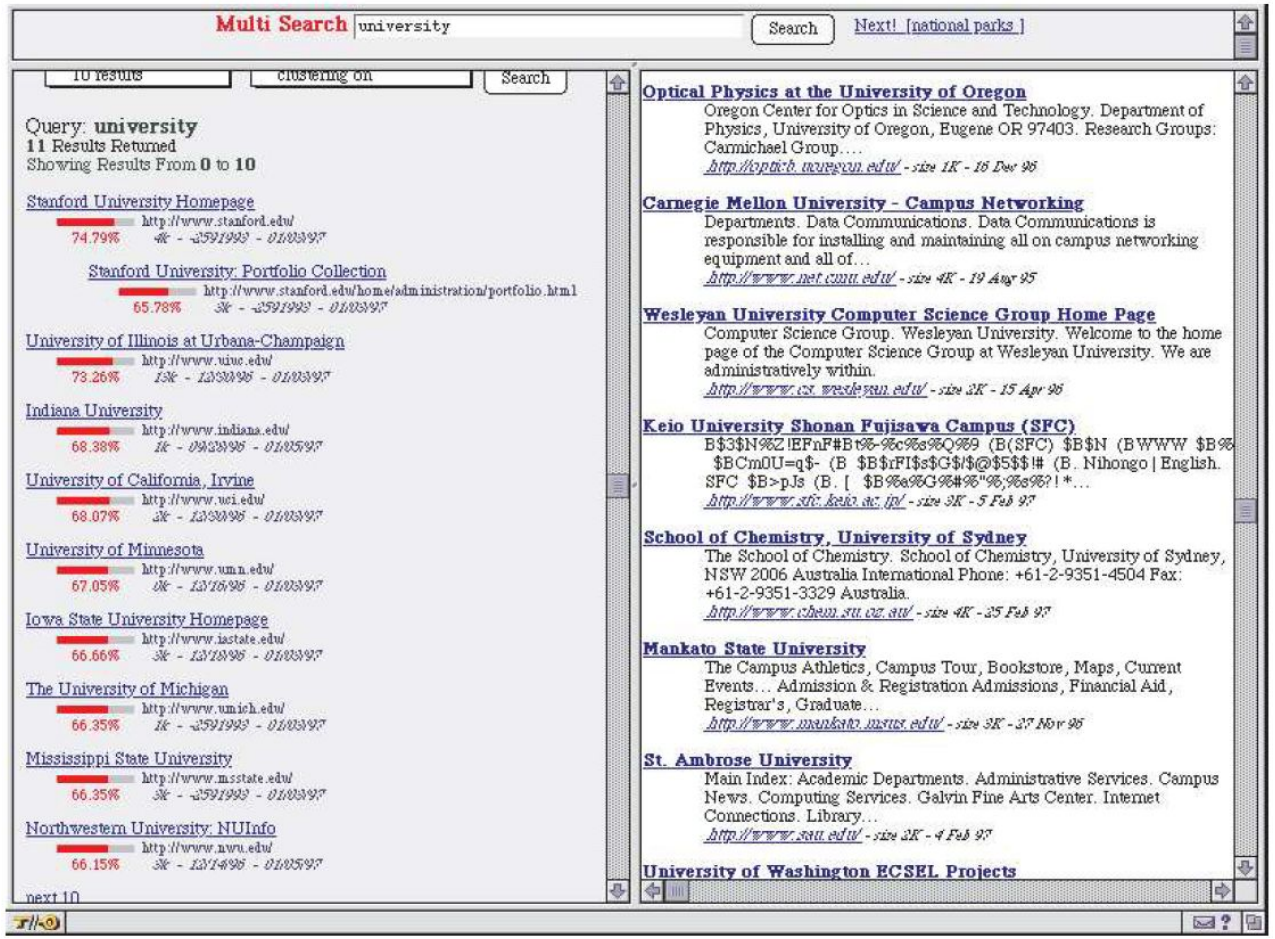
https://en.wikipedia.org/wiki/AltaVista

History

Google As We Know It

- 1996, Stanford grad students Larry Page and Sergey Brin begin creating “Backrub” search engine

- In 1998 published two main papers:
  - *The Anatomy of a Large-Scale Hypertext Web Search Engine*
  - *The PageRank Citation Ranking: Bringing Order to the Web*



http://infolab.stanford.edu/pub/papers/google.pdf https://www.cis.upenn.edu/~mkearns/teaching/NetworkedLife/pagerank.pdf

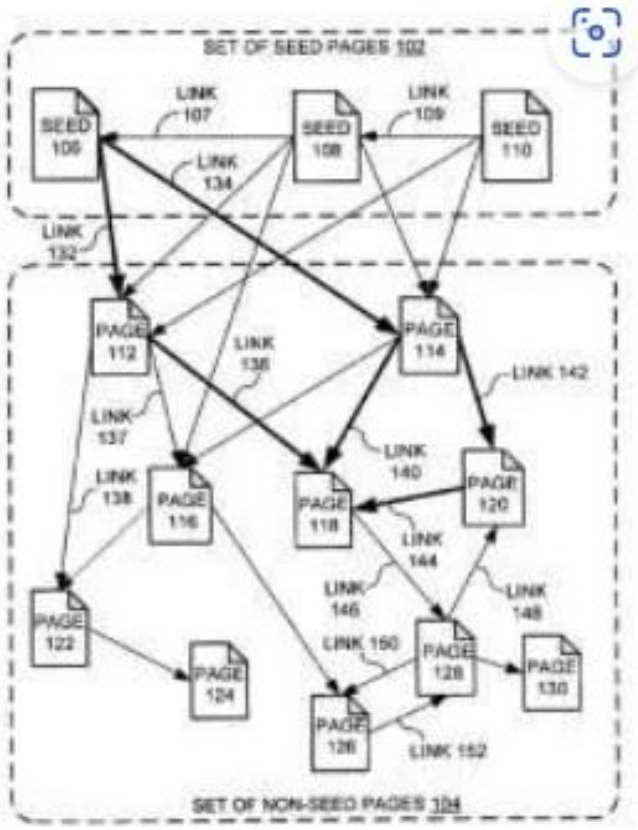
History

- “We have implemented two search engines… the second search engine is a full text search engine called Google”
- “For more queries, we encourage the reader to test Google themselves”
- “At worst, you can have manipulation in the form of buying links on important sites. But, this seems well under control since it costs money.”

History

PageRank

- 1998 - PageRank patent filed
- Circa early 2000s - Google realizes they shouldn't use open source algorithms
- 2006 (Granted 2015) - New patent, seen as PageRank alternative or update
- 2015 (Granted 2018) - Update to 2006 patent



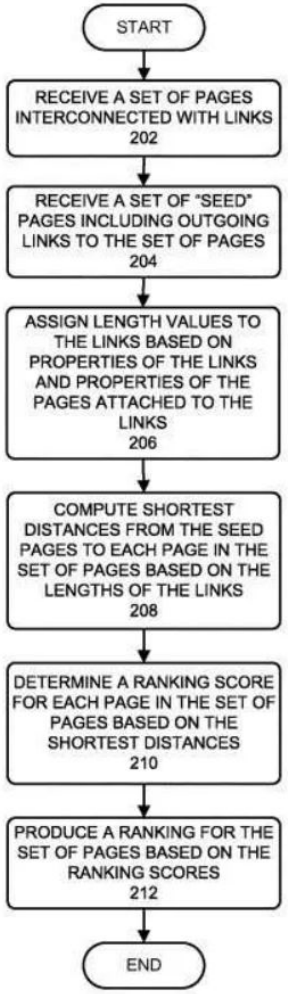


FIG. 2

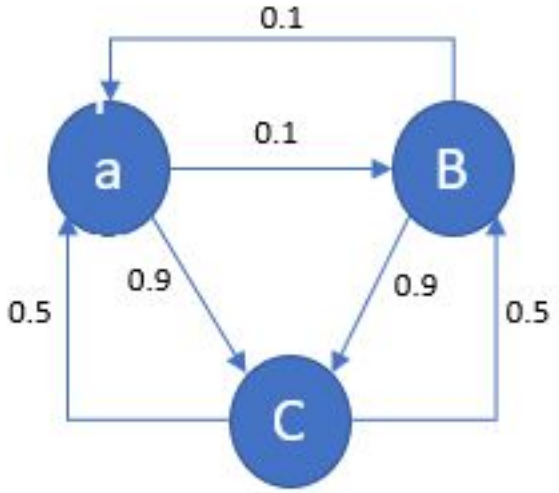
https://www.seobythesea.com/2015/11/recalculating-pagerank/ https://www.seobythesea.com/2018/04/pagerank-updated/

Algorithm

Intuition

- A page's rank can be represented by the following problem:
  - Given a "random surfer", what's the likelihood of them being on a given page?

- Roots in Markov Chains



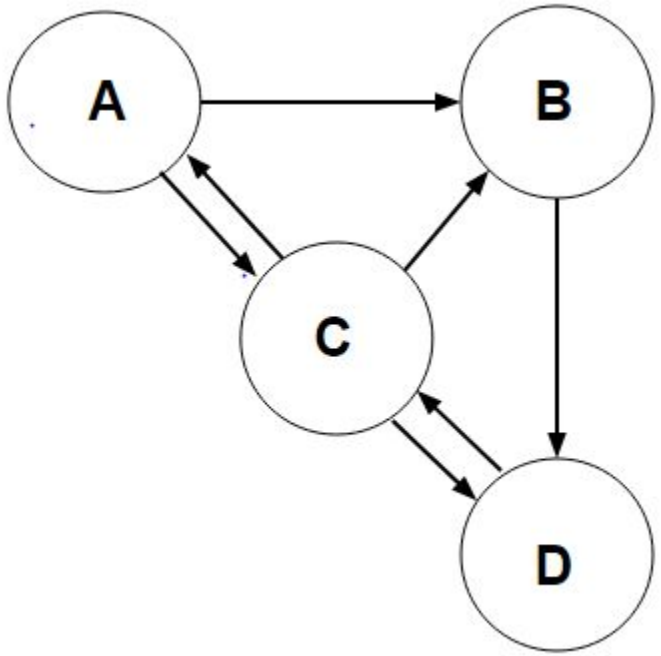
$$P = \begin{bmatrix} 0 & 0.1 & 0.9 \\ 0.1 & 0.0 & 0.9 \\ 0.5 & 0.5 & 0.0 \end{bmatrix}$$

https://towardsdatascience.com/markov-chain-analysis-and-simulation-using-python-4507cee0b06e

Algorithm

Key Formula: $PR(u) = ((1 - d) / N) + d * \Sigma (PR(v) / L(v))$

- PR(u) = Page Rank of page u
- PR(v) = Page Rank of page v
- d = damping factor (usually 0.85)
- N = number of pages
- L(v) = number of outbound links on page v



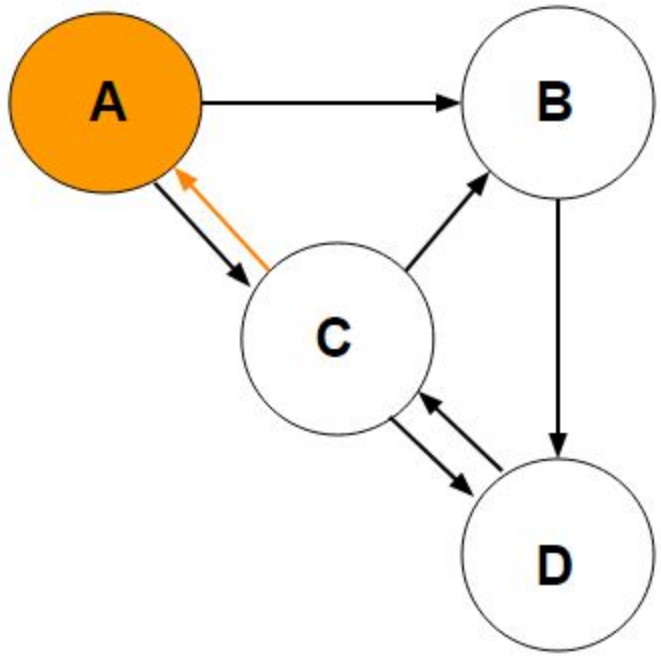
	Iteration 0	Iteration 1	Iteration 2
A	.25		
B	.25		
C	.25		
D	.25		

Iteration 0 = 1 / N = 1 / 4

Algorithm

Key Formula: $PR(u) = ((1 - d) / N) + d * \Sigma (PR(v) / L(v))$

- PR(u) = Page Rank of page u
- PR(v) = Page Rank of page v
- d = damping factor (usually 0.85)
- N = number of pages
- L(v) = number of outbound links on page v



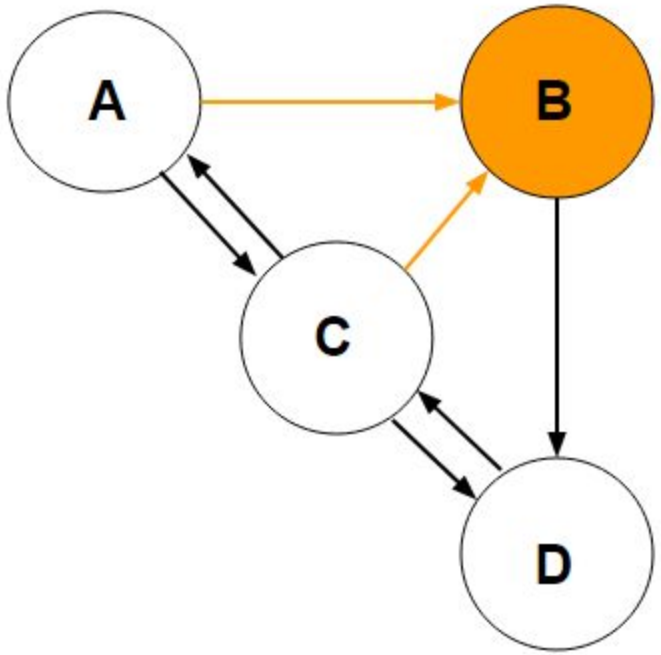
	Iteration 0	Iteration 1	Iteration 2
A	.25	.1083	
B	.25		
C	.25		
D	.25		

PR(A) = ((1 - .85) / 4) + (.85 * (.25 / 3))

Algorithm

Key Formula: $PR(u) = ((1 - d) / N) + d * \Sigma (PR(v) / L(v))$

- PR(u) = Page Rank of page u
- PR(v) = Page Rank of page v
- d = damping factor (usually 0.85)
- N = number of pages
- L(v) = number of outbound links on page v



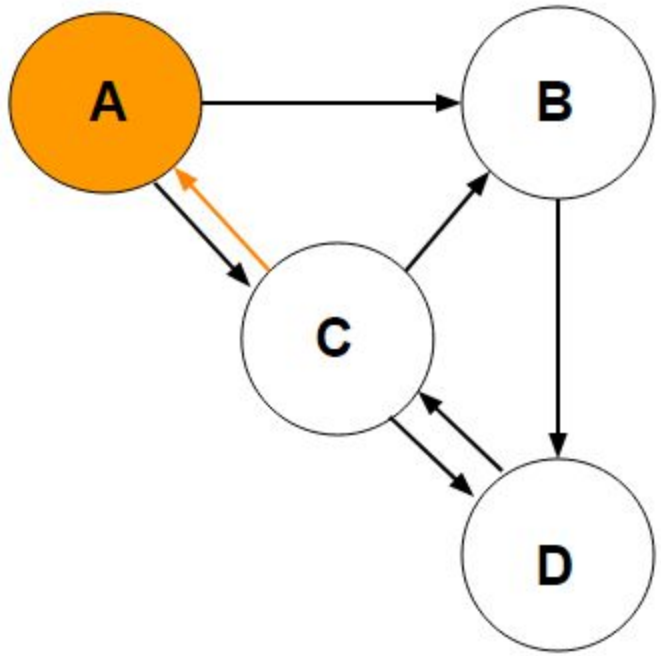
	Iteration 0	Iteration 1	Iteration 2
A	.25	.1083	
B	.25	.2146	
C	.25		
D	.25		

PR(B) = ((1 - .85) / 4) + (.85 * [(.25 / 2) + (.25 / 3)])

Algorithm

Key Formula: $PR(u) = ((1 - d) / N) + d * \Sigma (PR(v) / L(v))$

- PR(u) = Page Rank of page u
- PR(v) = Page Rank of page v
- d = damping factor (usually 0.85)
- N = number of pages
- L(v) = number of outbound links on page v



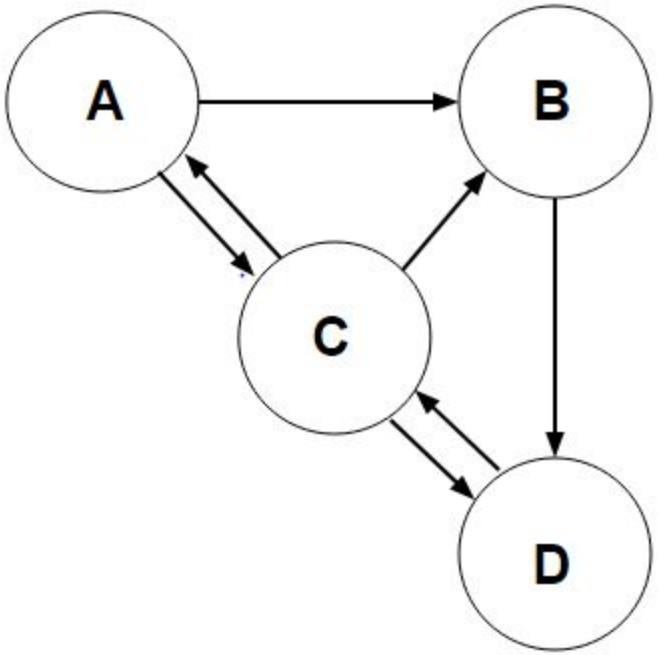
	Iteration 0	Iteration 1	Iteration 2
A	.25	.1083	.1385
B	.25	.2146	
C	.25	.3563	
D	.25	.3208	

PR(A) = ((1 - .85) / 4) + (.85 * (.3563 / 3))

Algorithm

Key Formula: $PR(u) = ((1 - d) / N) + d * \Sigma (PR(v) / L(v))$

- PR(u) = Page Rank of page u
- PR(v) = Page Rank of page v
- d = damping factor (usually 0.85)
- N = number of pages
- L(v) = number of outbound links on page v



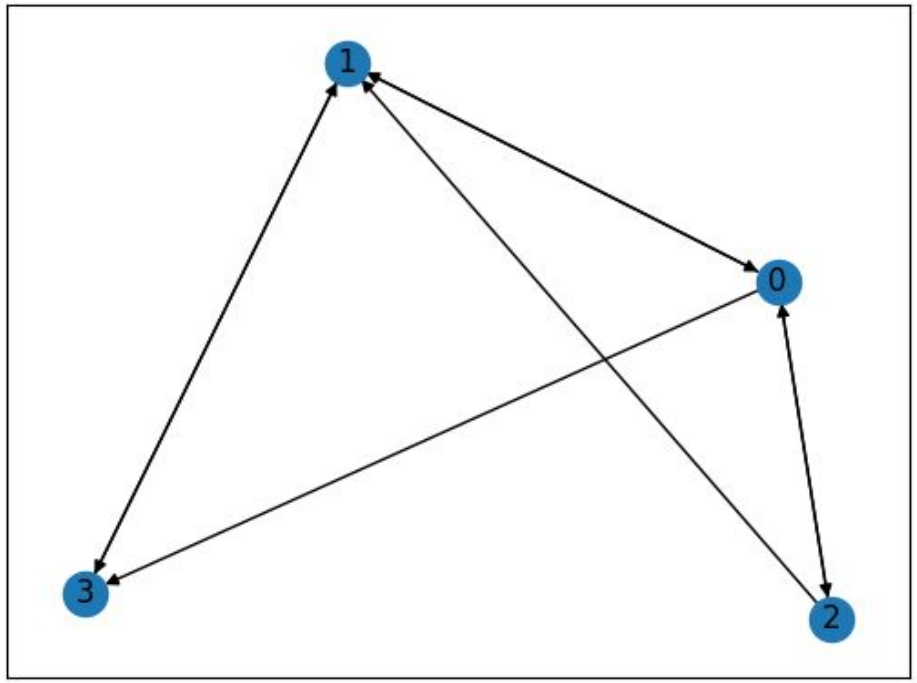
	Iteration 0	Iteration 1	Iteration 2
A	.25	.1083	.1385
B	.25	.2146	.1926
C	.25	.3563	.3643
D	.25	.3208	.3046

When do you stop?

- Reach a predetermined max iterations
- The differences between iterations is less than some very small epsilon value

Algorithm Guessing Game

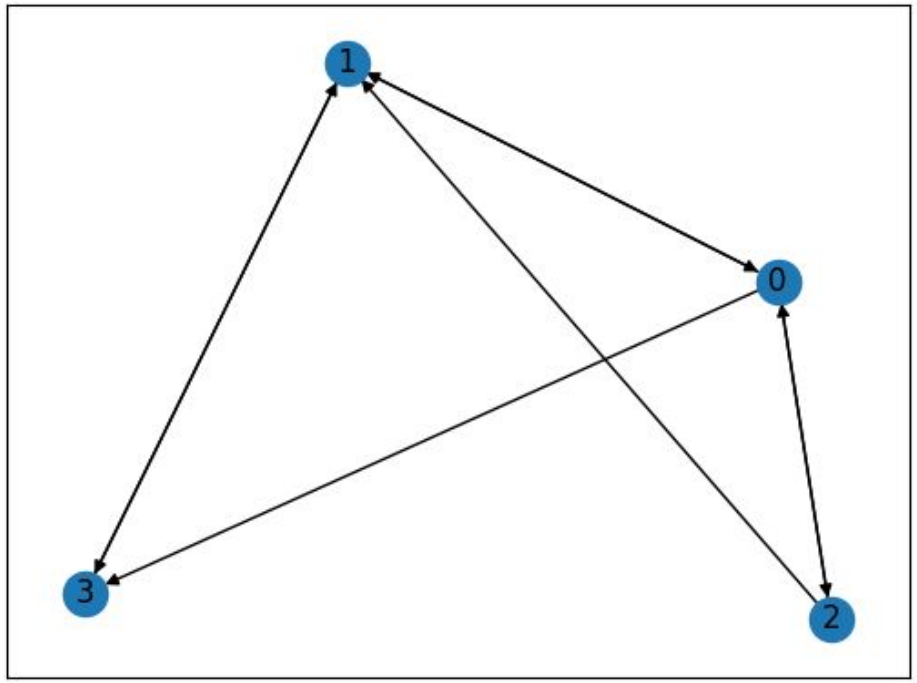
Which node has the highest page rank?



Algorithm Guessing Game

Which node has the highest page rank?

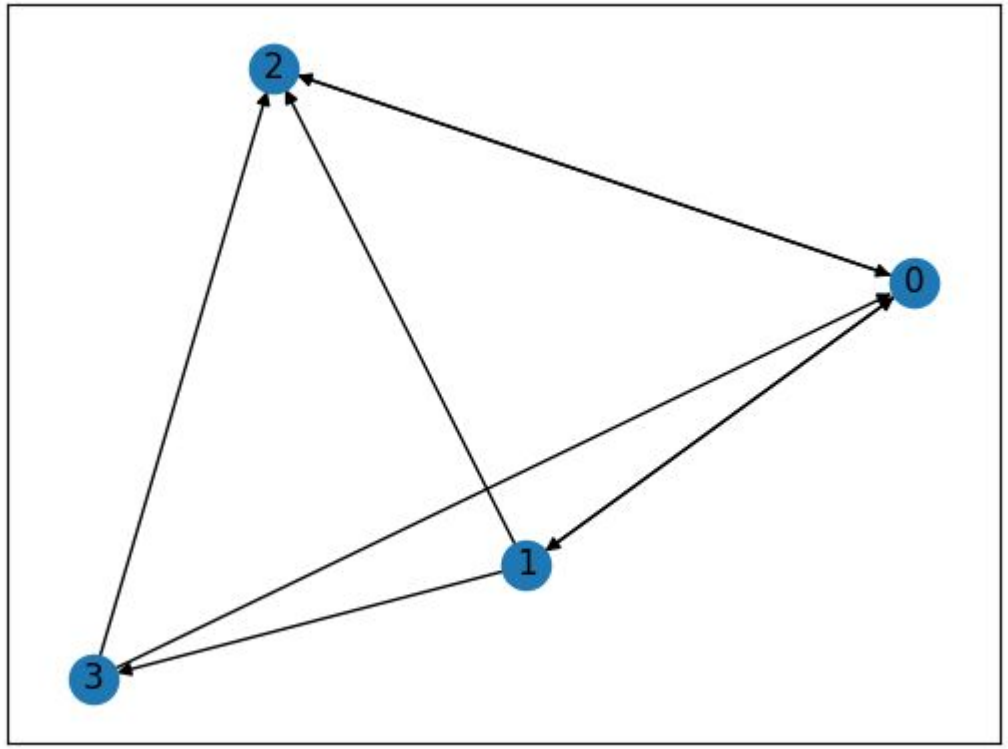
Answer: Node 1



Node	PageRank
0	0.2445
1	0.3803
2	0.1068
3	0.2684

Algorithm Guessing Game

Which node has the highest page rank?

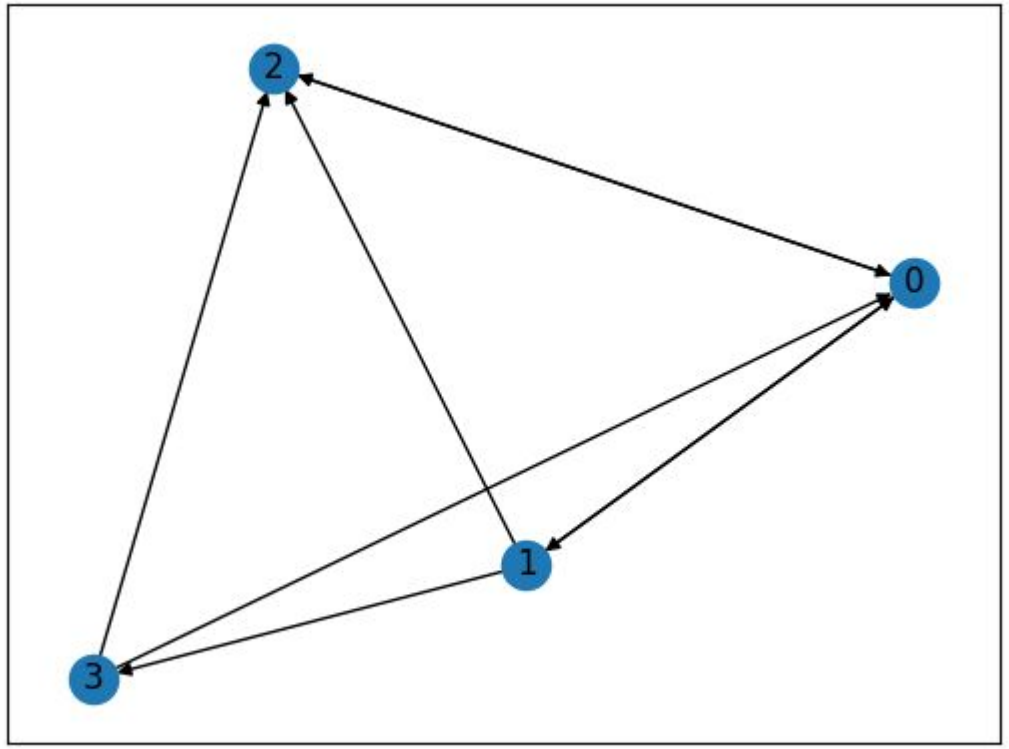


Algorithm Guessing Game

$$PR(u) = ((1 - d) / N) + d * \Sigma (PR(v) / L(v))$$

Which node has the highest page rank?

Answer: Node 0



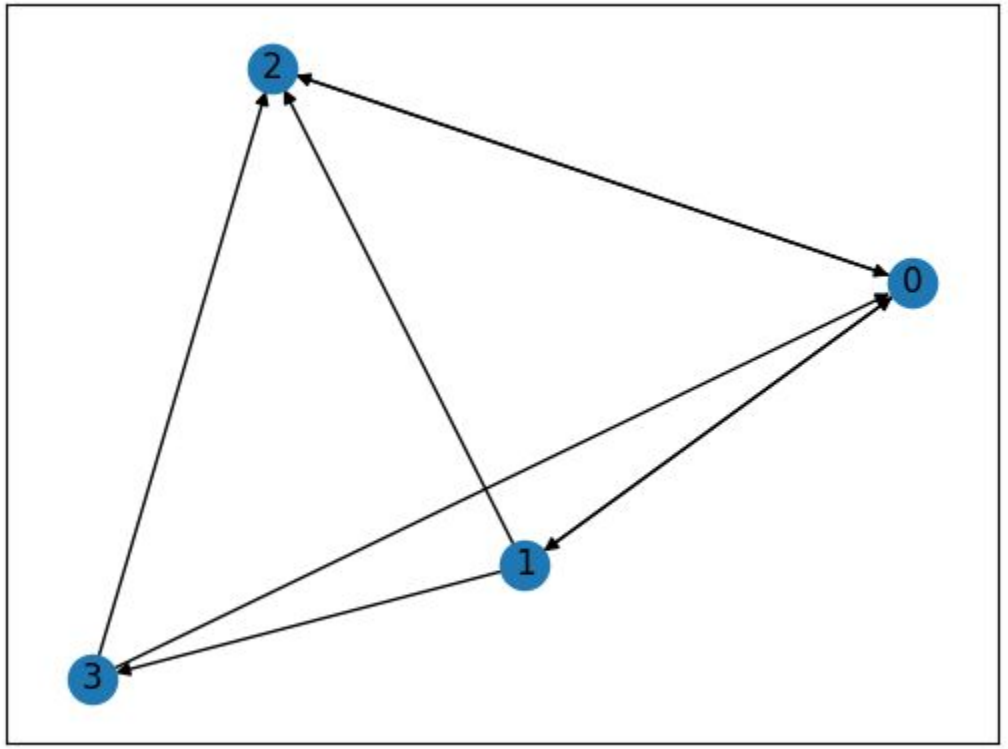
Node	PageRank
0	0.3949
1	0.2053
2	0.3041
3	0.0957

Applications

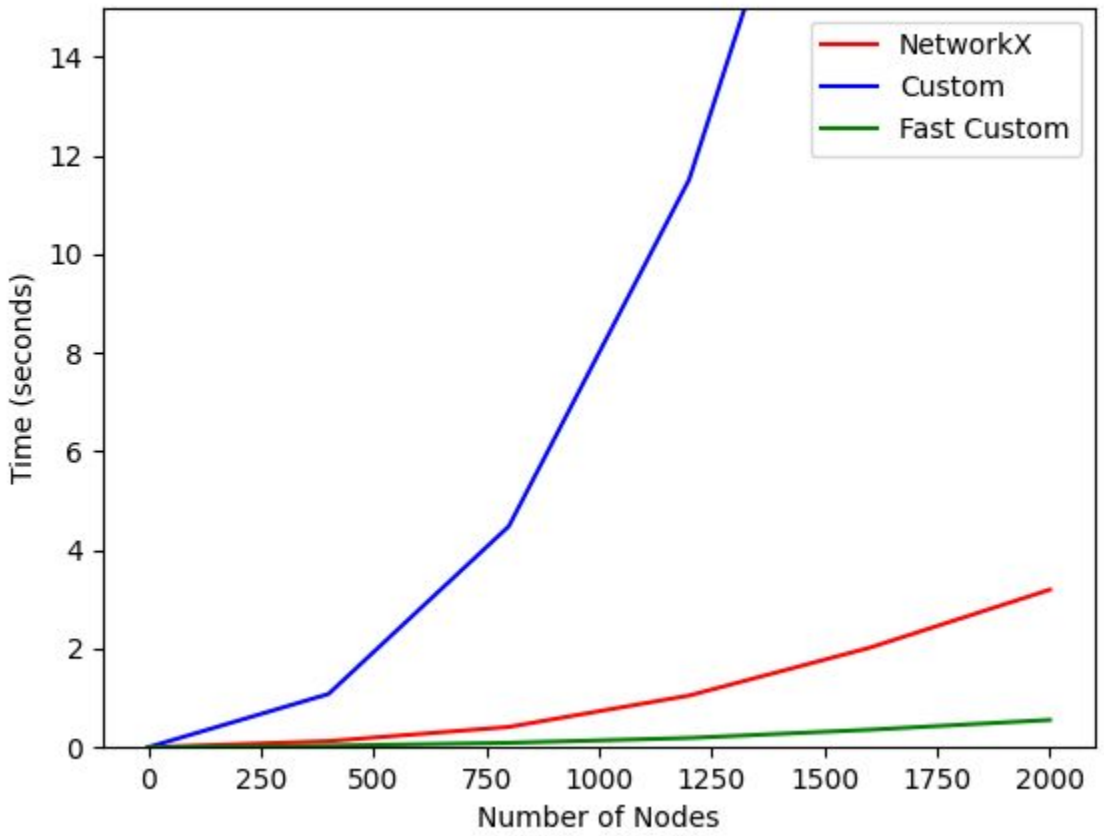
- Search Engines
- Recommender Systems
- Social Networks
- Citation Analysis
- Spam Detection

Implementation

- Custom Python implementation
- 4 implementations
  - NetworkX Library
  - Basic Custom Version
  - Improved Custom Version
  - Random Walk Average



Implementation



Node	PageRank	Random Walk
0	0.3949	0.3908
1	0.2053	0.2083
2	0.3041	0.3034
3	0.0957	0.0975

Implementation

Time Complexity: O(???)

- Somewhat disagreed upon
- O(kn) seems to fit custom implementation
  - k = iterations to convergence
  - n = number of nodes
- If considering k a constant, O(n)
- Ex-Googler claims O(nlogn)

```
summation_terms = []
for i, row in enumerate(graph):
    summation_terms.append(old_pr[i] / np.sum(row))

# now calc each new page rank
for i, row in enumerate(graph):
    new_pr = np.dot(summation_terms, graph[:, i])
    new_pr *= d
    new_pr += ((1 - d) / row.size)
    page_rank[i] = new_pr
```

Open Issues

- Manipulation
  - boost pages artificially
- Lack of personalization
- Scalability
- Fairness and Bias
  - how to get new pages to a high rank?

References

- https://en.wikipedia.org/wiki/Citation_analysis
- https://scholar.google.com/citations?user=PXCKvVgAAAAJ&hl=en&oi=ao
- http://www.garfield.library.upenn.edu/essays/V1p527y1962-73.pdf
- https://en.wikipedia.org/wiki/AltaVista
- https://www.seobythesea.com/2015/11/recalculating-pagerank/
- https://www.seobythesea.com/2018/04/pagerank-updated/
- https://www.cis.upenn.edu/~mkearns/teaching/NetworkedLife/pagerank.pdf
- http://infolab.stanford.edu/pub/papers/google.pdf
- https://towardsdatascience.com/markov-chain-analysis-and-simulation-using-python-4507cee0b06e
- https://www.tripadvisor.com/LocationPhotoDirectLink-g55217-d4279107-i153164118-Taco_John_s-Morristown_Tennessee.html
- http://townmapsusa.com/d/map-of-morristown-tennessee-tn/morristown_tn
- https://github.com/jmandzak/PageRank

Discussion

Test Questions Part 2

1. What was the name of Larry Page and Sergey Brin's original search engine that used backlinks?

2. What are the two stopping criteria for the iterative update process?

3. What were the nodes with the highest PageRank value for both of the guessing game examples?