PageRank

CS16: Introduction to Data Structures & Algorithms
Spring 2019

Outline

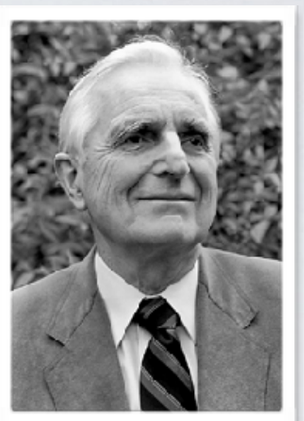
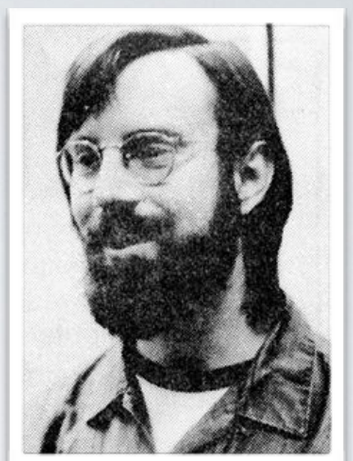
- The WWW & Search Engines
- Basic PageRank
- (Real) PageRank
- PageRank in practice

The World Wide Web

- Created by Tim-Berners Lee in 1989
- Collection of "pages"
- Pages are
  - identified by Uniform Resource Locator (URL)
  - composed of **text** & **hyperlinks** (pointers to other pages)

Hypertext

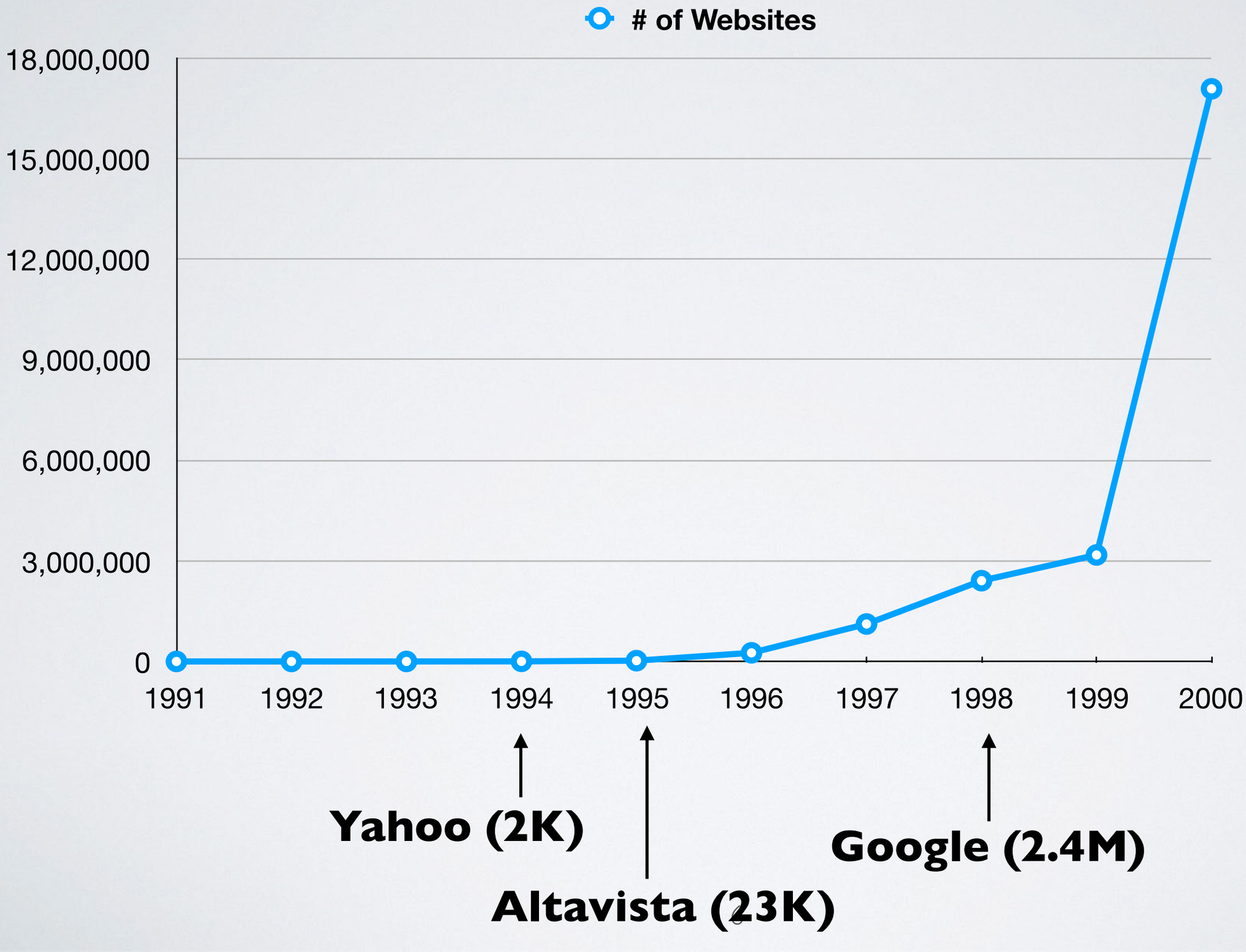


- Hypertext and hyperlinks predate the WWW
- Hypertext Editing System (HES) in 1967
  - Ted Nelson, Andy van Dam + Brown students
- File Retrieval and Editing System (FRESS) in 1968
  - Andy van Dam + Brown students (including Bob Wallace)
  - used in Brown's "Introduction to Poetry" in 1975 & 1976
- oN-Line System (NLS) in 1968
  - Douglas Engelbart

Growth of the Web

Growth of the Web



Search Engines

- The Web is great but how do find what we need?
- Search engine
  - system that indexes collection of web pages
  - returns relevant pages when queried with keyword(s)

- Q: how do we build a search engine?

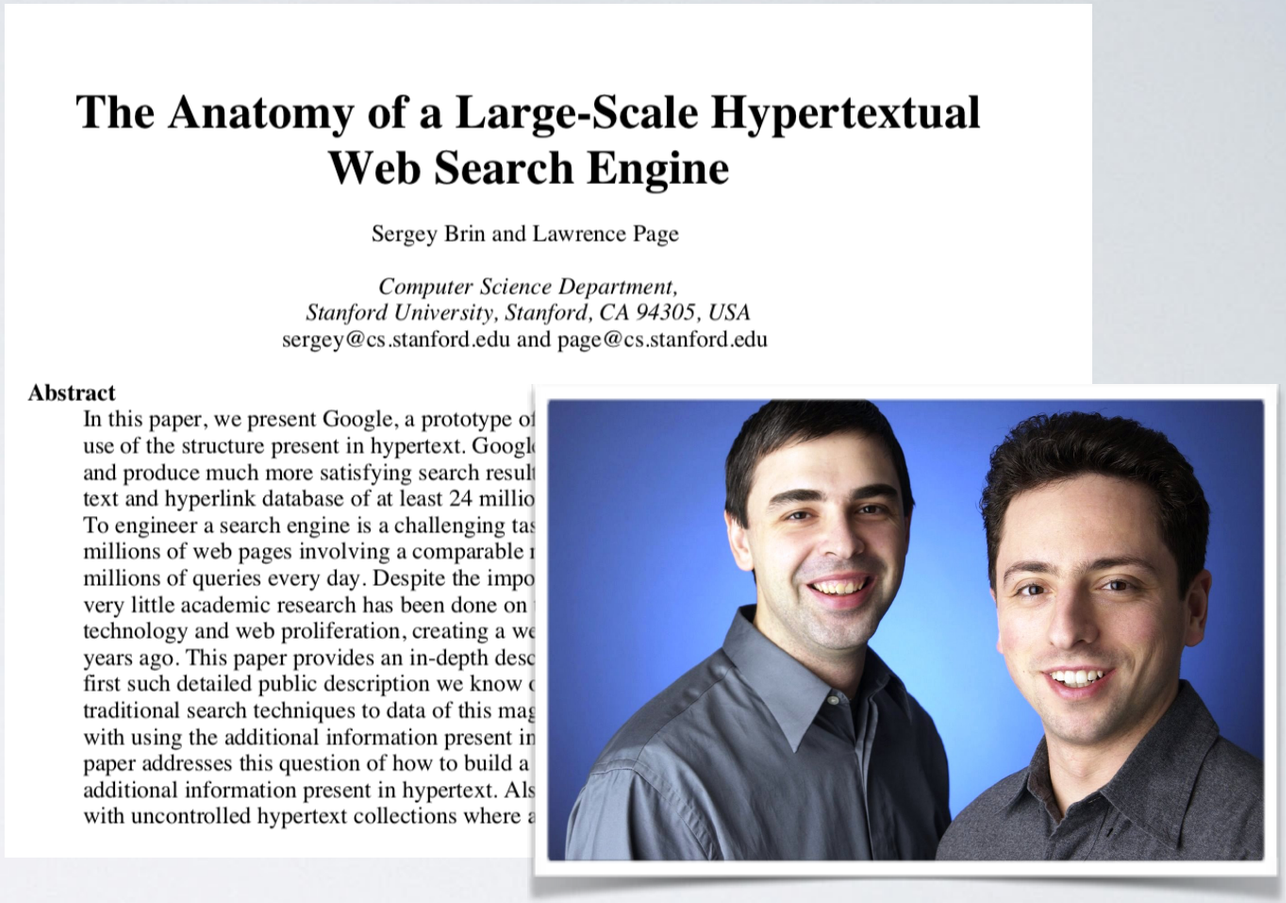
Search Engines

- Idea #1
  - build a dictionary that maps keywords to URLs
  - use hash tables or binary search trees (see end of Lecture 05)
  - what's the problem with this approach?
    - some keywords will have too many URLs to check
    - let's rank the pages by relevance!
- Q: how do we rank pages by relevance?

Search Engines

- Rank by frequency
  - build a dictionary that maps keywords to URLs
  - use hash tables or binary search trees (see end of Lecture 05)
  - store URLs ranked by the # of times keyword appears in page

- Q: Is this a good idea?
  - Why or why not?



The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page

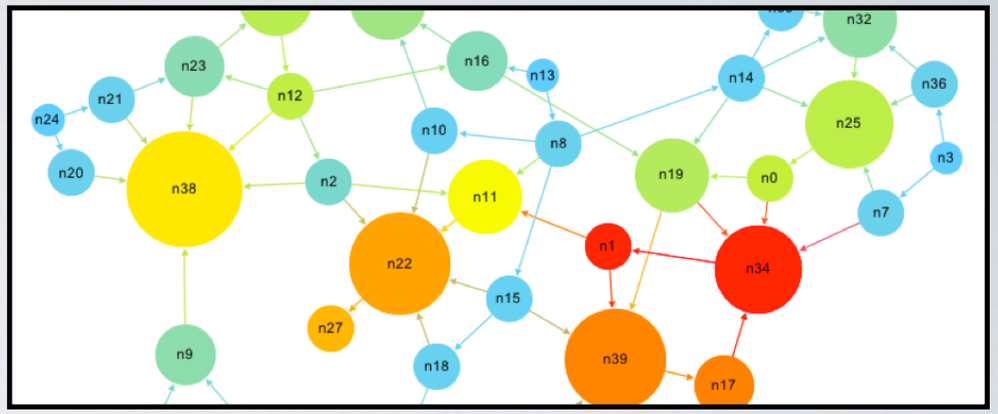
Computer Science Department,
Stanford University, Stanford, CA 94305, USA
sergey@cs.stanford.edu and page@cs.stanford.edu

Abstract

In this paper, we present Google, a prototype of
use of the structure present in hypertext. Google
and produce much more satisfying search result
text and hyperlink database of at least 24 millio
To engineer a search engine is a challenging tas
millions of web pages involving a comparable
millions of queries every day. Despite the impo
very little academic research has been done on
technology and web proliferation, creating a we
years ago. This paper provides an in-depth desc
first such detailed public description we know
traditional search techniques to data of this mag
with using the additional information present in
paper addresses this question of how to build a
additional information present in hypertext. Als
with uncontrolled hypertext collections where a

- Sergey Brin & Larry Page
- PhD students doing research in information retrieval
- noticed that links were important too!
- intuition that links conveyed information about importance
- But what exactly? and how can you make use of links?

PageRank



- How does PageRank work?
- Why does it work?
- How do you implement it efficiently?
  - Google indexes "hundreds of billions" of pages
  - answers and ranks in `0.5` seconds
  - processes `40,000` queries a second
  - `3.5` billion per day
- Using clever **algorithms** and **data structures**!

From Lecture 01

The PageRank Algorithm

- Views WWW as a directed graph **`G=(V,E)`**
  - web pages are the vertices
  - hyperlinks are the edges
- High-level idea
  - algorithm works by rounds
  - think of the pagerank of a page as some amount of fluid
  - at each round a page pushes its pagerank/fluid to the pages it links to

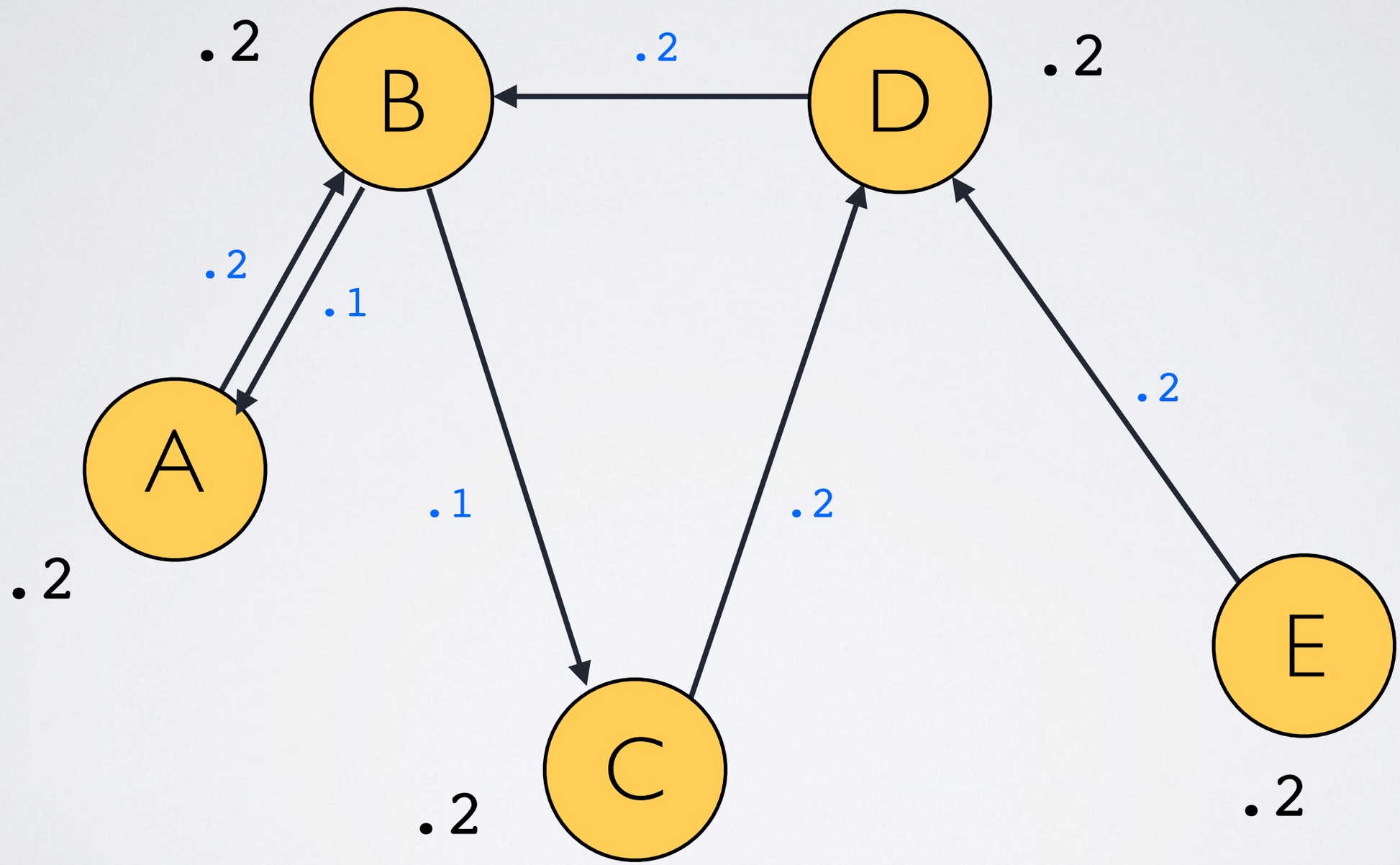
The Basic PageRank Algorithm

- At every round
  - each vertex splits its PR evenly among its outgoing edges
  - each vertex receives PR from all its incoming edges
  - this is done using an *update rule* which is run on every vertex
- The update rule for Basic PageRank is:

$$\mathsf{PR}(v) = \sum_{u \in \mathsf{in}(v)} \frac{\mathsf{PR}(u)}{|\mathsf{out}(u)|}$$

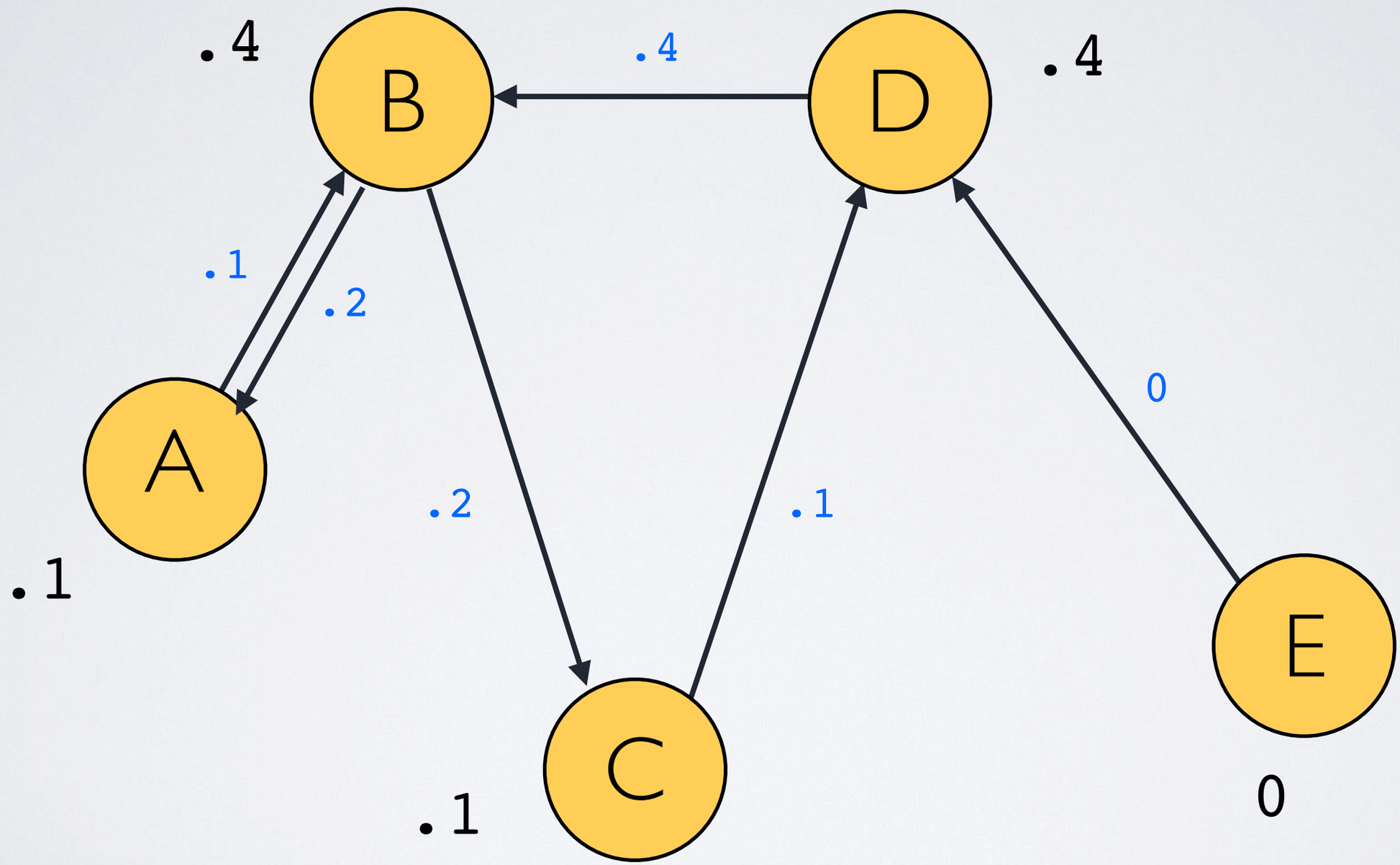
Basic PageRank: Example 1

Round 1



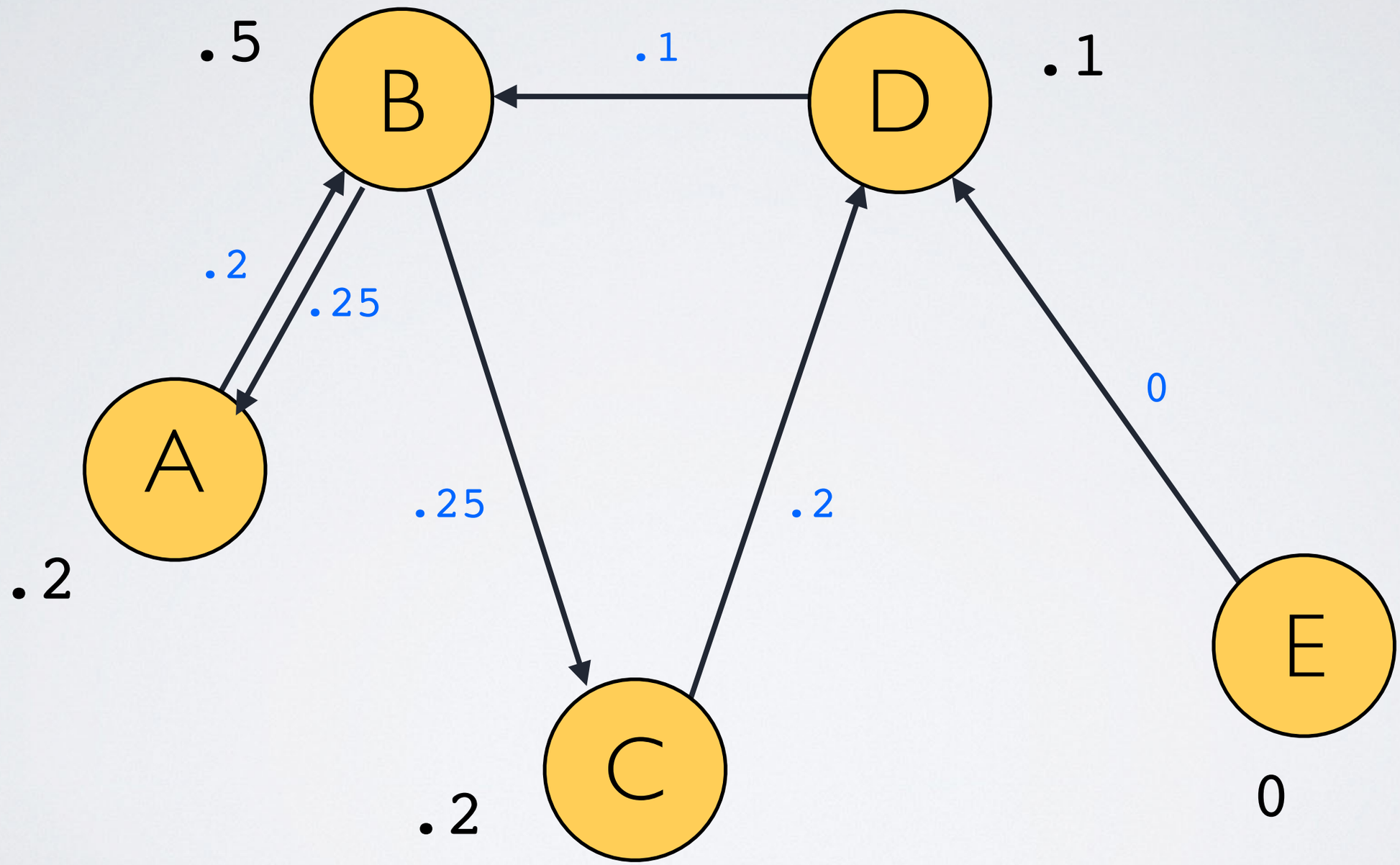
Basic PageRank: Example 1

Round 2



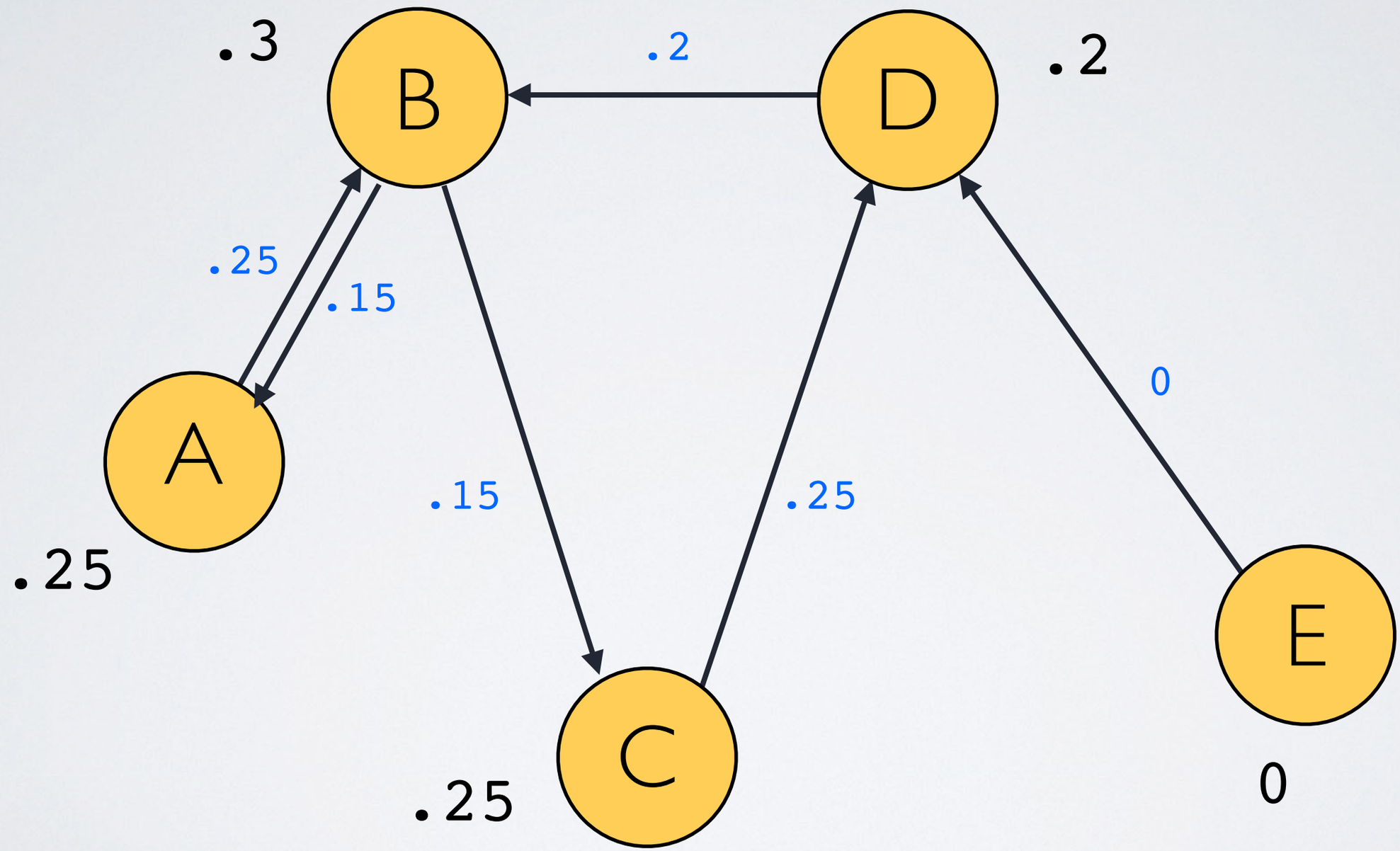
Basic PageRank: Example 1

Round 3



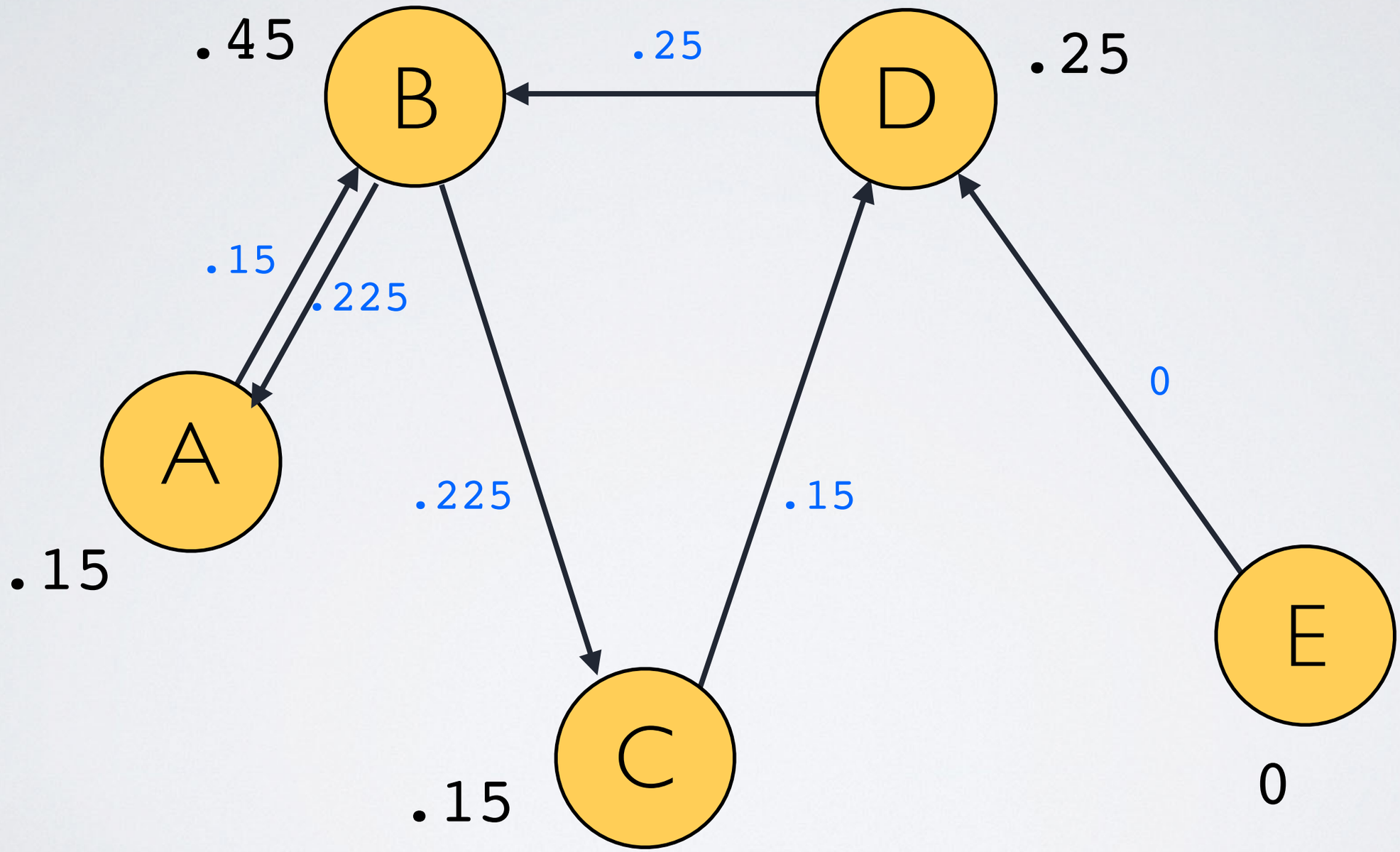
Basic PageRank: Example 1

Round 4



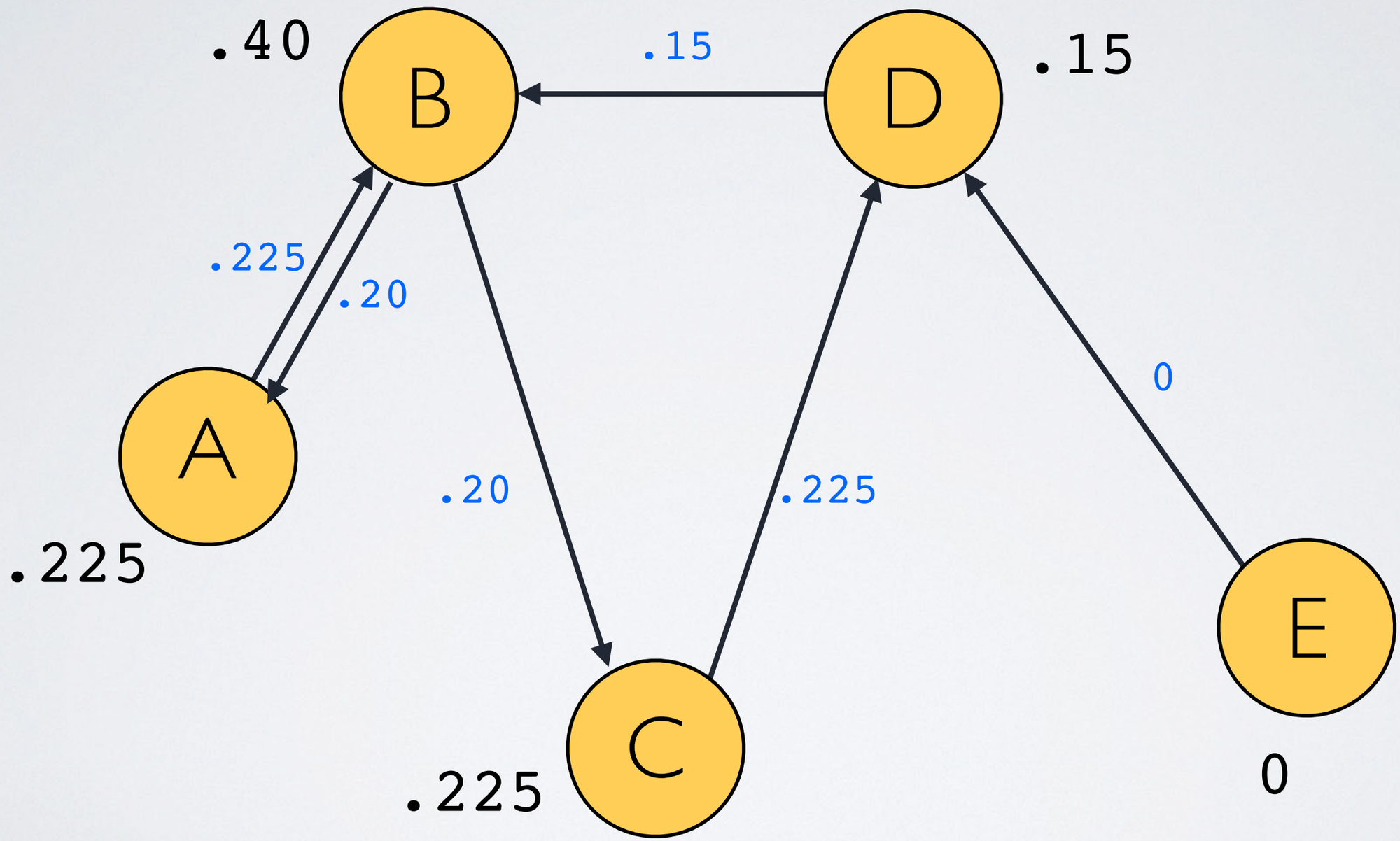
Basic PageRank: Example 1

Round 5



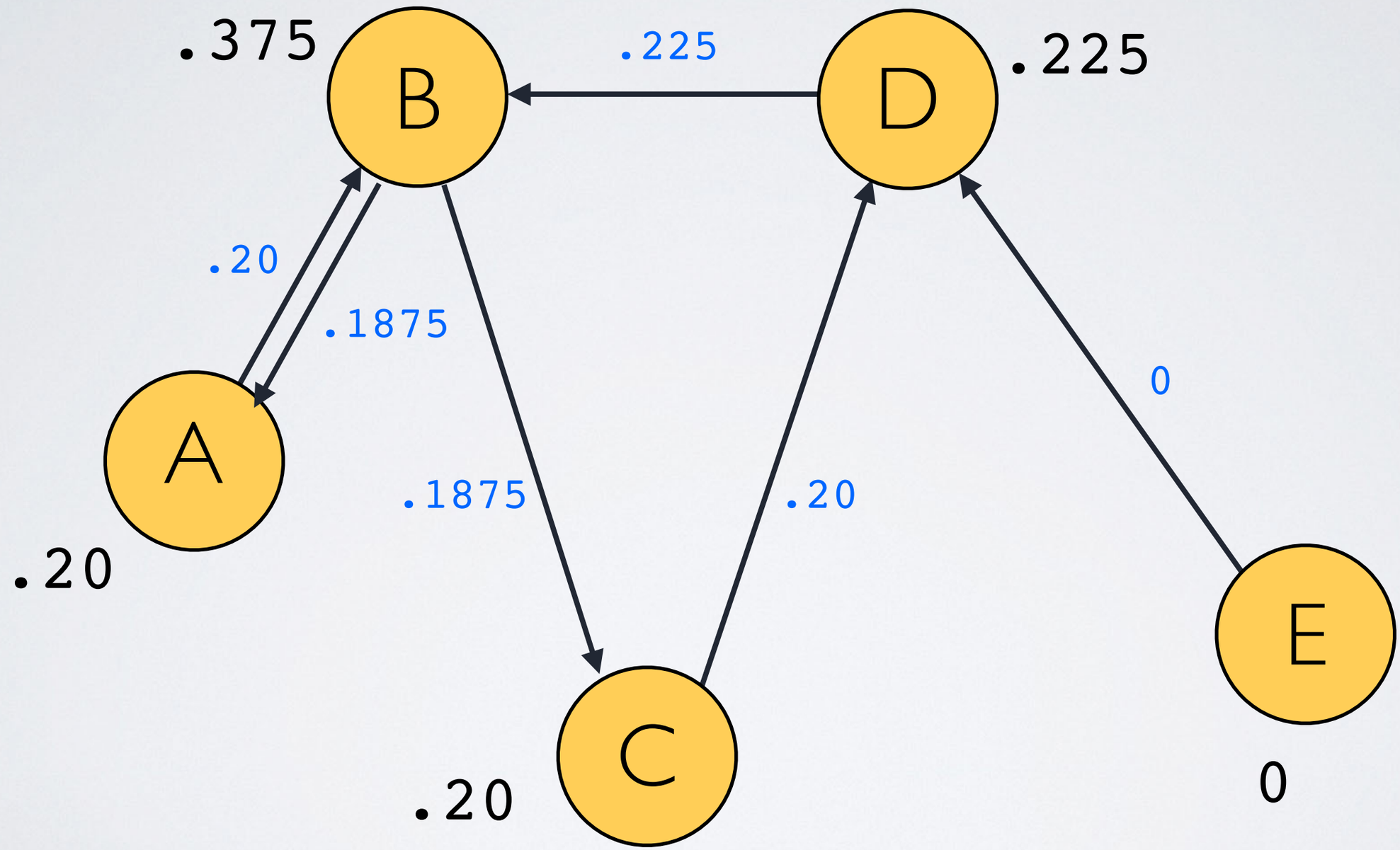
Basic PageRank: Example 1

Round 6



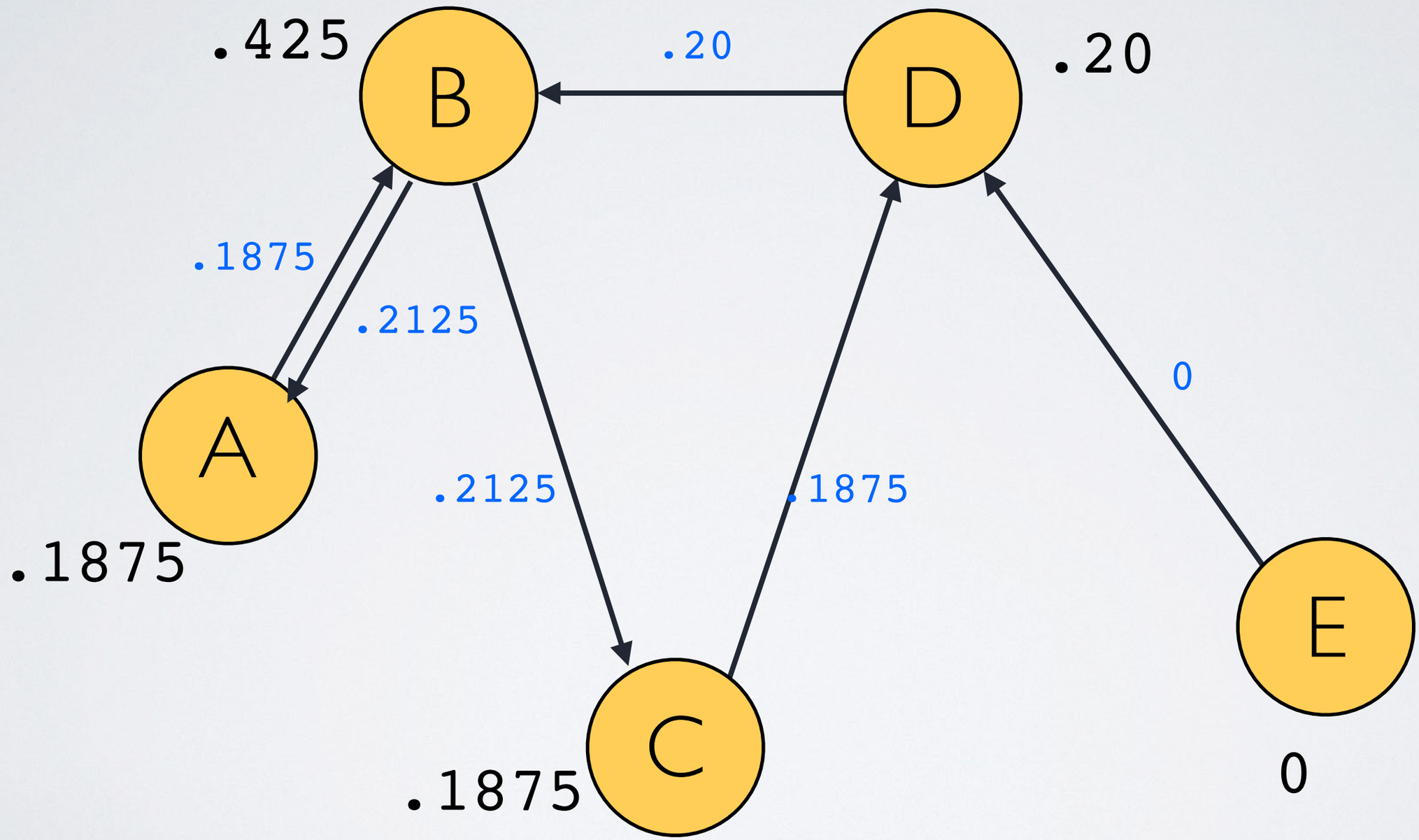
Basic PageRank: Example 1

Round 7



Basic PageRank: Example 1

Round 8



Basic PageRank

- At Round 8
  - B has pagerank `.425`
  - D has pagerank `.2`
  - A has pagerank `.1875`
  - C has pagerank `.1875`
  - E has pagerank `0`
- What happens if we keep going?
  - does the ranking stabilize or will C have pagerank higher than D?
  - can E end up with non-zero pagerank?
  - for certain graphs, if we keep going long enough pageranks will stabilize

Observations

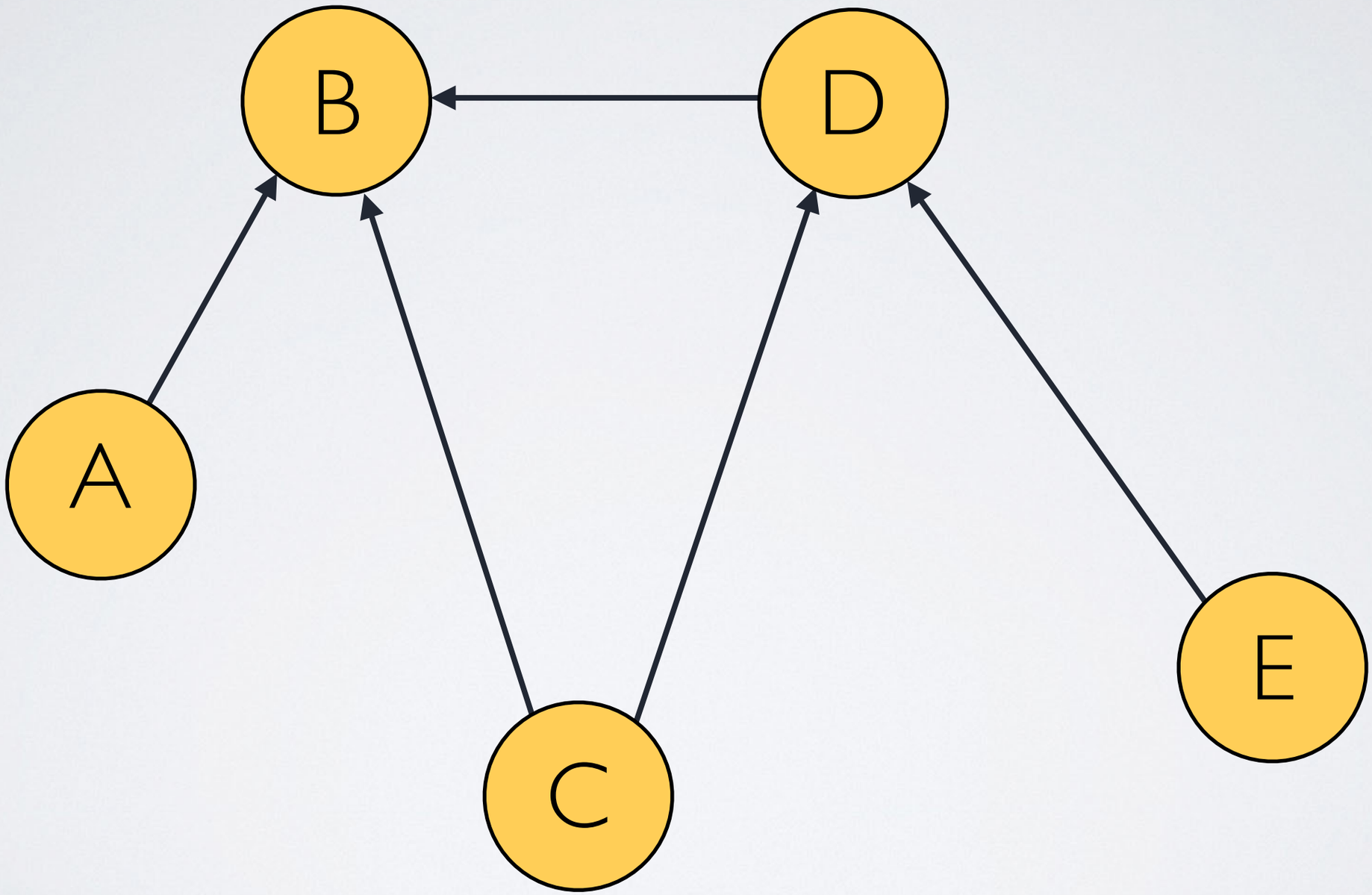
- The sum of all pageranks always equals 1
  - pagerank is moved around but never created or destroyed
- Pages with many incoming edges accumulate more pagerank (e.g., A, D)
  - even better if incoming edges from pages with high pagerank (e.g., B)
- Pages with no incoming edges lose all their pagerank (e.g., E)
- Intuitively
  - the more a page is linked to, the higher its rank
  - the more a page is linked to by high-ranked pages, the higher it ranks

Basic PageRank

```
BasicPageRank(G, k):
   for v in V:
      v.rank = 1/|V|
   for i from 1 to k:
      for v in V:
         v.prevrank = v.rank
      for v in V:
         v.rank = 0
         for u in v.incoming:
            v.rank = v.rank + u.prevrank/|u.outgoing|
```

- runtime of
  - $O(|V|+k(|V|+|E|))=O(|V|+|E|)$
  - assuming k is a constant

Basic PageRank: Example 2



Basic PageRank on Example 2

```
BasicPageRank(G, k):
   for v in V:
      v.rank = 1/|V|
   for i from 1 to k:
      for v in V:
         v.prevrank = v.rank
      for v in V:
         v.rank = 0
         for u in v.incoming:
            v.rank = v.rank + u.prevrank/|u.outgoing|
```

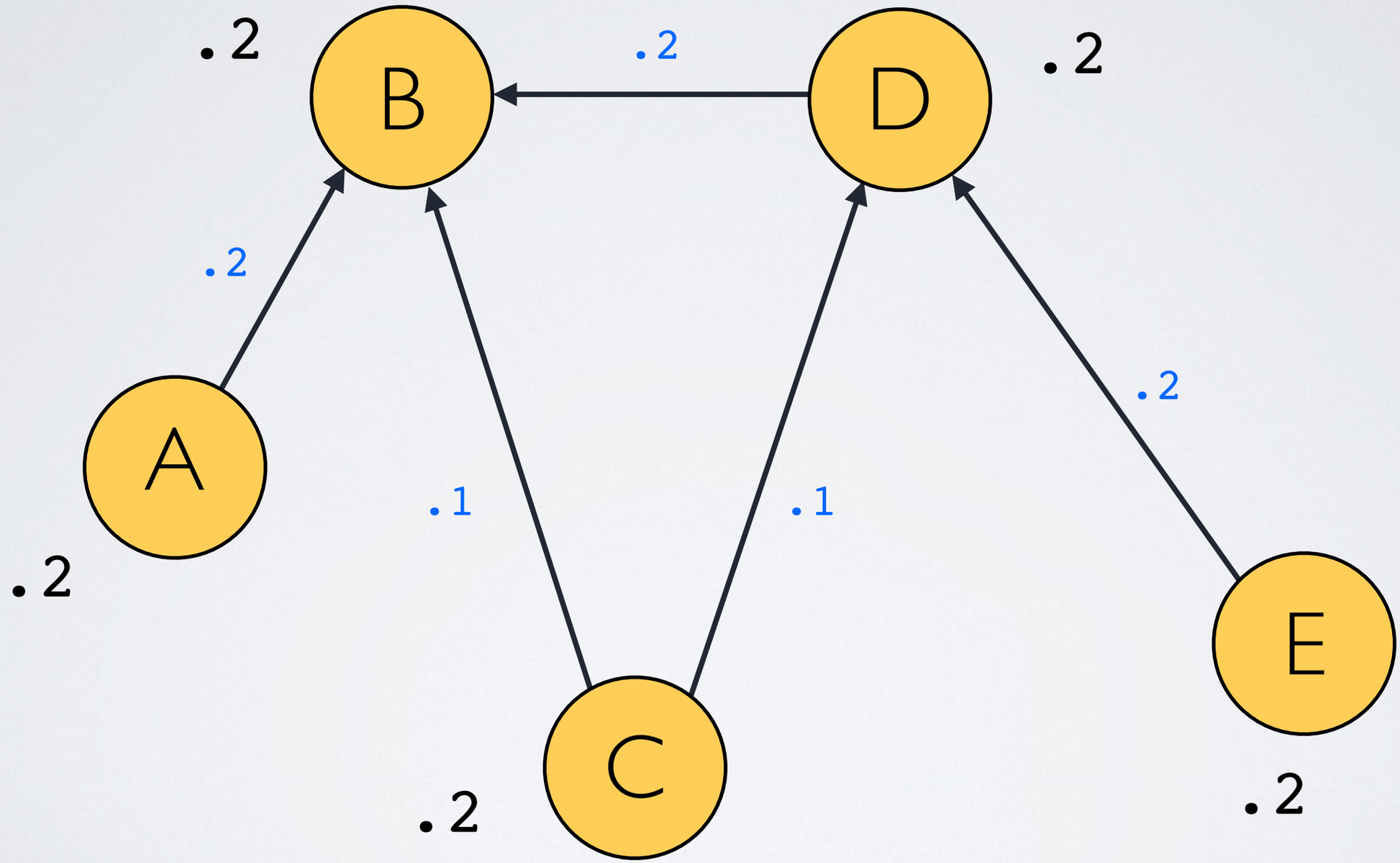
k = 3

Activity #1

3 min

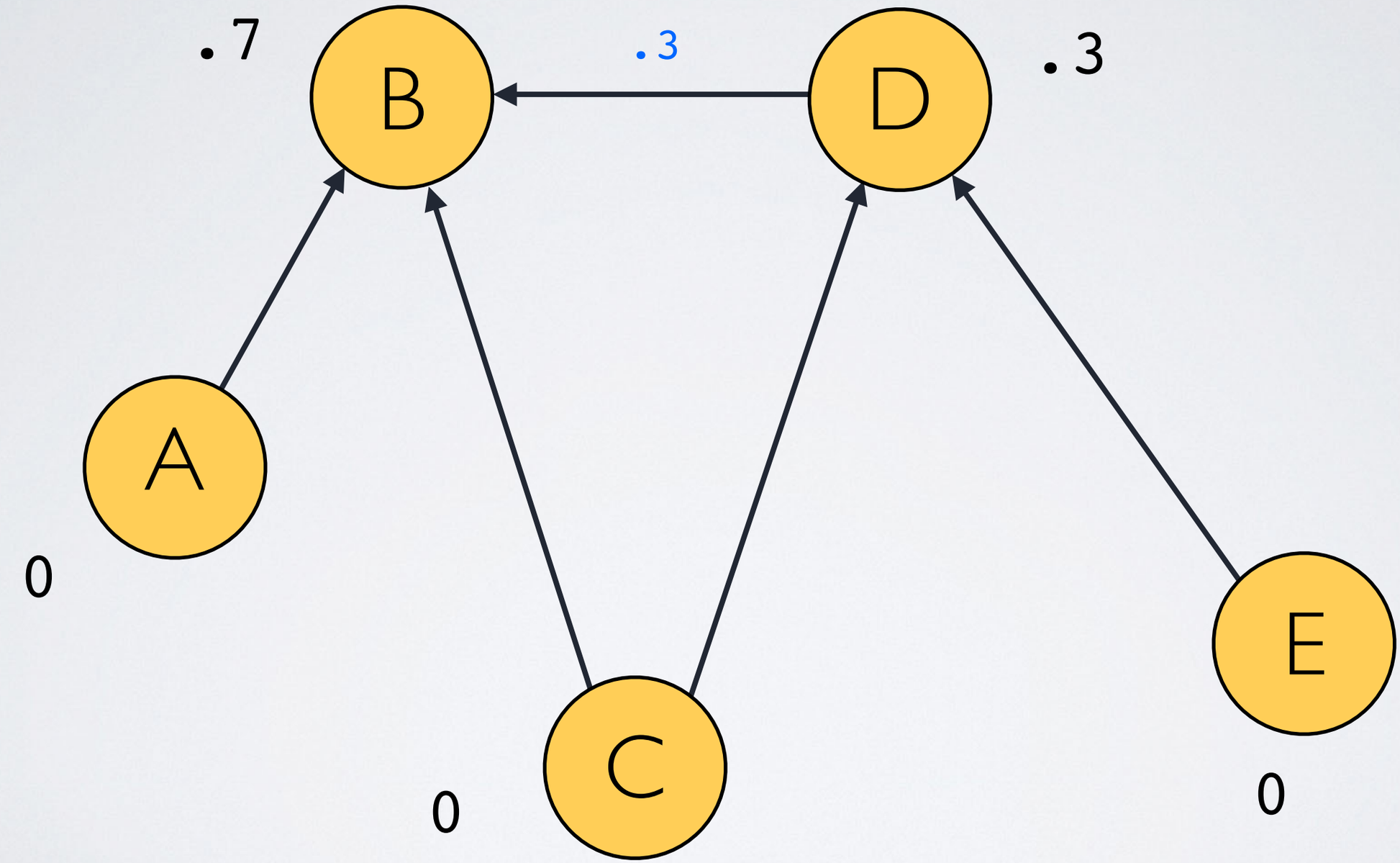
Basic PageRank: Example 2

Round 1



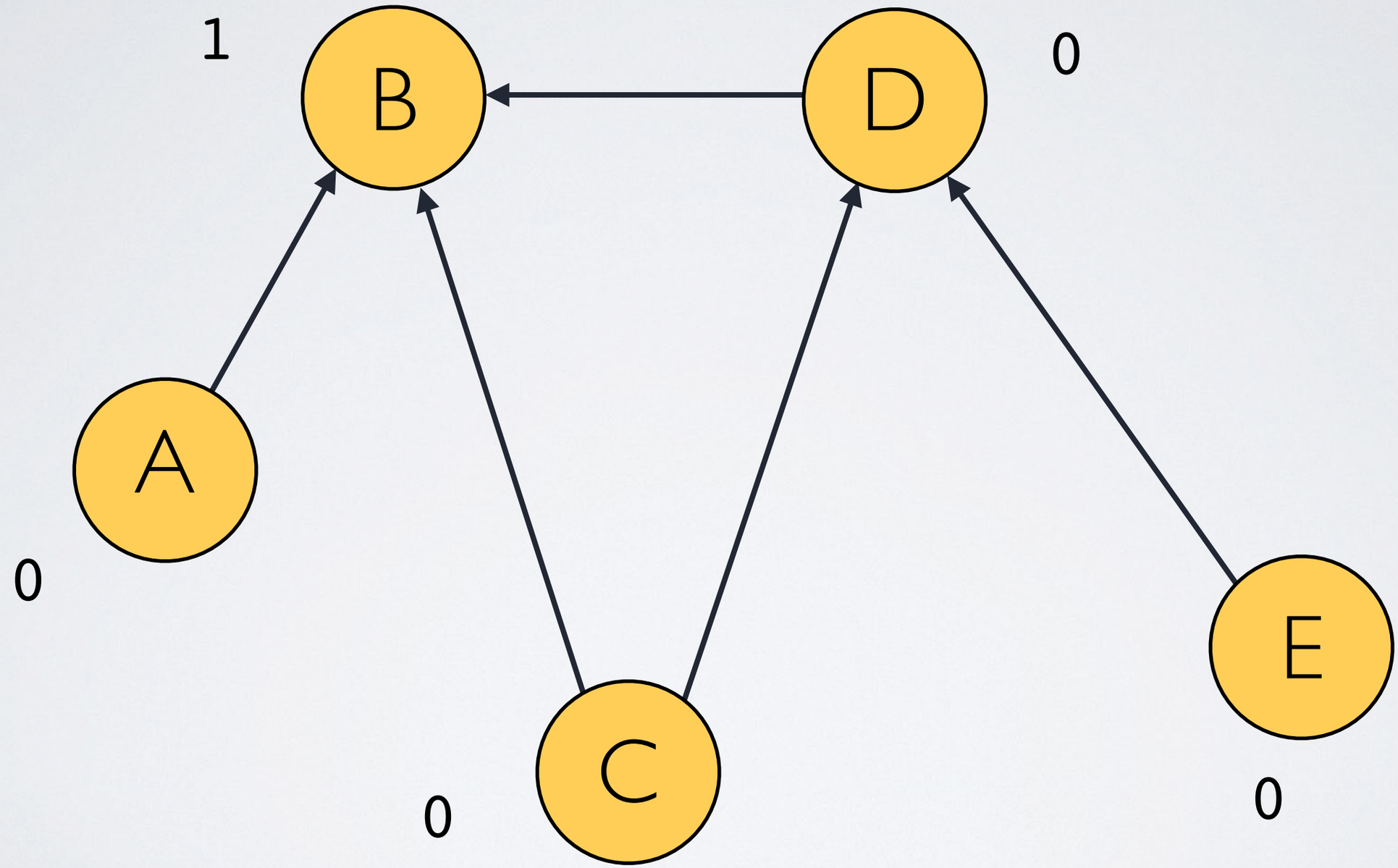
Basic PageRank: Example 2

Round **2**



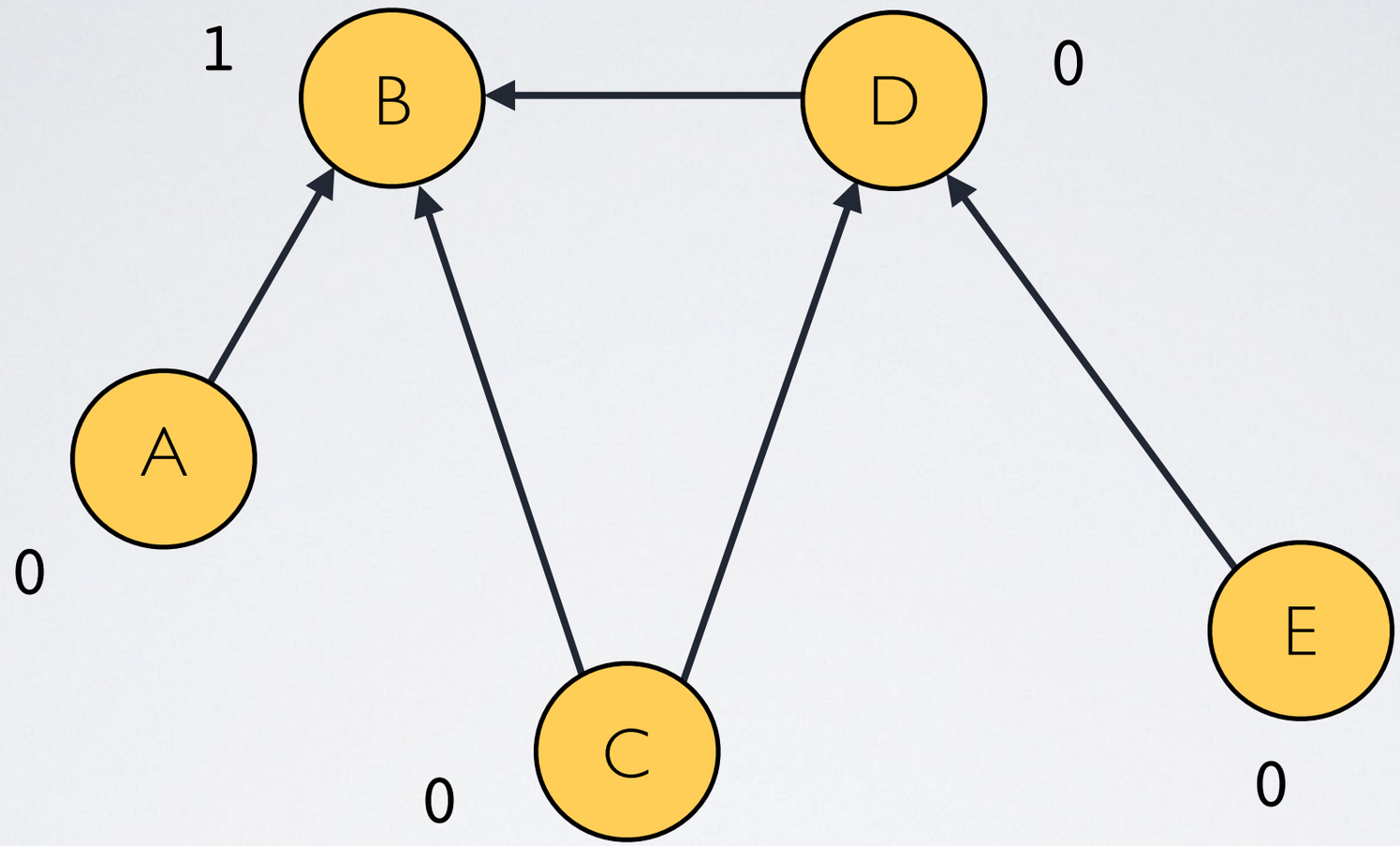
Basic PageRank: Example 2

Round 3



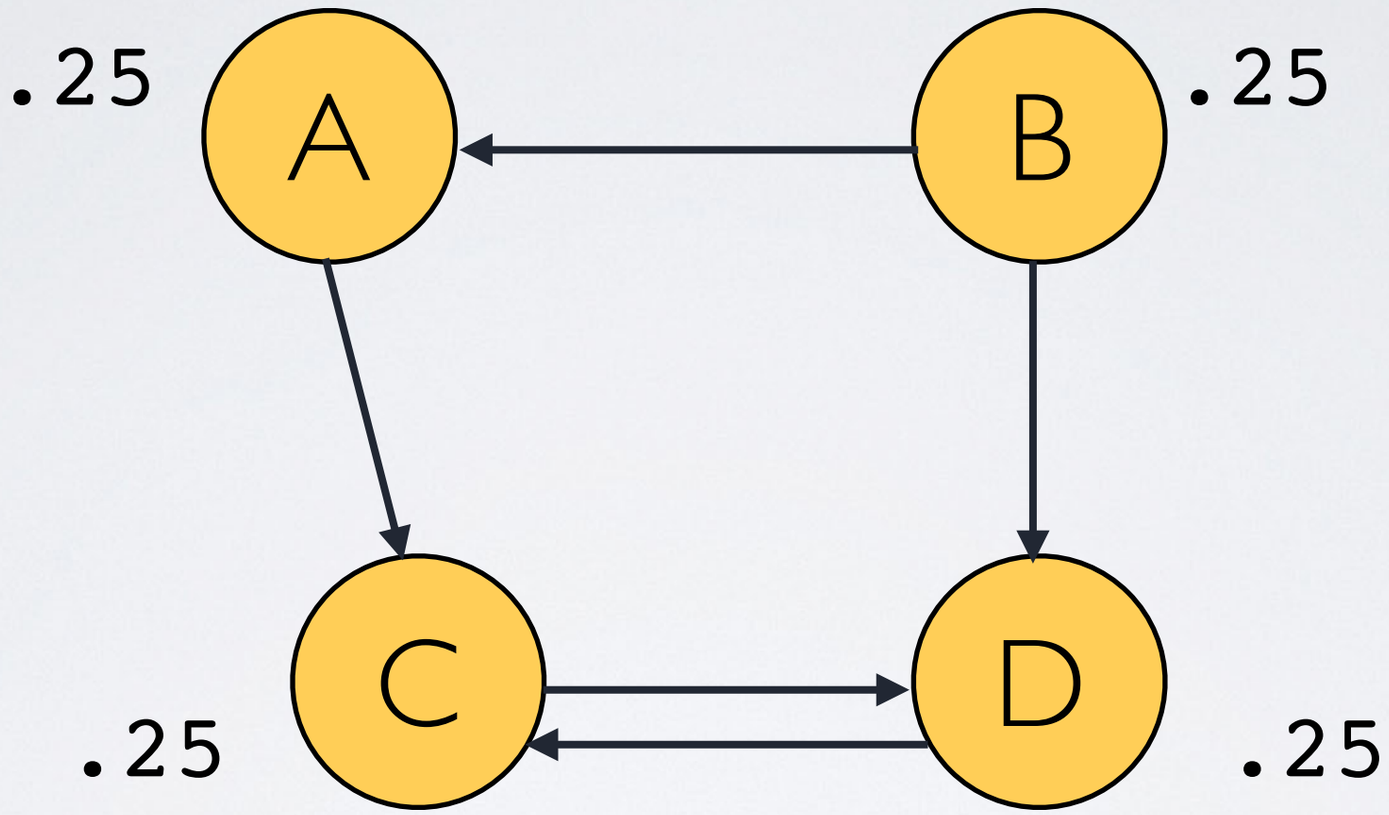
What's going on?

Basic PageRank: Example 2

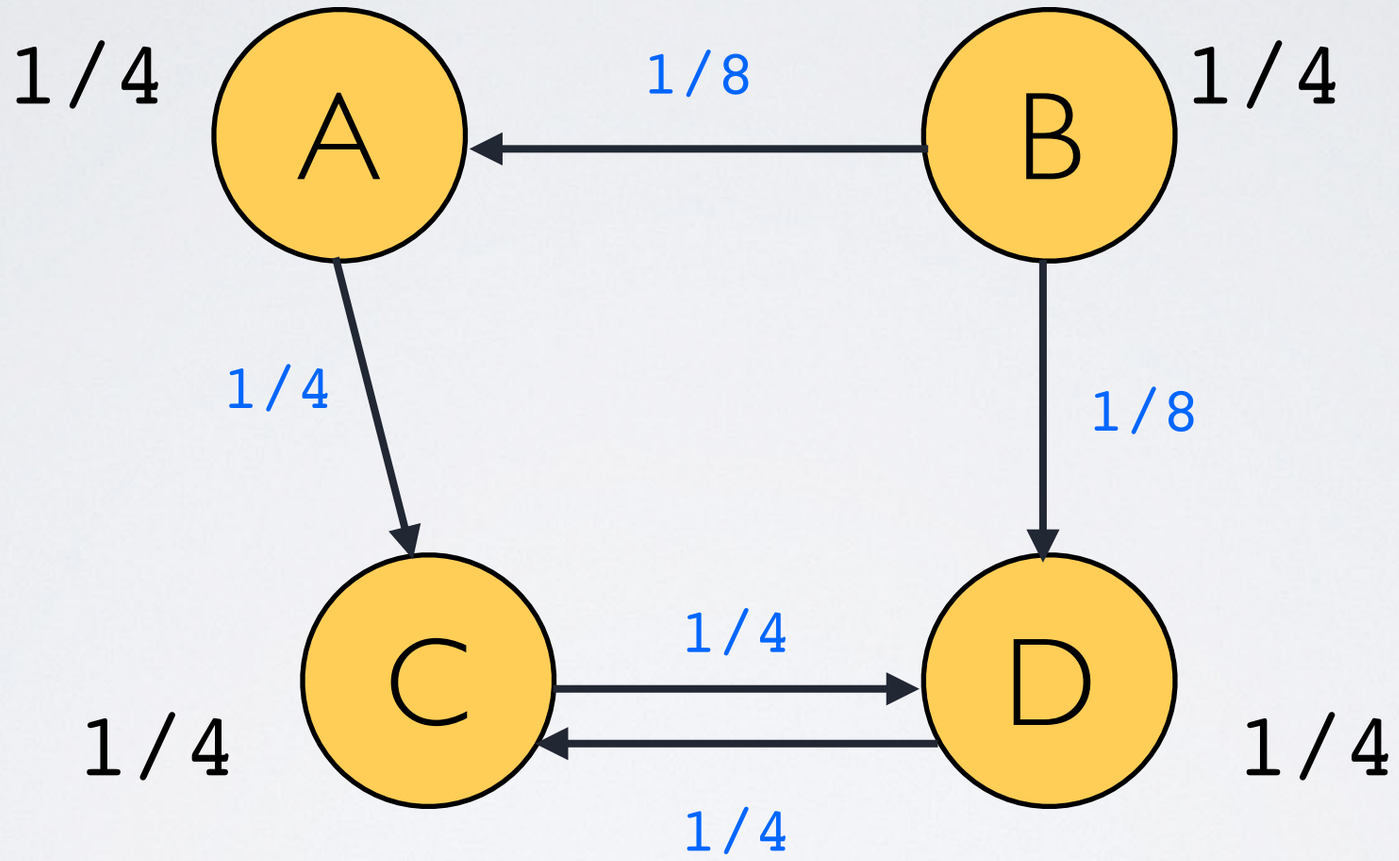


- All the pagerank got trapped at B
  - happens if graph has sinks (i.e., nodes w/ no outgoing edges)

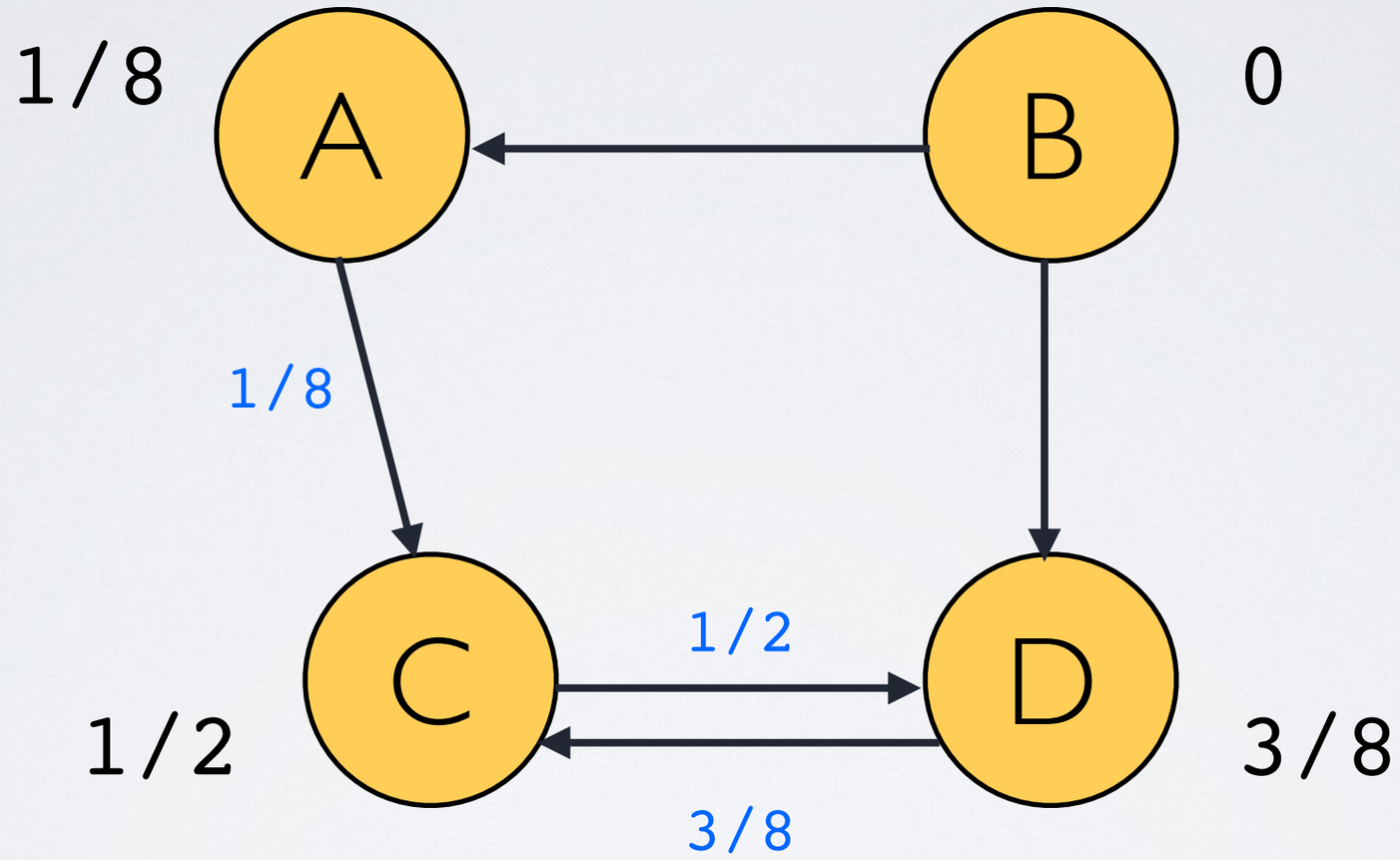
Basic PageRank: Example 3



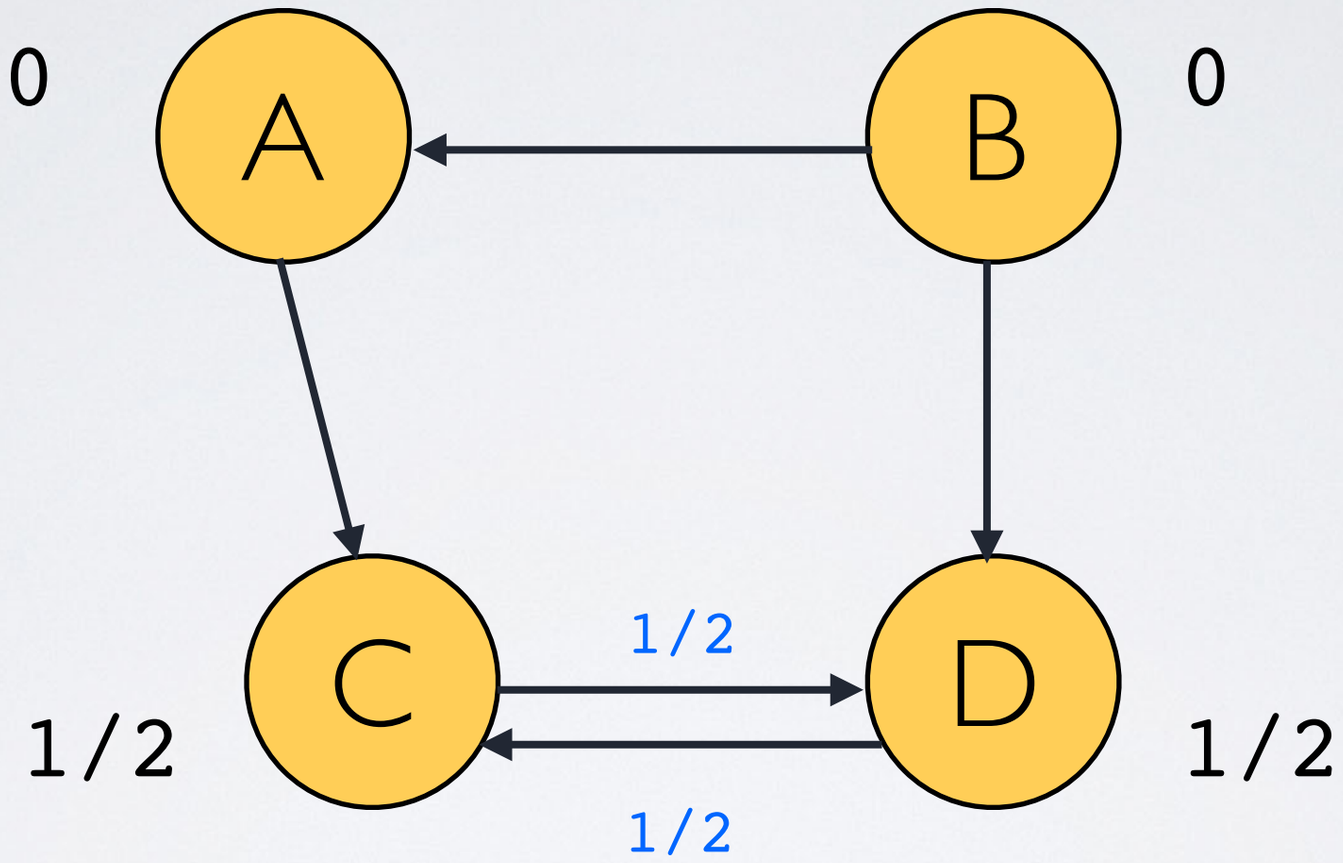
Basic PageRank: Example 3



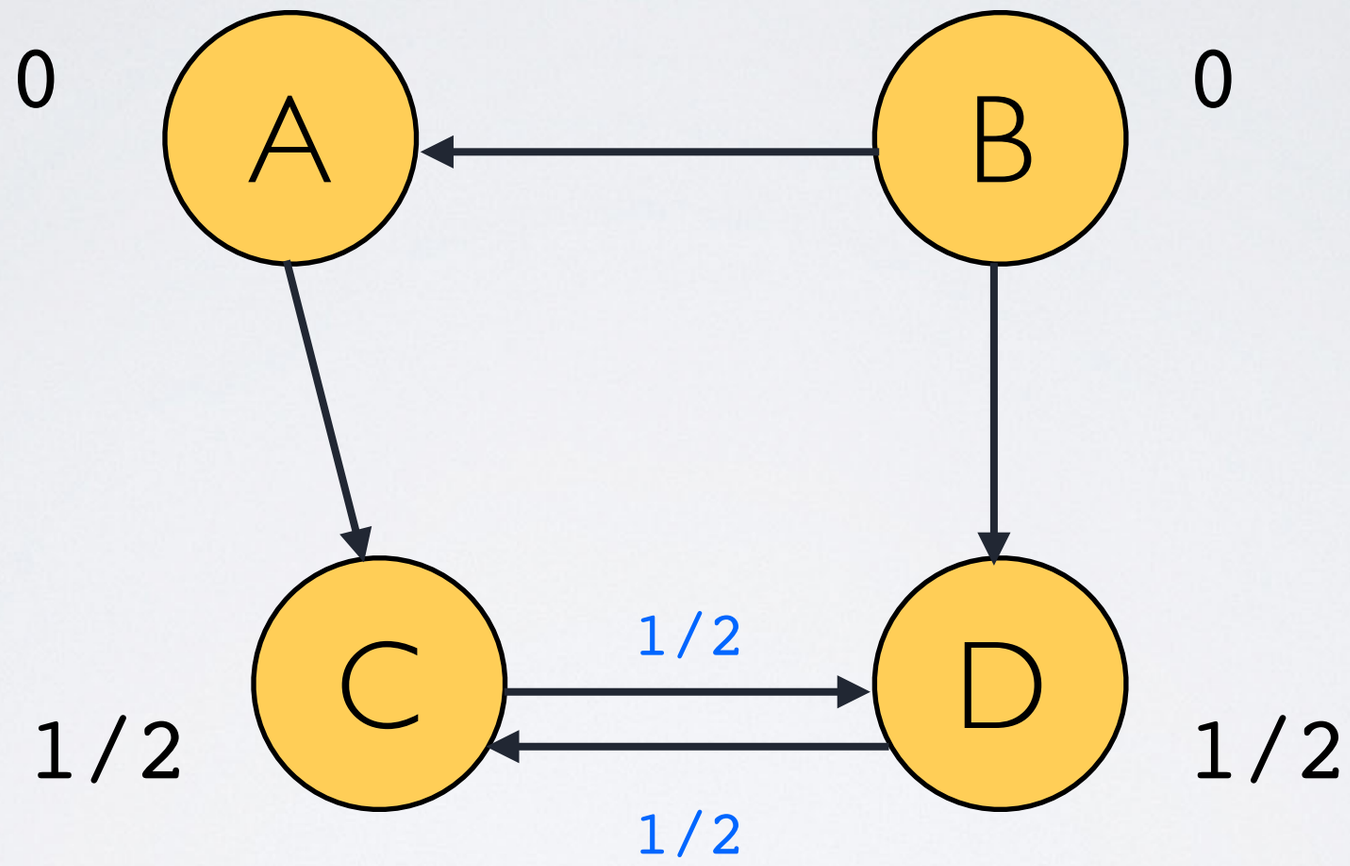
Basic PageRank: Example 3



Basic PageRank: Example 3

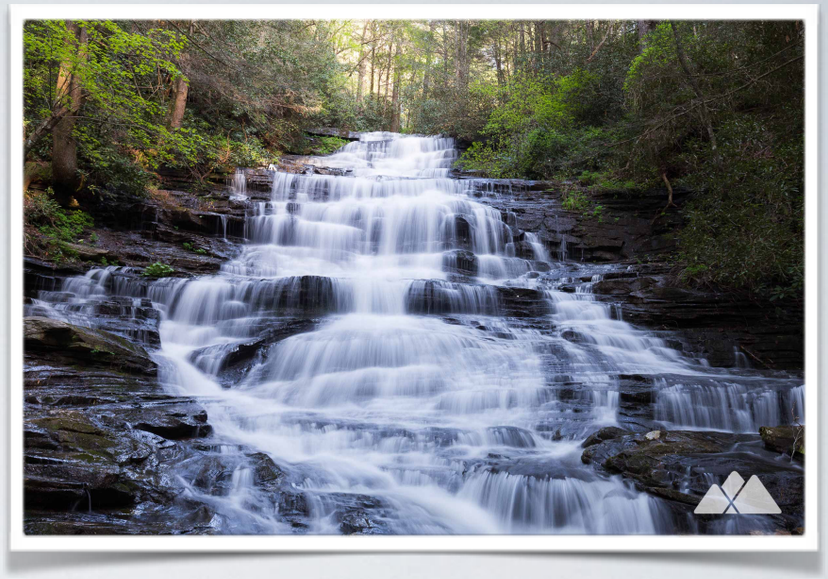


Basic PageRank: Example 3



- All the pagerank got trapped at C and D

Basic PageRank



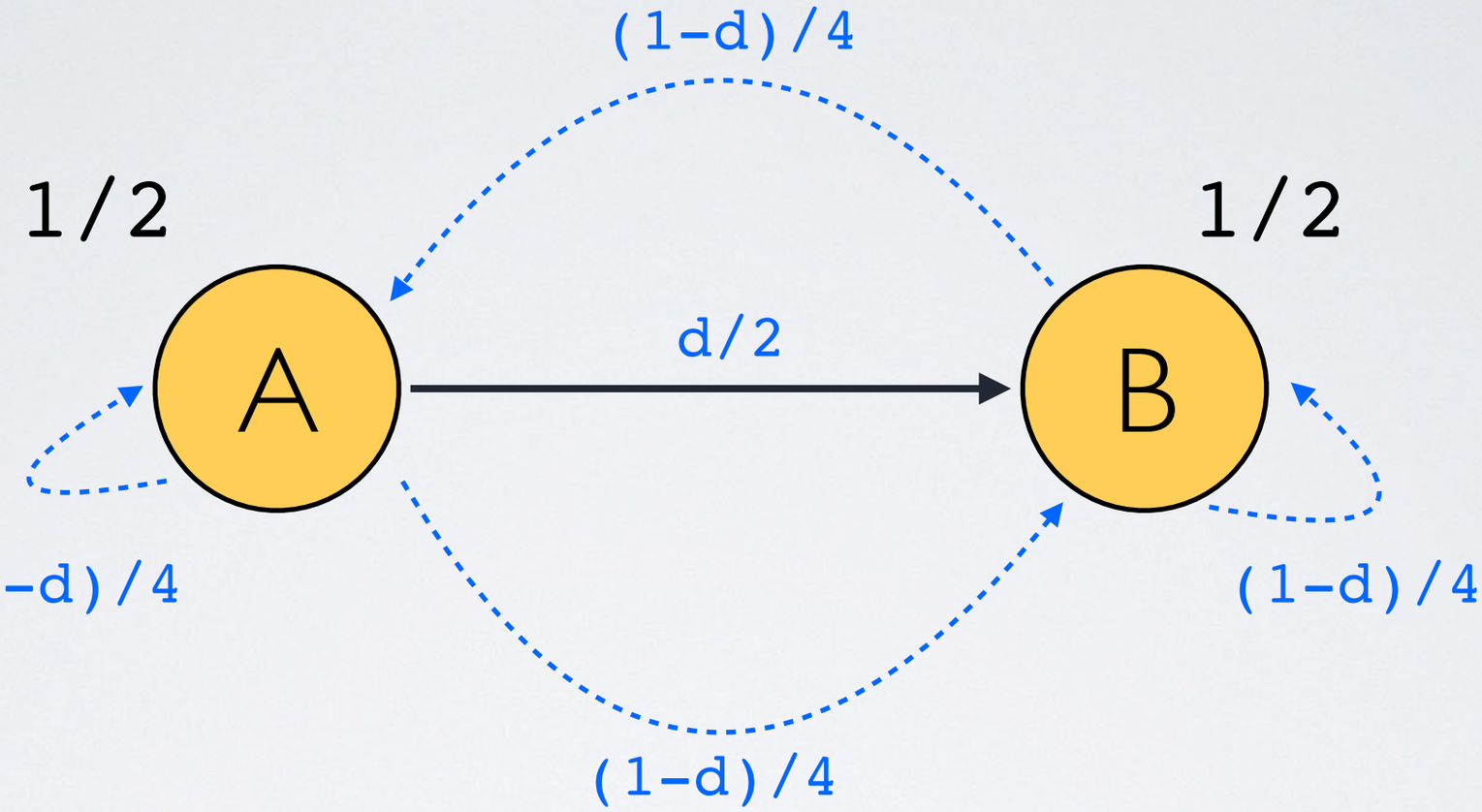
- Basic PageRank doesn't work for certain graphs
  - e.g.: graphs with sinks or with cycles with no outgoing edges
  - all the pagerank gets trapped there
- How do we handle "rank traps"?
- Water flows down from high elevation to low elevation
  - why doesn't all the water end up at the lowest points on Earth?
  - because some of the water evaporates…
  - …and rains back down on the high elevation points

Handling Rank Traps

- Let's make some of the pagerank evaporate!
  - We need a new *update rule*
- In basic update rule, nodes gave all their pagerank to neighbors
- In new update rule, a node will
  - give a **`d`** fraction of its PR to its neighbors (split evenly)
  - give a **`1-d`** fraction of its PR to all other nodes (split evenly)
  - this guarantees that pagerank doesn't accumulate anywhere
  - **`d`** is usually set to **`.85`**

What happens if the node is a sink?

Disappearing PageRank



Disappearing PageRank

- The sum of the pageranks *does not* sum to `1`:

$$\frac{(1-d)}{2} + \frac{1}{2} = 1 - \frac{d}{2} < 1$$

  - since `0 < d < 1`

- We lost `d/2` of B's pagerank when we updated

Handling Sinks

- There are several ways to handle sinks
- The simplest is to modify the graph as follows
  - if $\mathbf{v}$ is a sink, add an edge from $\mathbf{v}$ to every node in the graph
    - This includes an edge from $\mathbf{v}$ to itself
- Then use the update rule we described on slide #38

The Real PageRank Algorithm

- Add edges connecting every sink to every node
- At every round each vertex
  - splits a **`d`** fraction of its PR evenly among its outgoing edges
  - splits a **`(1-d)`** fraction of its PR evenly among all nodes
  - receives PR from its incoming edges & from its share of the "evaporated" pageranks of all nodes
- **`d`** is called the *damping factor* & is usually set to **`.85`**

The Real PageRank Algorithm

- At every round the PR of each vertex **v** is updated using:

$$\mathsf{PR}(v) = \left(\sum_{u \in \mathsf{in}(v)} \frac{d \cdot \mathsf{PR}(u)}{|\mathsf{out}(u)|} \right) + \sum_{u \in V} \frac{(1-d) \cdot \mathsf{PR}(u)}{|V|}$$

1. nodes with edges pointing to **v**

2. **d** fraction of **u**'s PR

3. number of edges leaving u

4. **(1-d)** fraction of u's PR

$$= \left(d \cdot \sum_{u \in \mathsf{in}(v)} \frac{\mathsf{PR}(u)}{|\mathsf{out}(u)|} \right) + \frac{1-d}{|V|} \cdot \sum_{u \in V} \mathsf{PR}(u)$$

$$= \left(d \cdot \sum_{u \in \mathsf{in}(v)} \frac{\mathsf{PR}(u)}{|\mathsf{out}(u)|} \right) + \frac{1-d}{|V|} \quad \downarrow 1$$

$$= \frac{1-d}{|V|} + d \cdot \sum_{u \in \mathsf{in}(v)} \frac{\mathsf{PR}(u)}{|\mathsf{out}(u)|}$$

The Real PageRank

- Runtime of a round `O(|E|)`
- How many rounds should we run?
- Ideally until the pageranks "stabilize"
  - pageranks stop changing even though we run more rounds
- We can prove that
  - if we run for large enough number of rounds then pageranks will stabilize
  - that number could be very large for some graphs…
  - …but in practice it's usually reasonable

Alternative Sink Handling

- You can also handle sinks without modifying the graph
  - but you need a slightly different update rule

$$\mathsf{PR}(v) = \frac{1-d}{|V|} + d \cdot \left(\sum_{u \in \mathsf{in}(v)} \frac{\mathsf{PR}(u)}{|\mathsf{out}(u)|} + \sum_{u \in \mathsf{sinks}(G)} \frac{\mathsf{PR}(u)}{|V|} \right)$$

Downsides of PageRank

Activity #2

What are some possible downsides of PageRank?

2 min

Downsides of PageRank

- Storage
  - Need to store a copy of the entire web graph
  - Google estimated to store about **50** billion web pages
    - Average size of a page is **2** MB
    - …that's about of **100** petabytes (**1** PB = 2^{50} bytes)!
- Hard to compute on such a large data set
  - need to store data on clusters of **1000**'s of machines
  - need to coordinate all these machines to execute PageRank

PageRank in Practice

- Google continuously computes the pagerank of every webpage
- When you query
  - your keyword narrows down the pages to return
  - then Google ranks them by their *precomputed* pagerank
- We don't know if Google still uses the original PageRank or some variant

Other Applications of PageRank

- PageRank is also in other fields
  - Biology (studying protein interactions)
  - Neuroscience (finding importance of brain regions)
  - Engineering (finding anomalies)
  - Mathematics (analyzing graphs)
  - Sports (ranking sports teams)
  - Literature (importance/influence of books)
  - Bibliometrics (which authors are more influential)

Readings

- A film from 1976 about Andy's FRESS system and its use in *Introduction to Poetry*
  - https://archive.org/details/AndyVanDamHypertextFilm
- The book *Networks, Crowds and Markets* by Easly and Kleinberg has a great overview of PageRank
  - the evaporation metaphor comes from there!
- Other applications of PageRank
  - https://arxiv.org/pdf/1407.5107.pdf
- Size of the web
  - http://www.worldwidewebsize.com