WHERE WE ARE

- **Graphs**
  - What is a graph?
  - Graph Structures
  - Graph Analytics
- **Traversals**
- **Patterns**
- **Recursion**
- **Graph Representations**

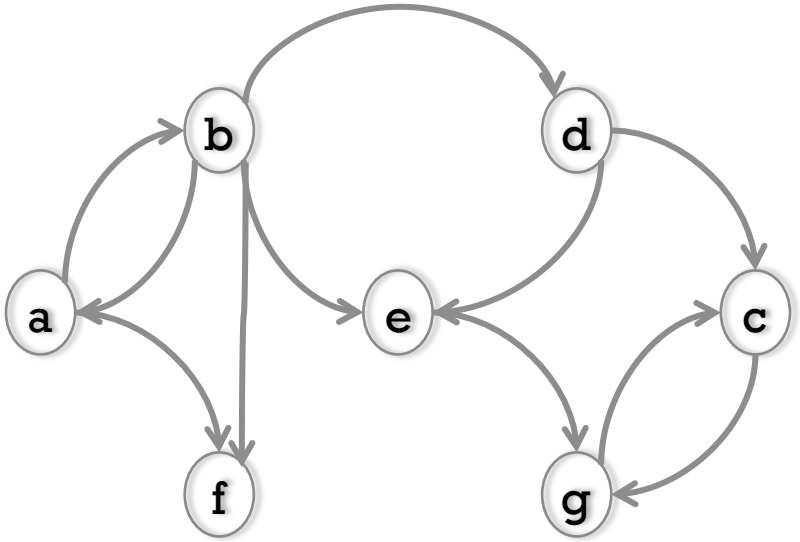
WHAT IS A GRAPH?

G = (V, E)

V is a set of vertices (synonym: nodes)

E is a set of edges.

- Each edge is a pair (v_{source}, v_{target})
- Edges may be considered directed or undirected



GRAPHS IN THE WILD

The Web

The Internet

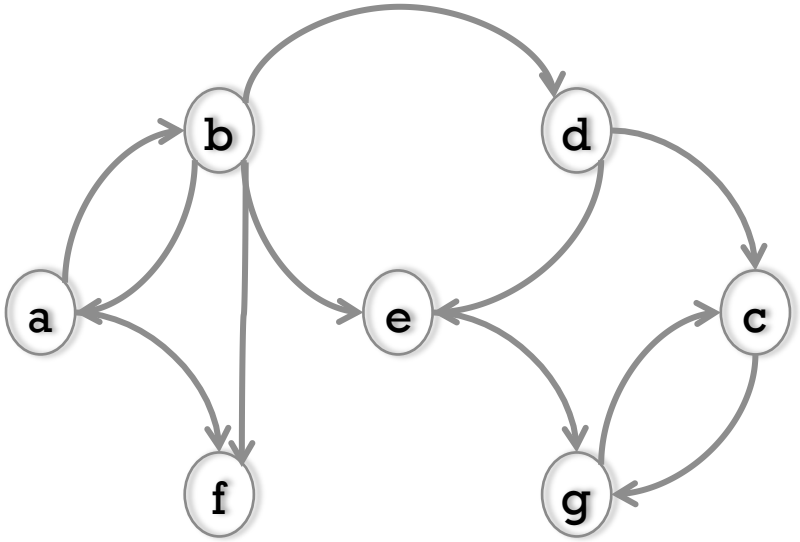
Social networks

Communication logs

- Ex: PRISM

many more – ubiquitous!

(why?)

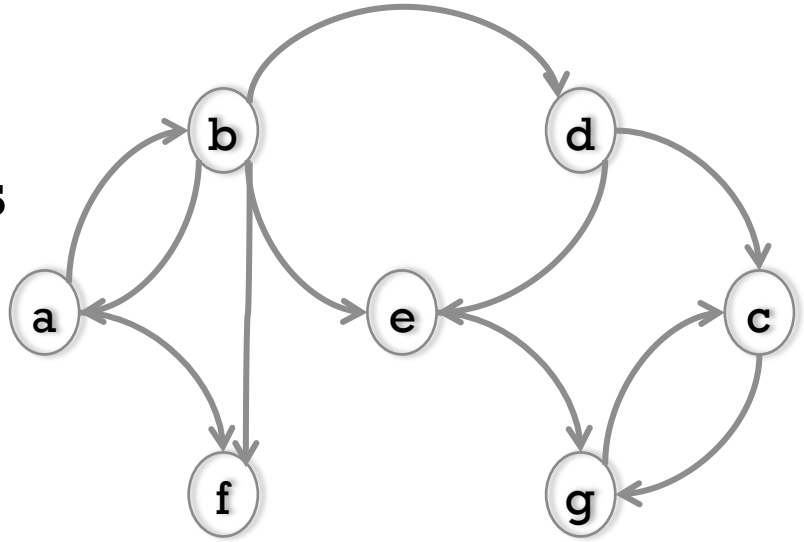


GRAPH ANALYTICS

Structural Algorithms

Traversal Algorithms

Pattern-Matching Algorithms

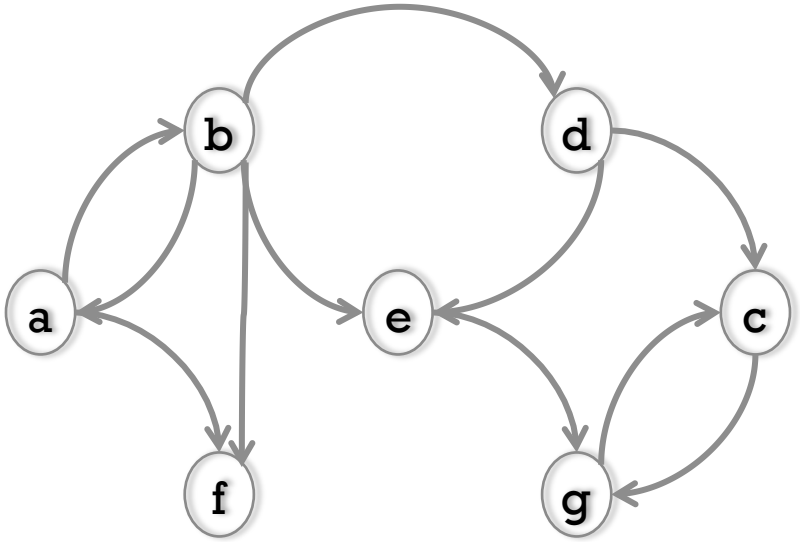


Analyzing Analytics, Bordawekar 2012

GRAPH ANALYTICS: SOME STRUCTURAL TASKS

Basic Metrics

- |V|, |E|
- |E| is more interesting than |V|
- in-degree(v), out-degree(v)



"I have a big graph"

"How many edges, and what is the highest in or out degree?"

Analyzing Analytics, Bordawekar 2012

WHERE WE ARE

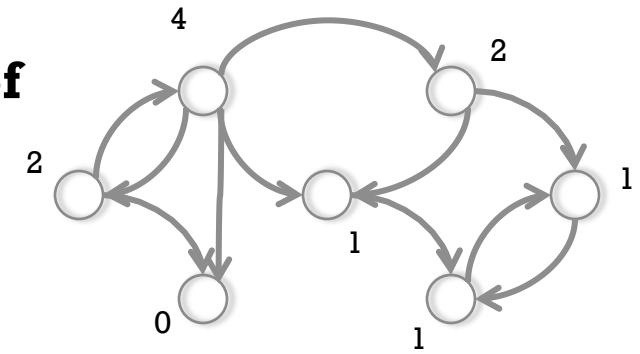
- **Graphs**
  - What is a graph?
  - Graph Structures
  - Graph Analytics
- **Traversals**
- **Patterns**
- **Recursion**
- **Graph Representations**

STRUCTURAL TASK: THE HISTOGRAM OF A GRAPH

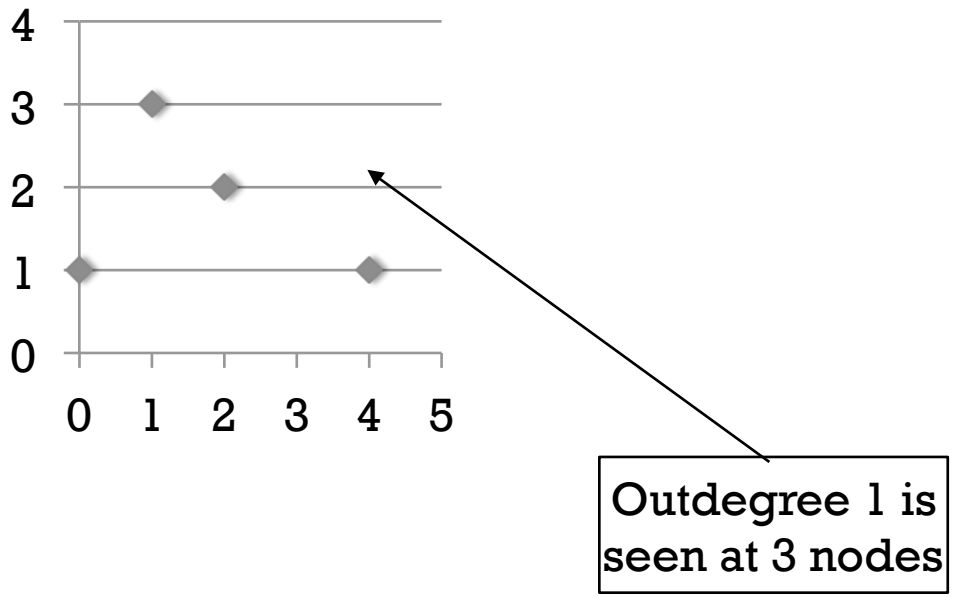
Outdegree of a vertex = number of outgoing edges

For each integer d, let n(d) = number of vertices with outdegree d

The outdegree histogram of a graph = the scatterplot (d, n(d))

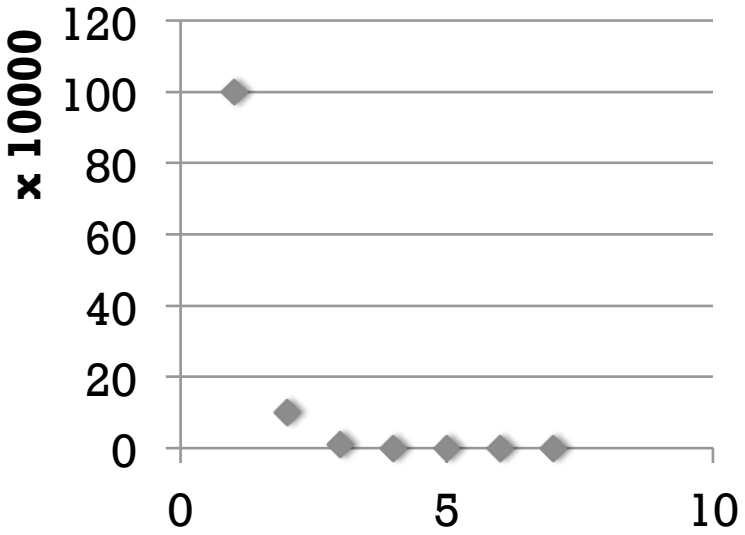


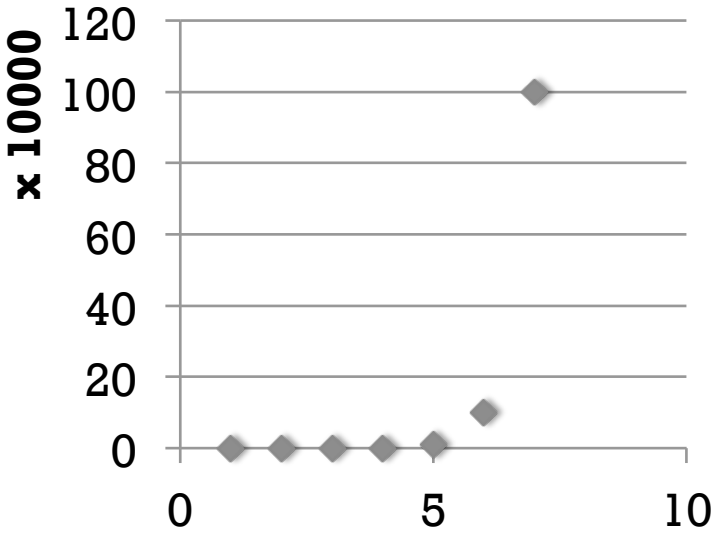
d	n(d)
0	1
1	3
2	2
3	0
4	1



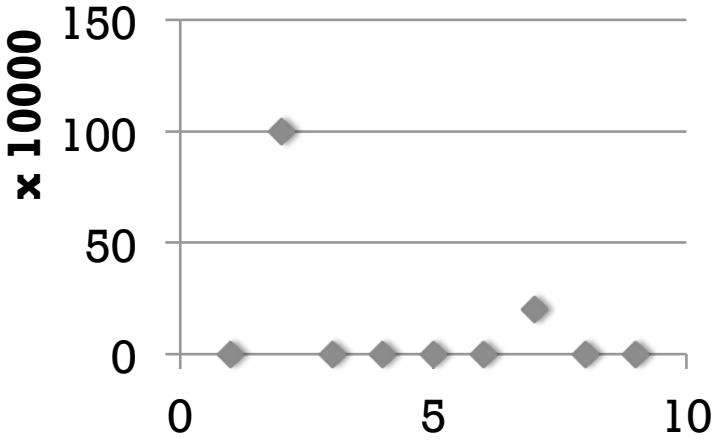
src: Dan Suciu

HISTOGRAMS TELL US SOMETHING ABOUT THE GRAPH





What can you say about these graphs?



src: Dan Suciu

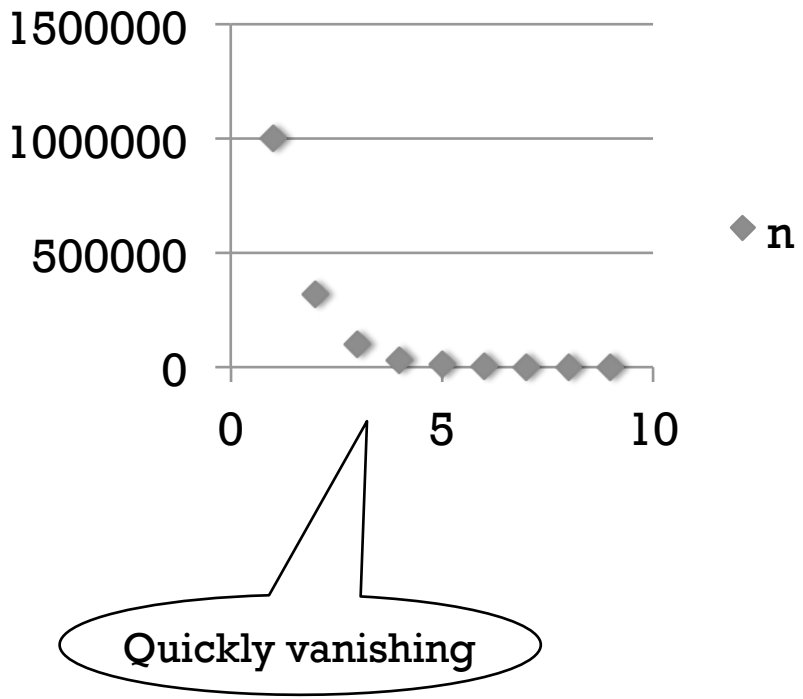
EXPONENTIAL DISTRIBUTION

$$n(d) \simeq c \left(\frac{1}{2}\right)^d$$

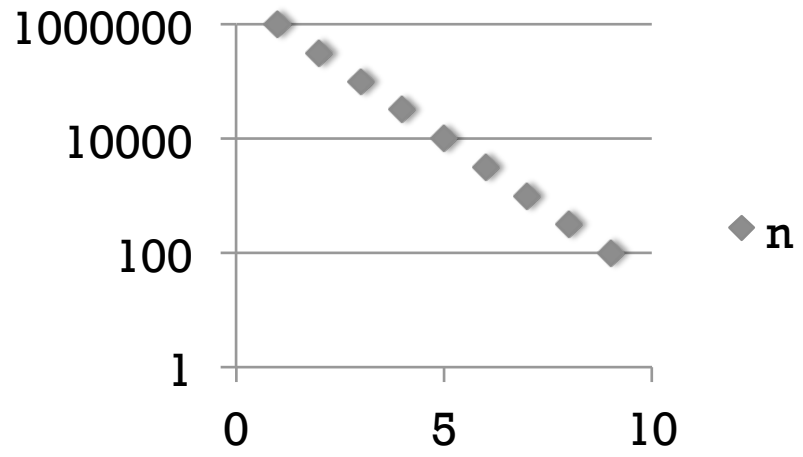
(generally, cx^d, for some $x < 1$)

A *random graph* has exponential distribution

Best seen when n is on a log scale



src: Dan Suciu



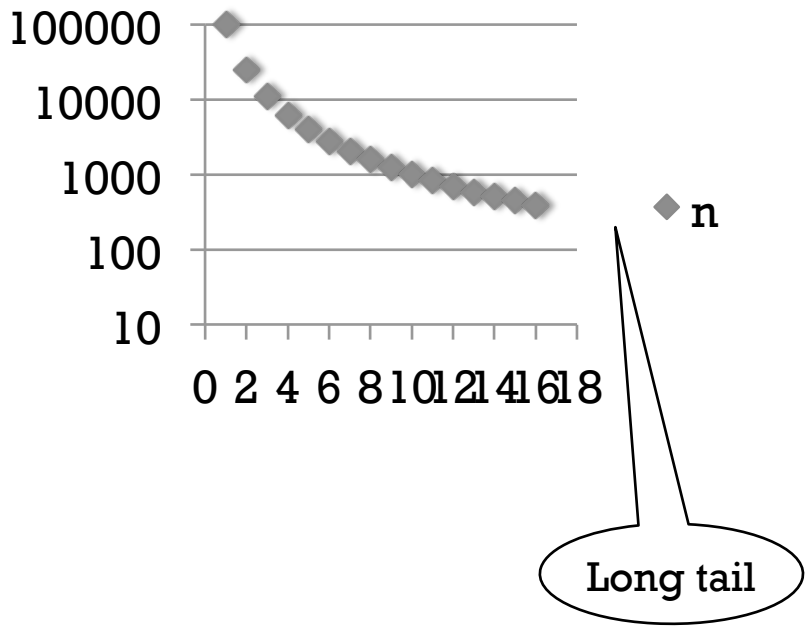
src: Dan Suciu

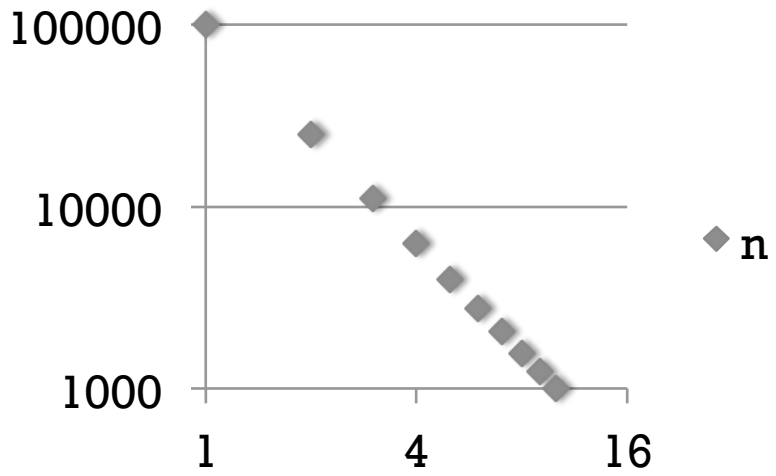
ZIPF DISTRIBUTION

$$n(d) \simeq \frac{1}{d^x}$$ **for some value x>0**

Human-generated data has Zipf distribution: letters in alphabet, words in vocabulary, etc.

Best seen in a log-log scale





src: Dan Suciu

A HISTOGRAM OF THE WEB

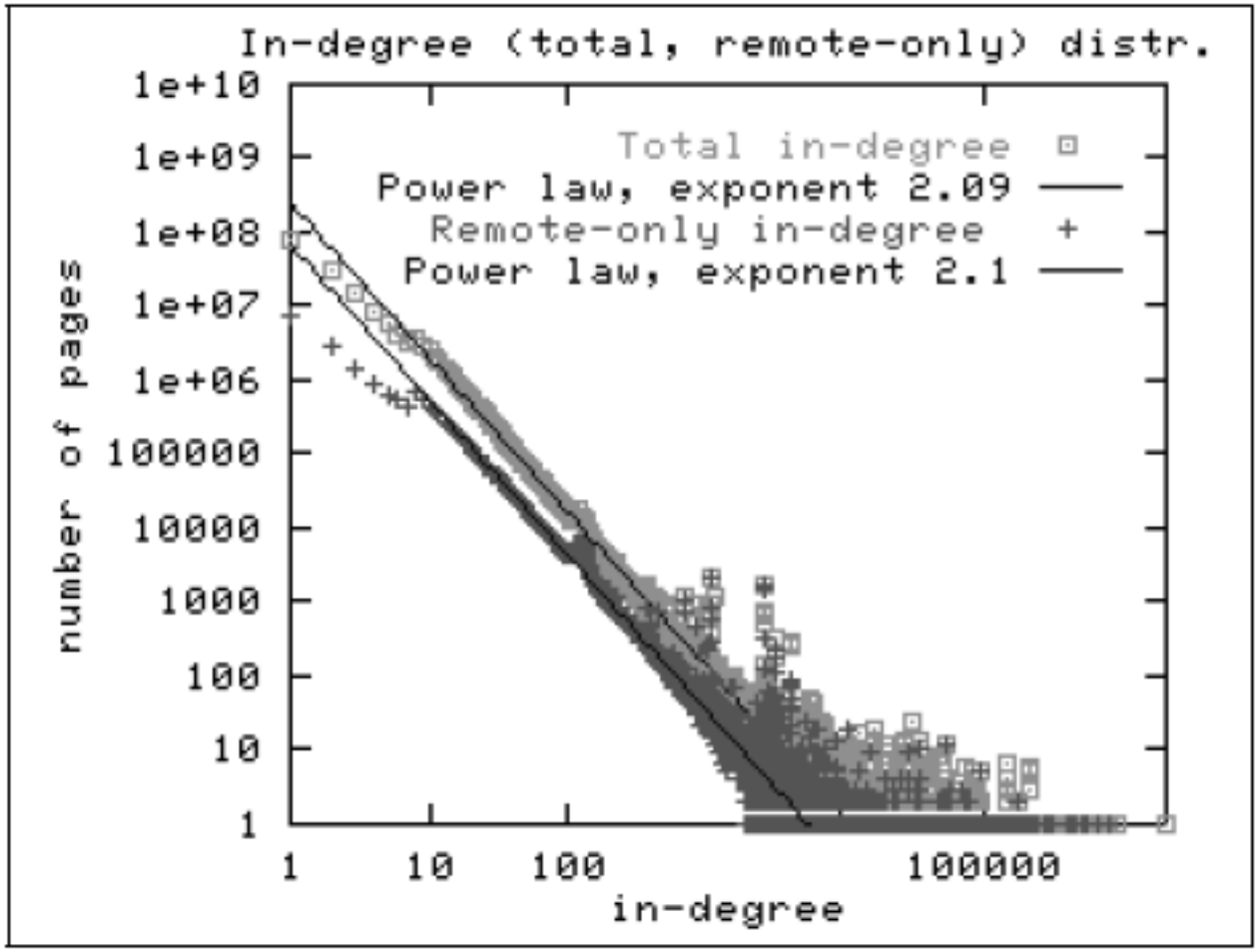


Figure 2: In-degree distribution.

Late 1990's
200M Webpages

Exponential ?

Zipf ?

Broder et al. 2000

THE BOWTIE STRUCTURE OF THE WEB

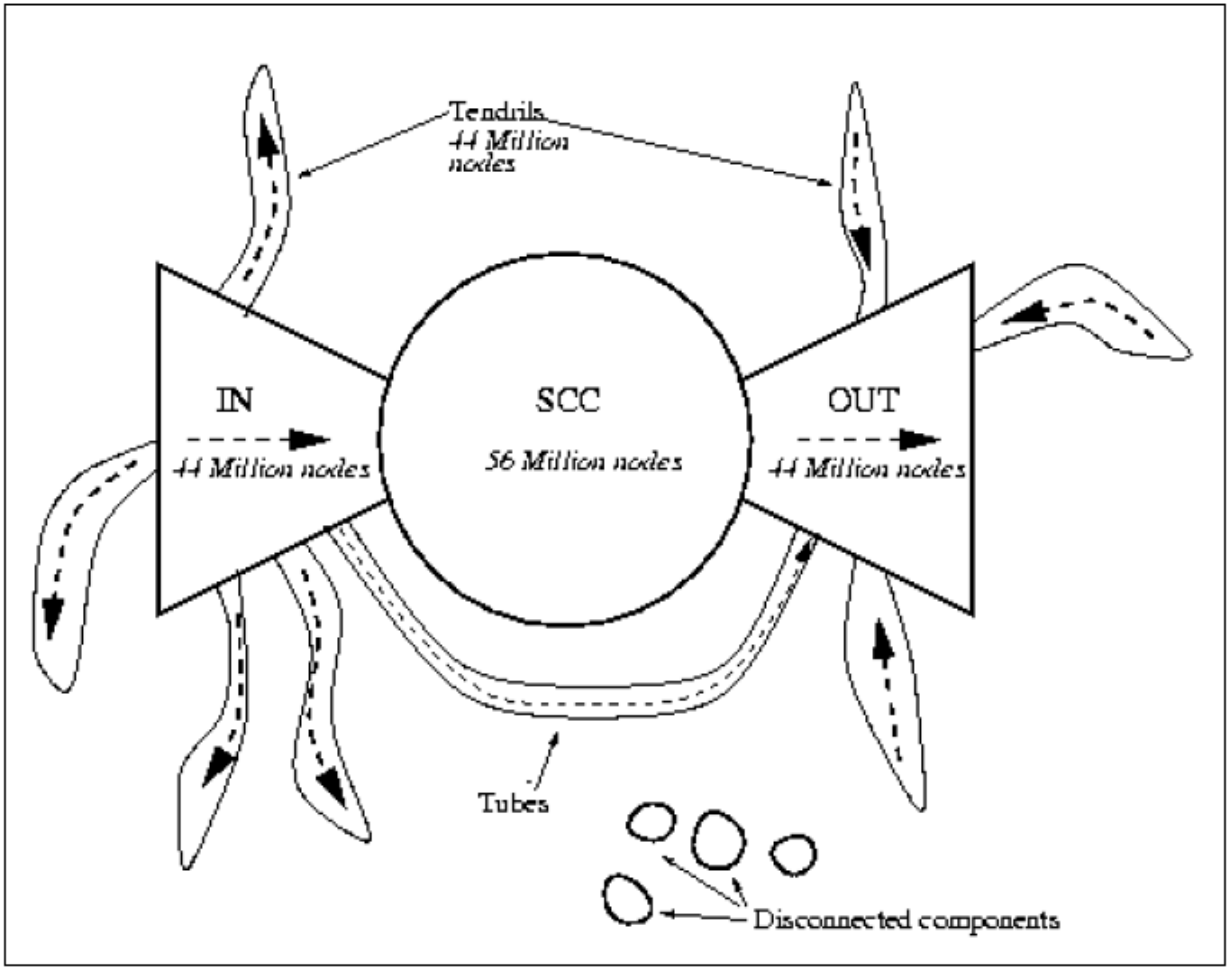


Figure 4: The web as a bowtie. SCC is a giant strongly connected component. IN consists of pages with paths to SCC, but no path from SCC. OUT consists of pages with paths from SCC, but no path to SCC. TENDRILS consists of pages that cannot surf to SCC, and which cannot be reached by surfing from SCC.

Broder et al. 2000

WHERE WE ARE

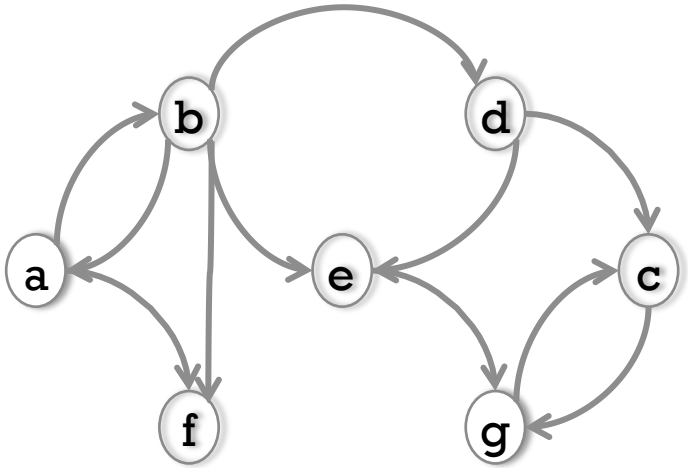
- **Graphs**
  - What is a graph?
  - Graph Structures
  - Graph Analytics
- **Traversals**
- **Patterns**
- **Recursion**
- **Graph Representations**

GRAPH ANALYTICS: SOME STRUCTURAL TASKS

Diameter

- Longest of all shortest paths

What is the diameter of this graph?

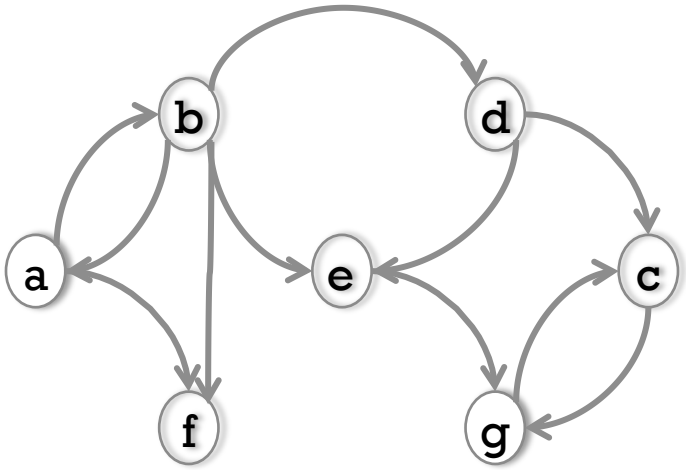


Analyzing Analytics, Bordawekar 2012

GRAPH ANALYTICS: SOME STRUCTURAL TASKS

Connectivity Coefficient

- Minimum number of vertices you need to remove that will disconnect the graph



What is the connectivity coefficient of this graph?

May depend on what is meant by "connectivity"

x and y are strongly connected if
*x is reachable from y **and** y is reachable from x*
x and y are connected if
*x is reachable from y **or** y is reachable from x*

Why might you want to compute the connectivity coefficient?

Analyzing Analytics, Bordawekar 2012

GRAPH ANALYTICS: SOME STRUCTURAL TASKS

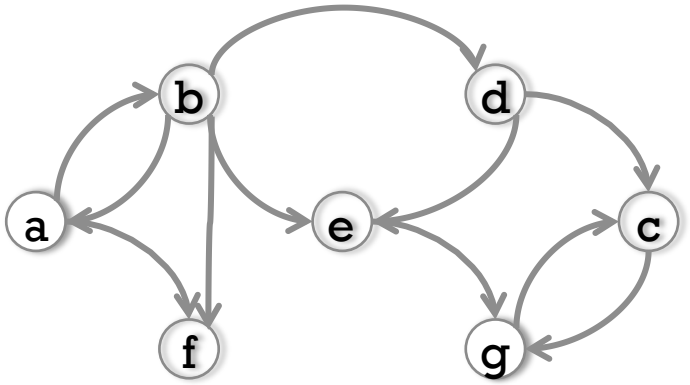
We may want to understand the "importance" of a vertex

Various notions of *Centrality*

- Closeness Centrality of a vertex:
  - Average length of all its shortest paths
- Betweeness Centrality of a vertex *v*:
  - the fraction of all shortest paths that pass through *v*

What is the betweeness centrality of vertex *e*?

Why might you want to compute betweenness centrality?



Analyzing Analytics, Bordawekar 2012

GRAPH ANALYTICS: SOME STRUCTURAL TASKS

Degree Centrality

- Degree centrality of *v*: *degree(v) / |E|*
  - "Important" nodes are those with high degree

But: say we both have 5 friends

- We have the same degree centrality
- But what if your 5 friends are Barack Obama, Larry Page, Bill Gates, the Dalai Lama, and Oprah Winfrey?
- Shouldn't you be considered more "important?"

Analyzing Analytics, Bordawekar 2012

EIGENVECTOR CENTRALITY (I.E., PAGERANK)

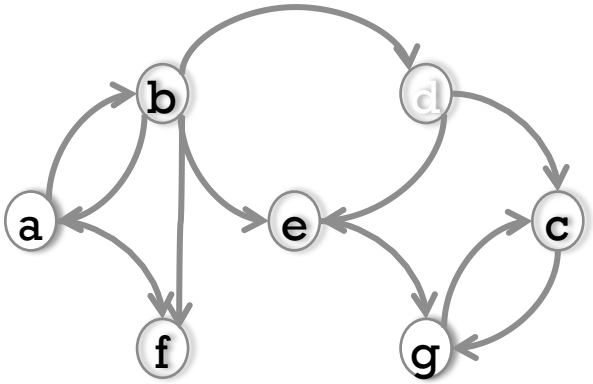
Basic idea (but oversimplified)

```
while not converged:
  for each vertex v
    rank(v) = sum of ranks from incoming edges
```

Two problems:

1) If a page with millions of outgoing links links to me, that's less valuable than a page with only a few outgoing links.

2) If I'm 27 hops away from Barack Obama, he shouldn't really influence my rank. We need a *damping factor.*



PAGERANK: 2ND ATTEMPT

```
while not converged:
```

$$PR(A) = \frac{1-d}{N} + d\left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \cdots\right).$$

ensures ranks sum to 1

- Assume edges B→A, C→A, D→A, …
- *PR(X)* is the PageRank of vertex *X*
- *L(X)* is the number of outgoing links from *X*
- *d* is the damping factor
- *N* is the number of vertices

WHERE WE ARE

- **Graphs**
  - What is a graph?
  - Graph Structures
  - Graph Analytics
- **Traversals**
- **Patterns**
- **Recursion**
- **Graph Representations**

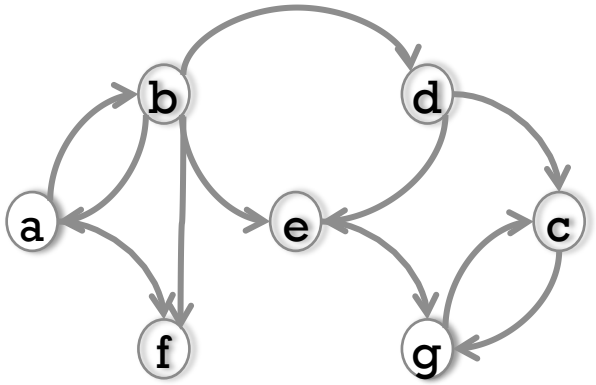
TRAVERSAL TASKS

Minimum Spanning Tree

- Find the smallest subset of edges that (weakly) connect the graph

What is a minimum spanning tree of this graph?

Why might you want to compute this?



TRAVERSAL TASKS: EULER'S BRIDGES OF KONIGSBERG

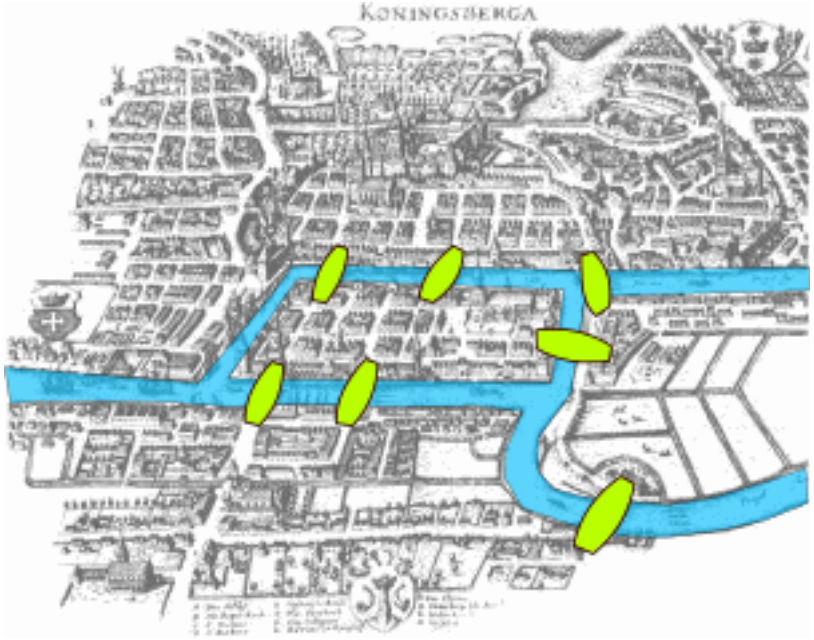
Can we visit every vertex and cross each bridge only once?

Observation: if you enter by a bridge, you must leave by a bridge

Result: If you don't care where you start and end, then at most two vertices can have an odd degree.

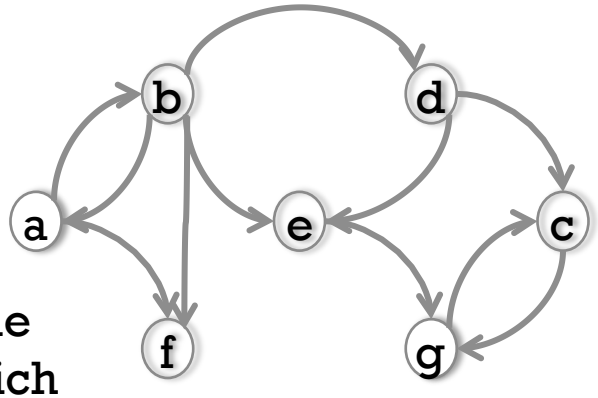
Result: If you want to start and end in the same place, every vertex must have an even degree.

Very easy to check!



TRAVERSAL TASKS: HAMILTONIAN PATHS

Can we create a path that visits every vertex only once?



A lot harder! Why?

Intuition: Each vertex is involved in a lot of possible paths. But we can only "use" the vertex once. Which is the "best" way to use the vertex?

Extension: Assume there is a cost to traversing each edge. Can we find the path that visits every vertex with the minimum cost?

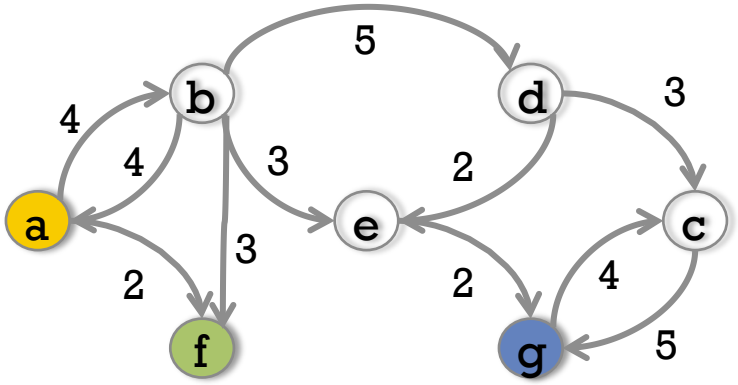
No efficient algorithm can exist. Heuristics and approximations are the best we can do.

TRAVERSAL TASKS

Maximum Flow

- Input: A graph with labeled edges indicating the "capacity" of that edge, and special vertices called sources and sinks
- Find a subgraph that maximizes flow between sources and sinks

Condition: for each vertex, incoming flow must equal outgoing flow



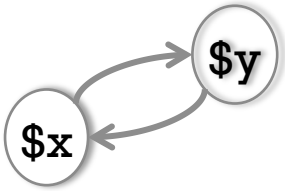
WHERE WE ARE

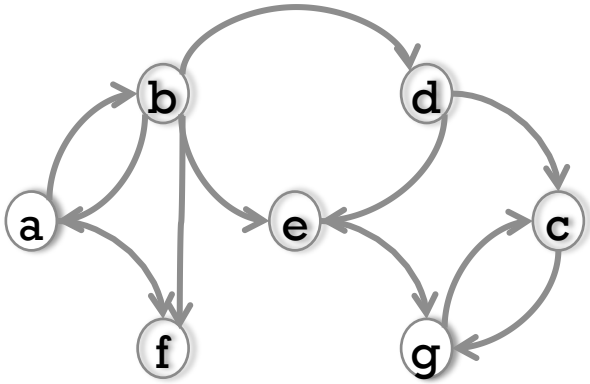
- **Graphs**
- **Traversals**
- **Patterns**
  - Pattern Matching
  - Pattern Expressions
- **Recursion**
- **Graph Representations**

PATTERN-MATCHING TASKS

Find all instances of a pattern

May involve vertex or edge labels





POPULAR PATTERN-MATCHING TASK: TRIANGLES

Find Triangles a → b → c → a

The total number of triangles is another measure of connectedness.

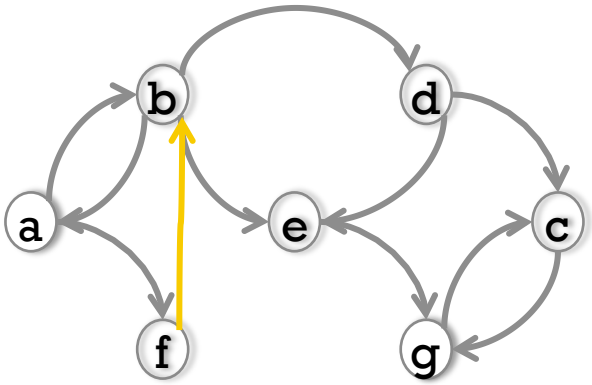
Lots of algorithms; a popular challenge problem for computer scientists

Utility in practice is not so clear (to me)

Key idea to remember: a naïve algorithm will find the same triangle multiple times.

a → b → c → a
b → c → a → b
c → a → b → c

You can extend this to k-cycles, etc.

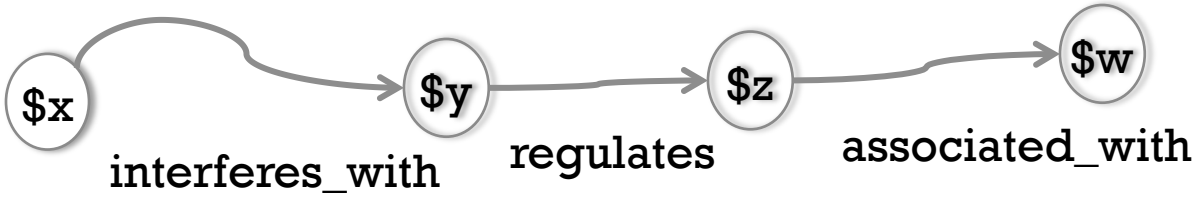


ANOTHER PATTERN MATCH EXAMPLE

Given a graph with edge labels

- Drug X interferes with Drug Y
- Drug Y regulates the expression of gene Z
- gene Z is associated with disease w

Find drugs that interfere with another drug involved in the treatment of a disease



NEED SOME KIND OF PATTERN EXPRESSION LANGUAGE

SPARQL

SQL-like language over a fixed schema:
subject, predicate, object

```
SELECT ?x WHERE
?x interferes_with ?y .
?y regulates ?z .
?z associated_with?w .
```

subject	predicate	object
terazine	interferes_with	betamin
betamin	regulates	GT1234
GT1234	associated_with	cancer
doratin	regulates	XY6789

RDF = "Resource Description Framework"
Defines a formal data model for (subject, predicate, object) "triples."
A triple is just a labeled edge in a graph.

NEED SOME KIND OF PATTERN EXPRESSION LANGUAGE

Datalog

Assume a relation R(subject, predicate, object)

```
Ans(x) :-
    R(x, interferes_with, y),
    R(y, regulates, z),
    R(z, associated_with, w)
```

NEED SOME KIND OF PATTERN EXPRESSION LANGUAGE

SQL

Assume a relation R(subject, predicate, object)

```
SELECT i.subject
FROM R i, R r, R a
WHERE i.predicate = "interferes_with"
AND r.predicate = "regulates"
AND a.predicate = "associated_with"
AND i.object = r.subject
AND r.object = a.subject
```

WHERE WE ARE

- **Graphs**
- **Traversals**
- **Patterns**
  - Pattern Matching
  - Pattern Expressions
- **Recursion**
- **Graph Representations**

NEED SOME KIND OF PATTERN EXPRESSION LANGUAGE

Relational Algebra

Assume a relation R(subject, predicate, object)

$$\sigma_{p=\text{interferes\_with}}(R) \bowtie_{o=s} \sigma_{p=\text{regulates}}(R) \bowtie_{o=s} \sigma_{p=\text{associated\_with}}(R)$$

NEED SOME KIND OF PATTERN EXPRESSION LANGUAGE

Datalog, 2nd formulation

Assume relations

InterferesWith(drug1, drug2),
Regulates(drug, gene),
AssociatedWith(gene, disease)

```
Ans(x) :-
   InterferesWith(x, y),
   Regulates(y, z),
   AssociatedWith(z, w)=
```

NEED SOME KIND OF PATTERN EXPRESSION LANGUAGE

SQL, 2nd formulation

Assume relations

InterferesWith(drug1, drug2),
Regulates(drug, gene),
AssociatedWith(gene, disease)

```
SELECT i.drug1
FROM InterferesWith i, Regulates r, AssociatedWith a
WHERE i.drug2 = r.drug
AND r.gene = a.gene
```

ASIDE ON SPARQL

Designed for graph query

Not algebraically closed

- Input is a graph, but output is a set of records

Limited expressiveness

- Support for path expressions since `v1.1`
- But not arbitrary recursion

If your input is tabular, you have to shred it into a graph before using SPARQL

- Often a 3x-6x blow up in size!

ASIDE: PRISM-LIKE EXAMPLE IN DATALOG

"In October, a foreign national named Mike Fikri purchased a one-way plane ticket from Cairo to Miami, where he rented a condo."

BoughtFlight('Fikri', 'Cairo', 'Miami', 'oneway' 4/1/2013)

"Over the previous few weeks, he'd made a number of large withdrawals from a Russian bank account and placed repeated calls to a few people in Syria."

Withdrawal('Fikri', 5000, 'some bank', 4/4/2013)
Withdrawal('Fikri', 2000, 'some bank', 4/5/2013)
…

"More recently, he rented a truck, drove to Orlando, and visited Walt Disney World by himself."

Rented('Fikri', 'truck', 'Miami', 'Orlando', 4/13/2013)

http://www.businessweek.com/magazine/palantir-the-vanguard-of-cyberterror-security-11222011.html

PRISM EXAMPLE MODELED IN DATALOG

```
flag(person, 1, date) :- BoughtFlight(person, origin, destination, oneway, date),
                         FlaggedAirports(origin),
                           USAirports(destination),
                           oneway='oneway'

ForeignWithdrawal(person, sum(amount), date) :-
                                   Withdrawal(person, amount, bank, date),
                                   ForeignBank(bank), amount > 1000

flag(person, 1, date) :- ForeignWithdrawal(person, totamount, date), totamount>10k

flag(person, 1, date) :- Rented(person, vehicle, origin, dest, date),
ImportantLocation(dest)
totalflags(person, sum(flag), min(date), max(date) :- flag(person, flag, date)

alert(person, flagcnt, mindate, maxdate) :-
                                   totalflags(person, flagcnt, mindate, maxdate),
                                   maxdate – mindate < 10 days, flagcnt > 3
```

http://www.businessweek.com/magazine/palantir-the-vanguard-of-cyberterror-security-11222011.html

ANOTHER DATALOG EXAMPLE

Who contacted who, when? We don't care how.

```
contacted(Person1, Person2, Time) :-
  email(Person1, Person2, Message, Time, …)

contacted(Person1, Person2, Time) :-
  called(Time, Voicemail, Person1, Person2, …)

contacted(Person1, Person2, Time) :-
  text_message(Time, Message, Person1, Person2, …),
```

ANOTHER DATALOG EXAMPLE

Who could have known before June 3 that X was going to happen?

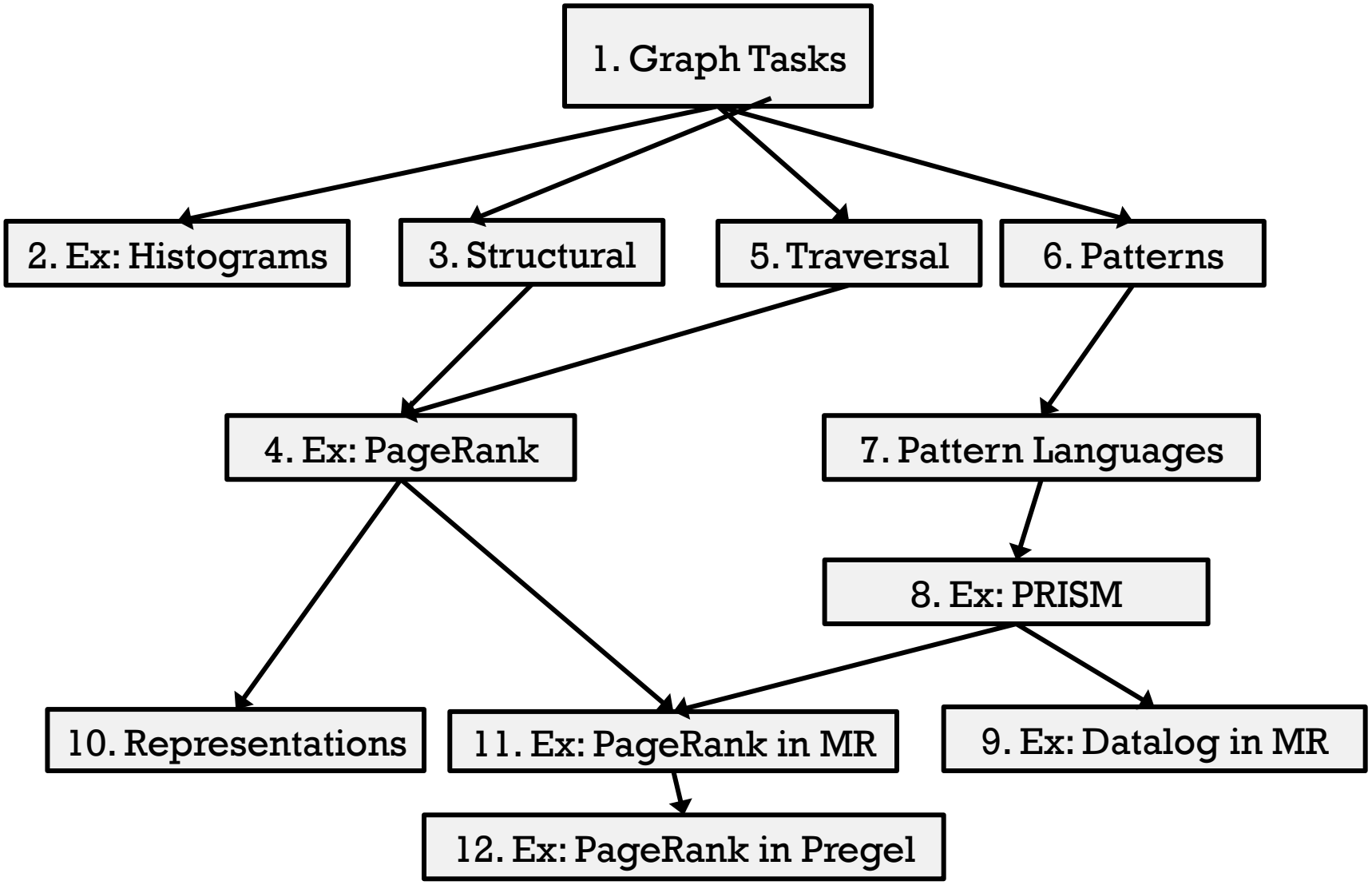
```
knew("Sam").

knew(Person2) :-
   knew(Person1)
   email(Person1, Person2, Message, Time),
   Time < "June 3".

knew(Person2) :-
   knew(Person1),
   met_with(Person1, Person2, Time),
   Time < "June 3".
```

self-reference!

WHERE WE ARE


1. Graph Tasks
2. Ex: Histograms
3. Structural
5. Traversal
6. Patterns
4. Ex: PageRank
7. Pattern Languages
8. Ex: PRISM
10. Representations
11. Ex: PageRank in MR
9. Ex: Datalog in MR
12. Ex: PageRank in Pregel

WHERE WE ARE

- **Graphs**
- **Traversals**
- **Patterns**
- **Recursion**
  - What's recursion?
  - Recursive Graphs
- **Graph Representations**

EVALUATION OF RECURSIVE PROGRAMS

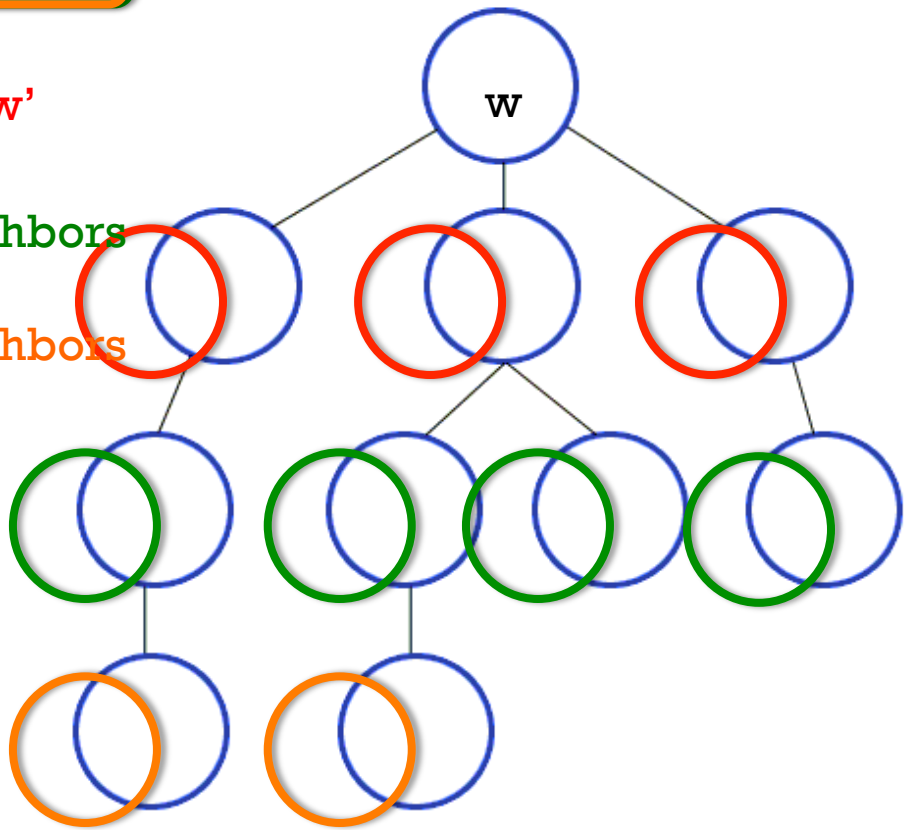
Reachability from a given node

```
A(y) :- Edge('w', y)
A(y) :- A(x), Edge(x,y)
```

step 0: A0 = all nodes connected to 'w'

step 1: for each node in A0, find neighbors

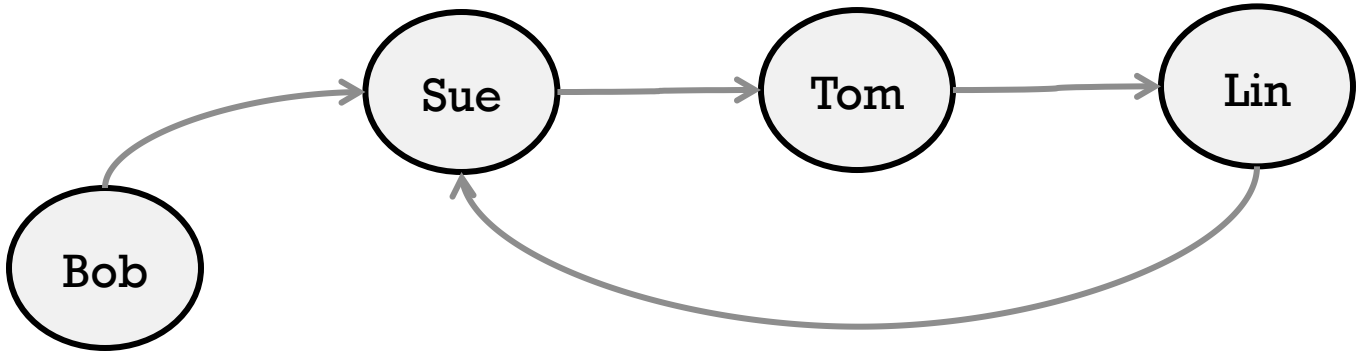
step 2: for each node in A1, find neighbors



EVALUATING RECURSIVE QUERIES

Problems with this approach

- What if there are cycles in the graph?



EXAMPLE OF SEMI-NAÏVE EVALUATION

$$\Delta A^0 = R', i = 1$$

while ΔA^{i-1} is not empty:

$$A^i = (\Delta A^0 \cup \cdots \cup \Delta A^{i-1})$$

$$\Delta A^i = (\Delta A^{i-1} \bowtie R) - A^i$$

$$i = i + 1$$

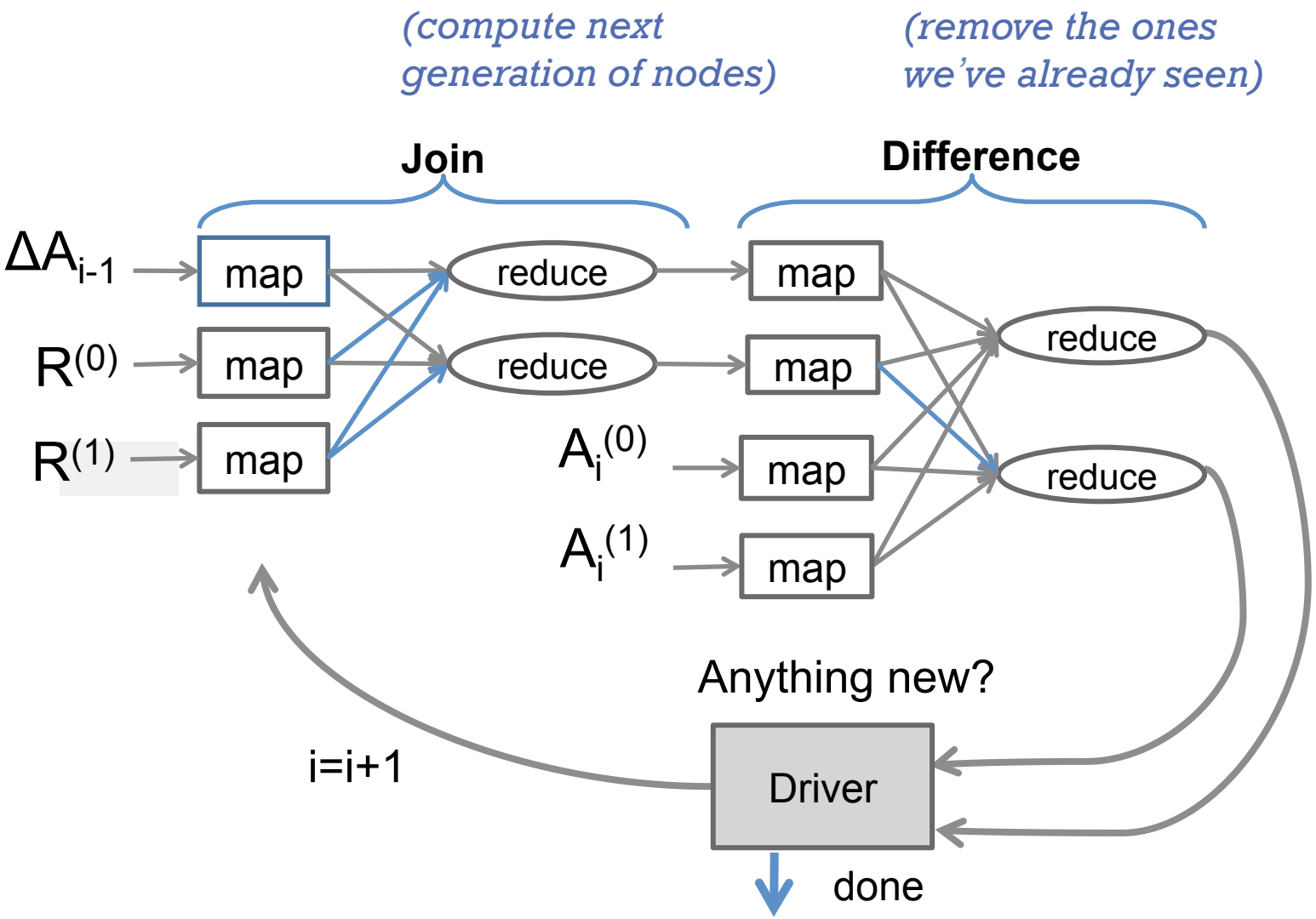
Reachability from 'a' in datalog

```
A(y)  :- R('a', y)
A(y)  :- A(x), R(x,y)
```

Join

Dupe-elim

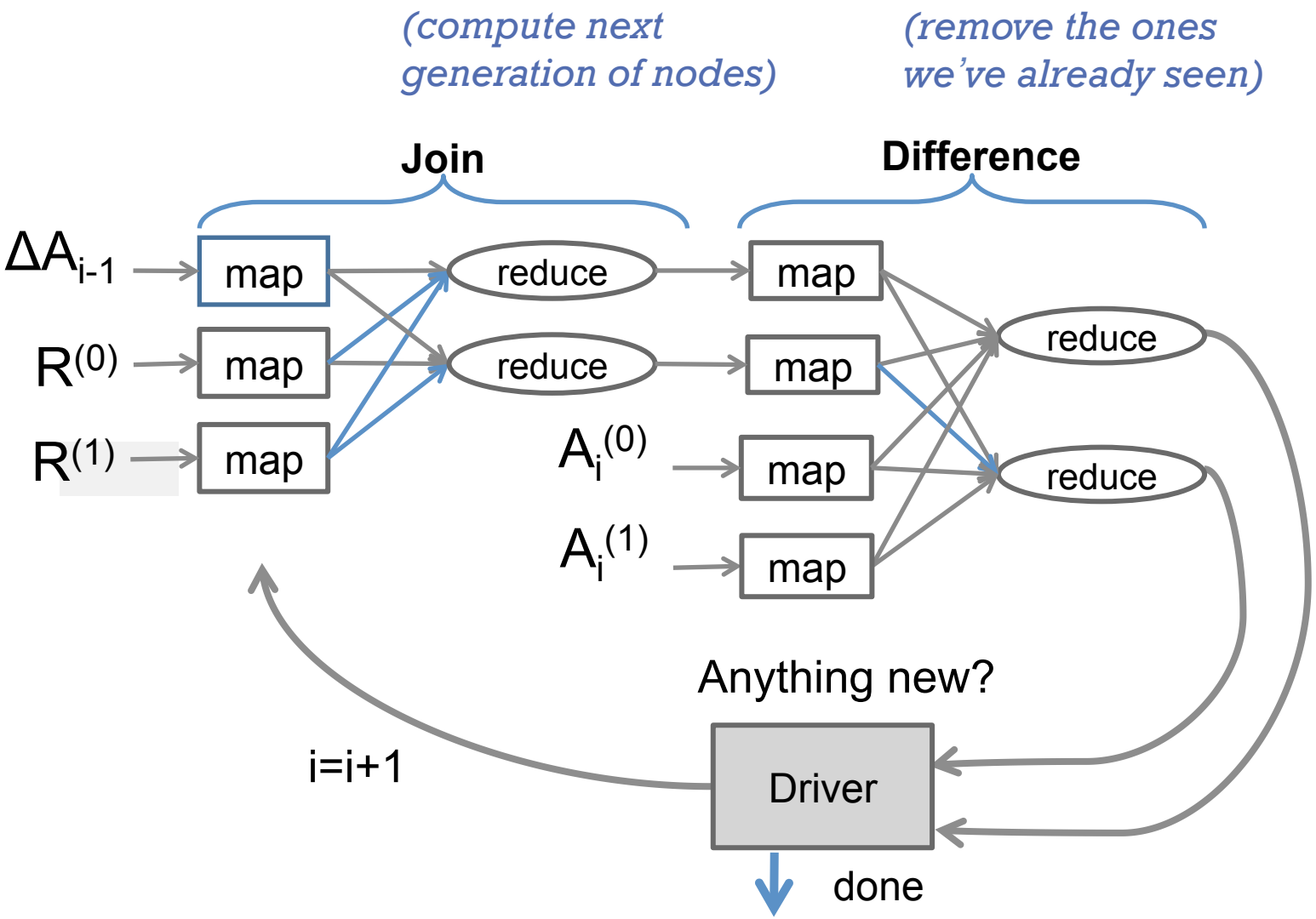
IN MAPREDUCE



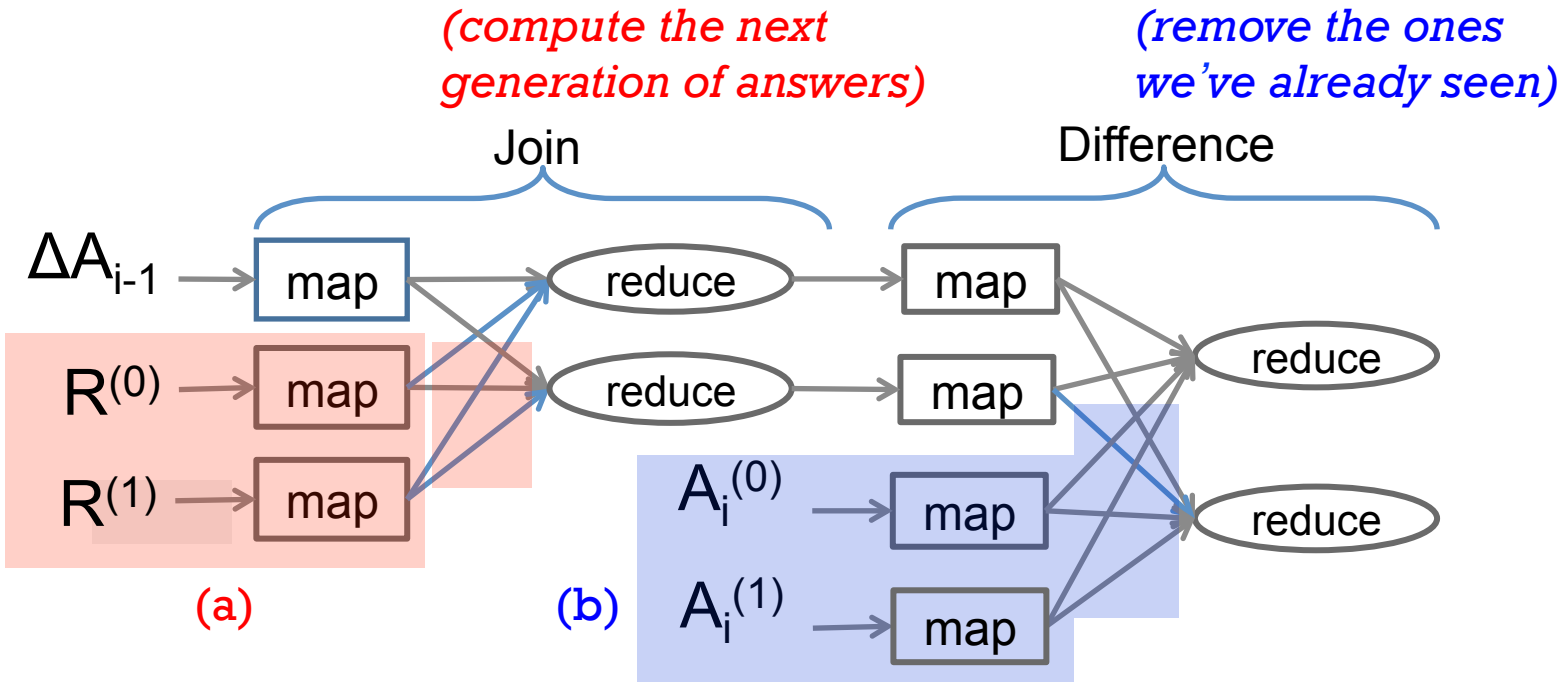
WHERE WE ARE

- **Graphs**
- **Traversals**
- **Patterns**
- **Recursion**
  - What's recursion?
  - Recursive Graphs
- **Graph Representations**

IN MAPREDUCE



EVALUATING RECURSIVE QUERIES AT SCALE



(a) R is loop invariant, but gets loaded and shuffled on each iteration

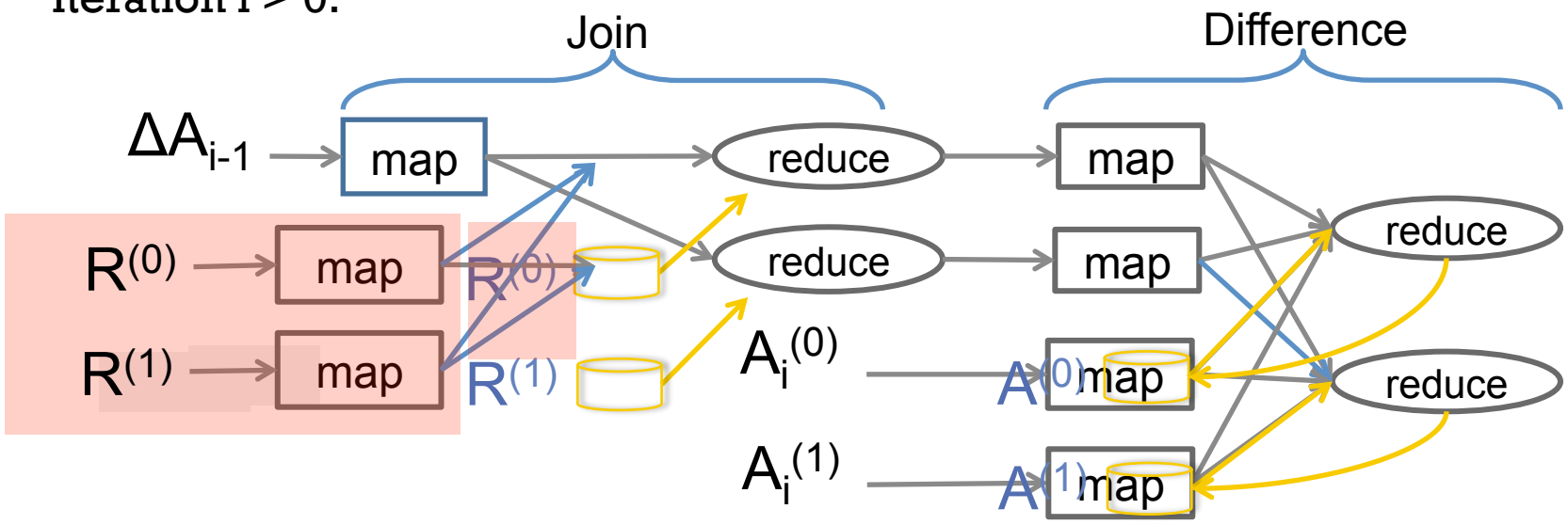
(b) A_i grows slowly and monotonically, but is loaded and shuffled on each iteration.

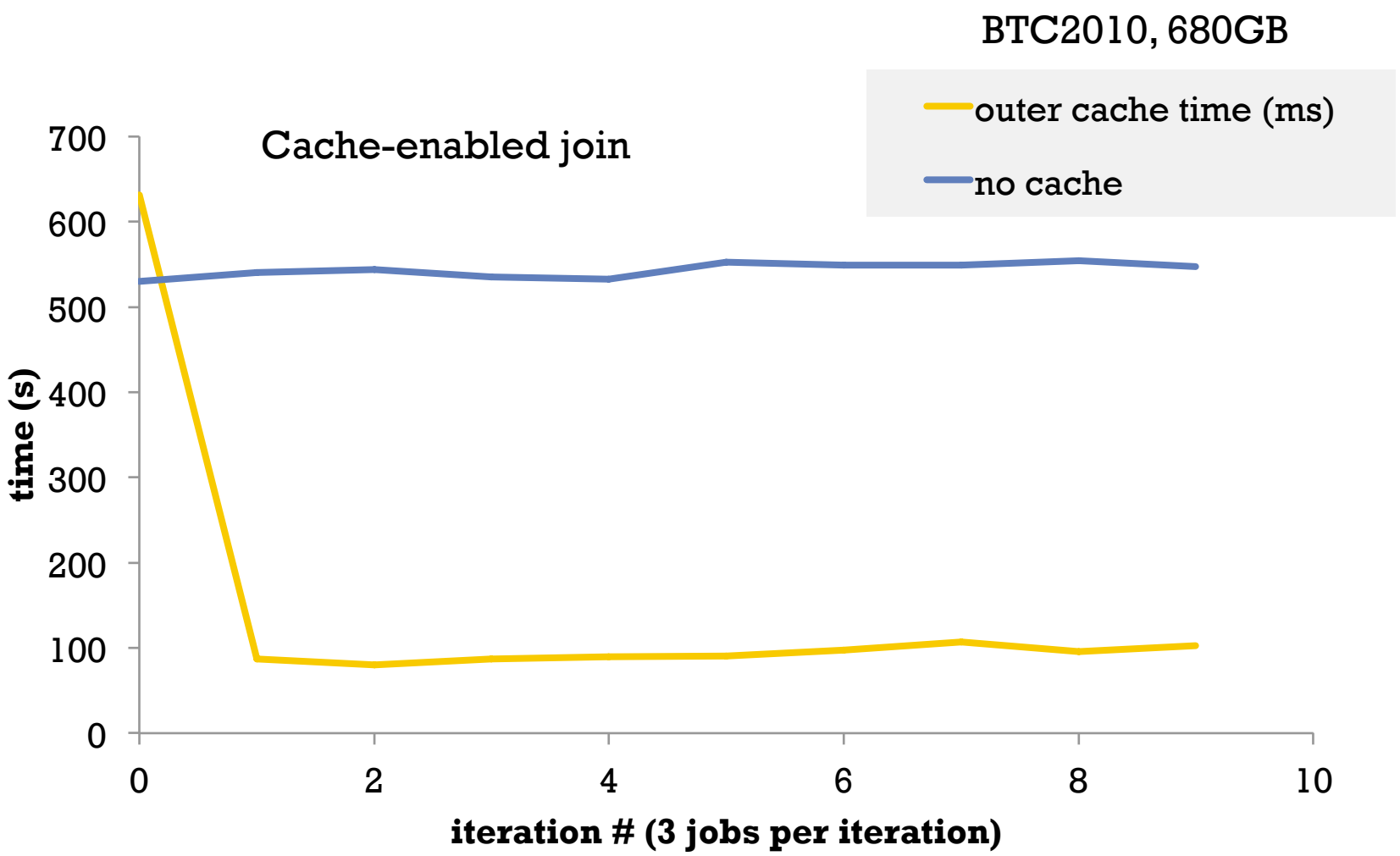
Bu et al. VLDB10, VLDBJ12
Shaw et al. Datalog12

IDEA: CACHE LOOP-INVARIANT DATA

Iteration i = 0: Load a distributed cache

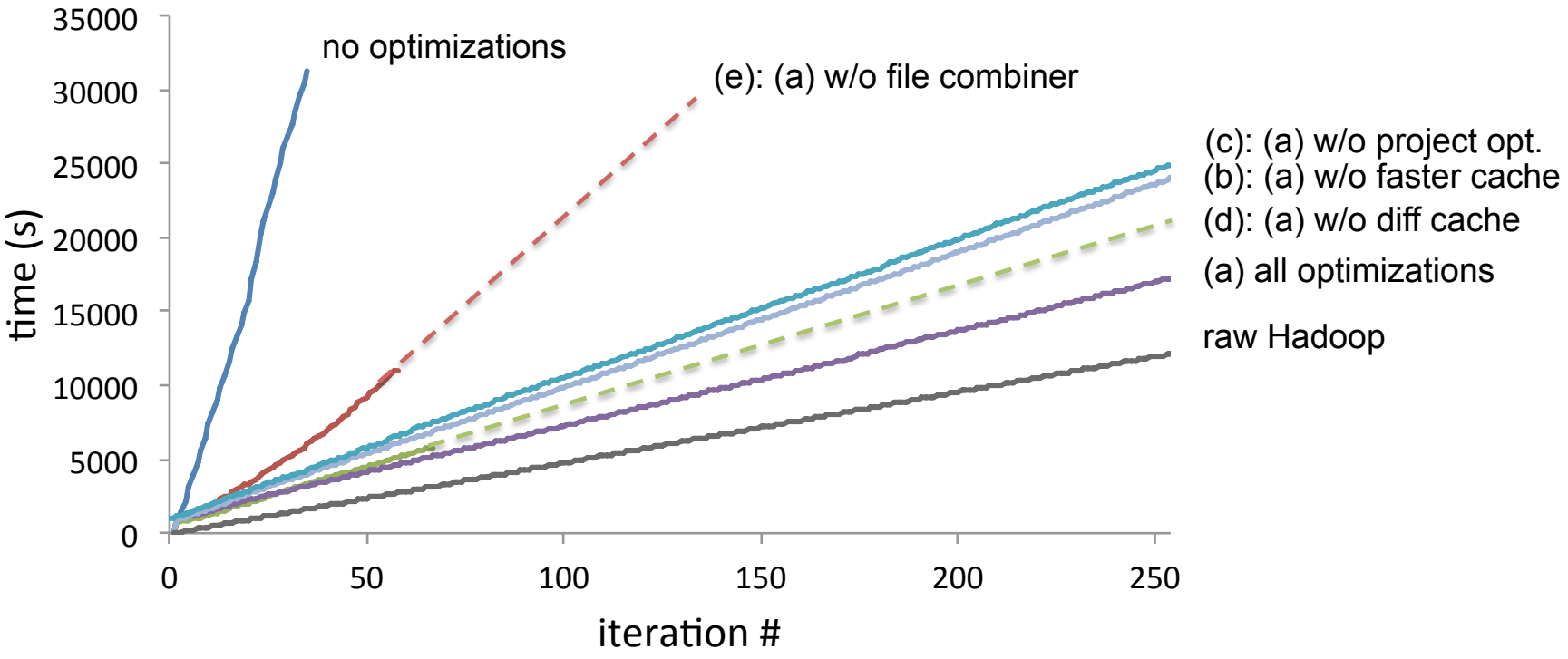
Iteration i > 0:




BTC2010, 680GB
Cache-enabled join
outer cache time (ms)
no cache
time (s)
700
600
500
400
300
200
100
0
0
2
4
6
8
10
iteration # (3 jobs per iteration)

EFFECT OF VARIOUS OPTIMIZATIONS FOR A RECURSIVE GRAPH QUERY ON BTC 2010

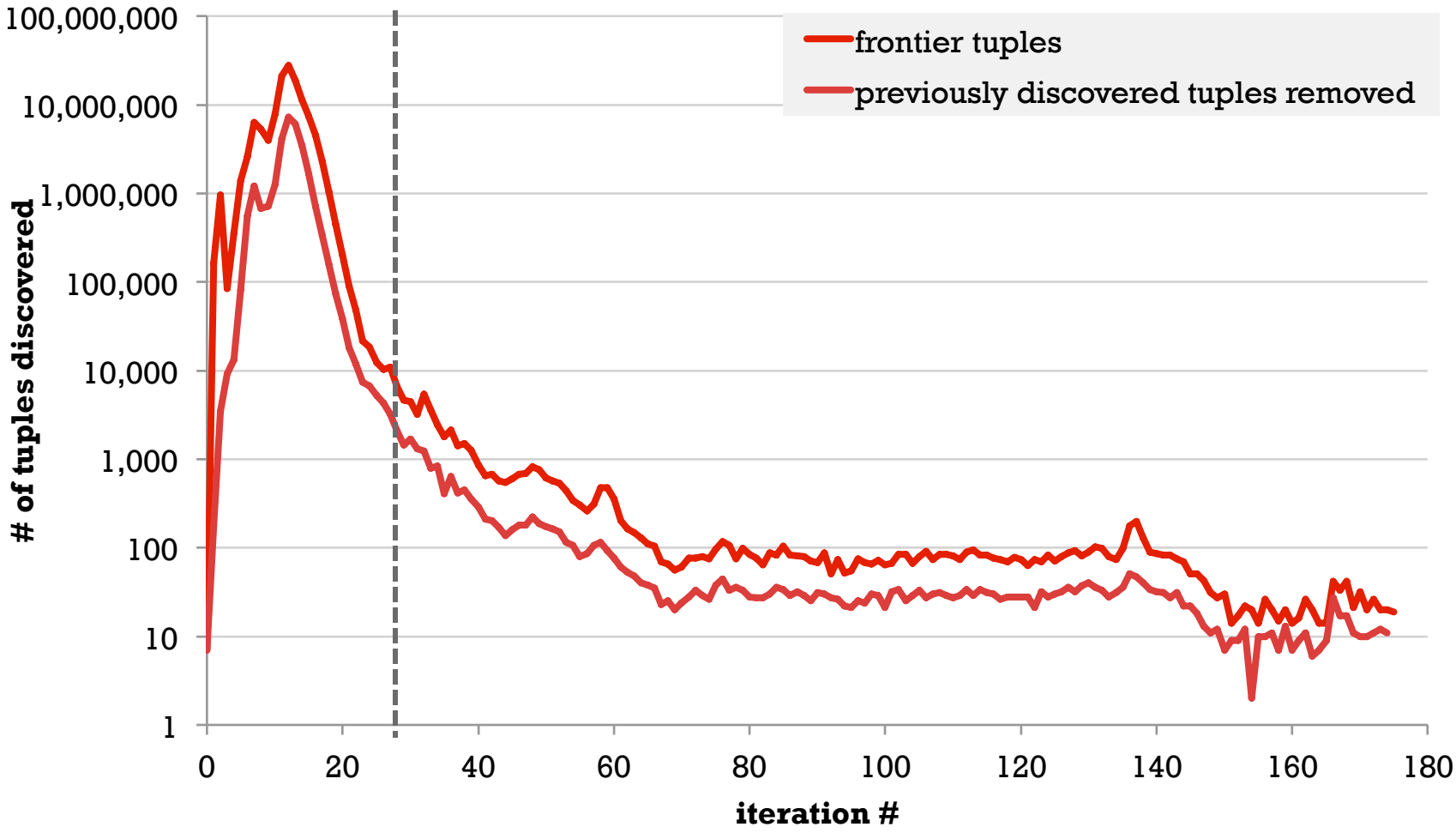
(query: transitive reachability from 7 nodes)



Takeaways:

- 10x improvement over no optimizations.
- All optimizations are useful
- We're approaching the raw overhead of Hadoop (bottom gray line)

NEW TUPLES DISCOVERED BY ITERATION NUMBER

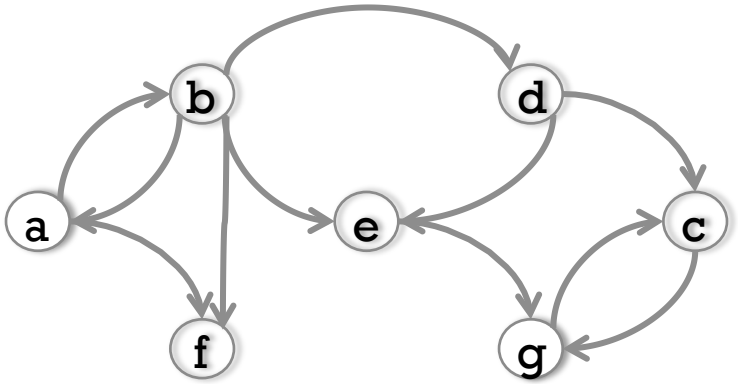


WHERE WE ARE

- **Graphs**
- **Traversals**
- **Patterns**
- **Recursion**
- **Graph Representations**
  - Examples of Graph Representations
  - Graphs with Big Data

GRAPH REPRESENTATIONS

Edge Table



Source	Target
a	b
b	a
a	f
b	f
b	e
b	d
d	e
d	c
e	g
g	c
c	g

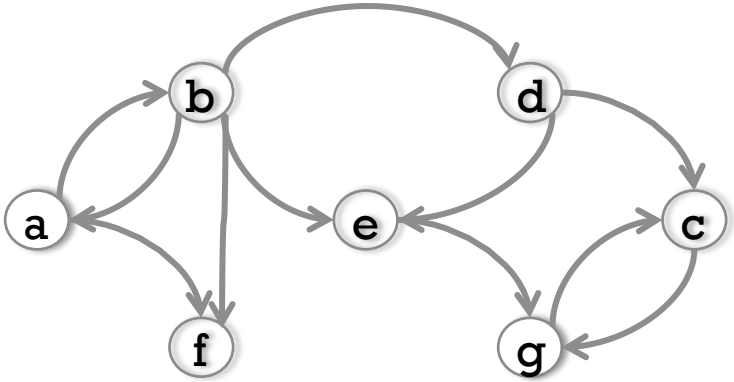
EXAMPLE USING EDGE TABLE

Find top 5 highest in-degree vertices

```
SELECT top 5 target, incount
  FROM (
    SELECT target, count(source) as incount
      FROM edges
    GROUP BY target
)
ORDER BY incount
```

GRAPH REPRESENTATIONS

Adjacency List

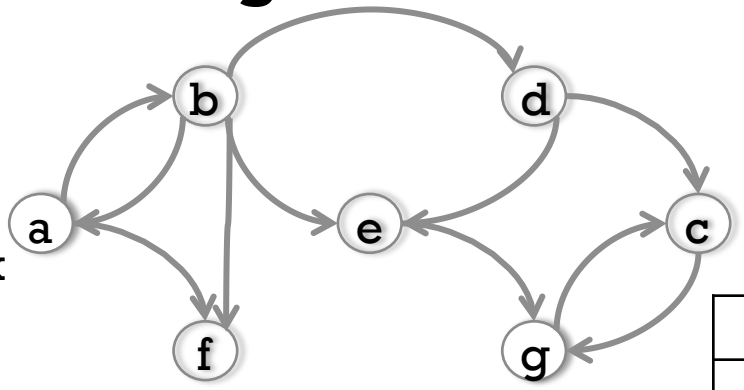


Source	Target
a	b, f
b	a, d, e, f
d	c, e
e	g
g	c
c	g

EXAMPLE USING ADJACENCY LIST (AND MAPREDUCE)

Find top 5 highest in-degree vertices

Map:
- Input key: vertex
- Input value: adjacency list
- Output key: adjacent vertex
- Output value: vertex



Reduce:
- Input key: vertex
- Input value: list of vertices
- Output key: vertex
- Output value: length of list

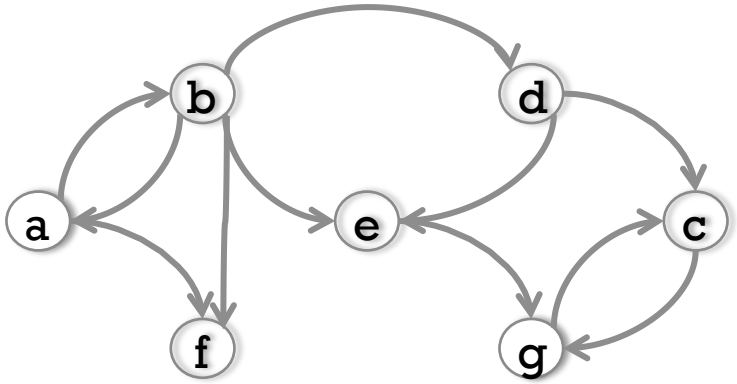
Source	Target
a	b, f
b	a, d, e
d	c, e
e	g
g	c
c	g

map

Source	Target
b	a
f	a
a	b
d	b
e	b
e	d
c	d
g	e
c	g
g	c

GRAPH REPRESENTATIONS

Adjacency Matrix



	a	b	c	d	e	f	g
a	0	1	0	0	0	1	0
b	1	0	0	1	1	1	0
c	0	0	0	1	0	0	1
d	0	0	1	0	1	0	0
e	0	0	0	0	0	0	1
f	0	0	0	0	0	0	0
g	0	0	1	0	0	0	0

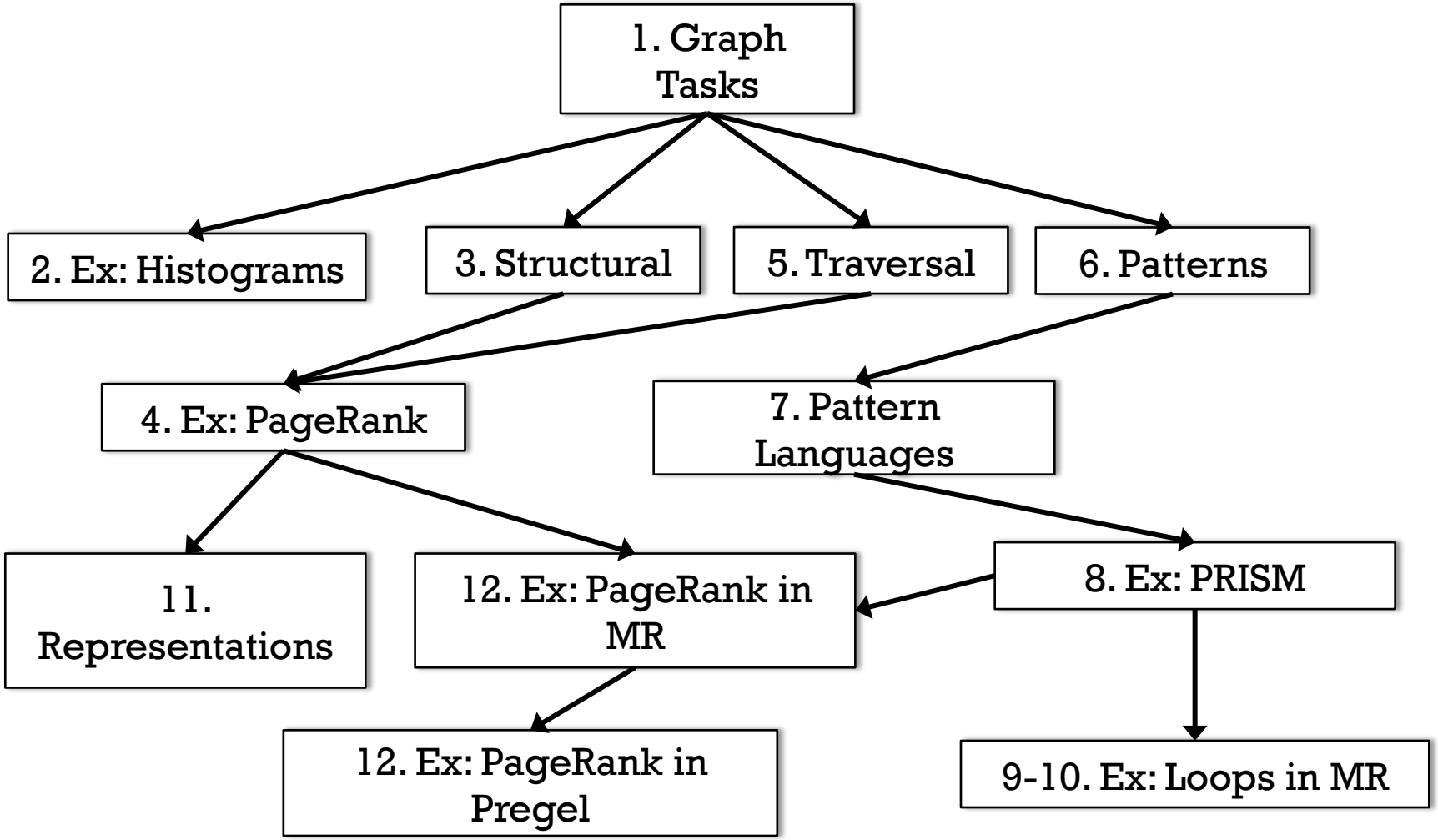
EXAMPLE USING ADJACENCY MATRIX

Find top 5 highest in-degree vertices

C = sum(A,1)
[B, IX] = sort(C)
[B, IX] = sort([5,6,4])
# B is [4,5,6]
# IX = [3,1,2]
Top = IX(1,5)

	a	b	c	d	e	f	g
a	0	1	0	0	0	1	0
b	1	0	0	1	1	1	0
c	0	0	0	1	0	0	1
d	0	0	1	0	1	0	0
e	0	0	0	0	0	0	1
f	0	0	0	0	0	0	0
g	0	0	1	0	0	0	0

WHERE WE ARE


1. Graph Tasks
2. Ex: Histograms
3. Structural
5. Traversal
6. Patterns
4. Ex: PageRank
7. Pattern Languages
11. Representations
12. Ex: PageRank in MR
8. Ex: PRISM
12. Ex: PageRank in Pregel
9-10. Ex: Loops in MR

WHERE WE ARE

- **Graphs**
- **Traversals**
- **Patterns**
- **Recursion**
- **Graph Representations**
  - Examples of Graph Representations
  - Graphs with Big Data

BIG GRAPHS

Paul Butler, Facebook, 2010

https://www.facebook.com/notes/facebook-engineering/visualizing-friendships/469716398919

Social scale

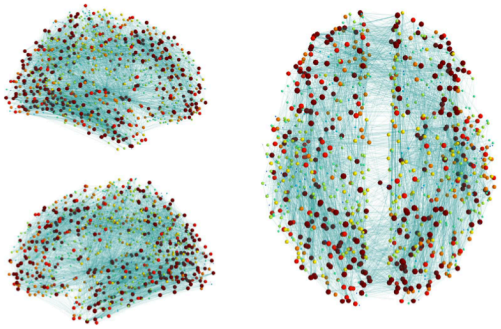
- 1 billion vertices, 100 billion edges

Web scale

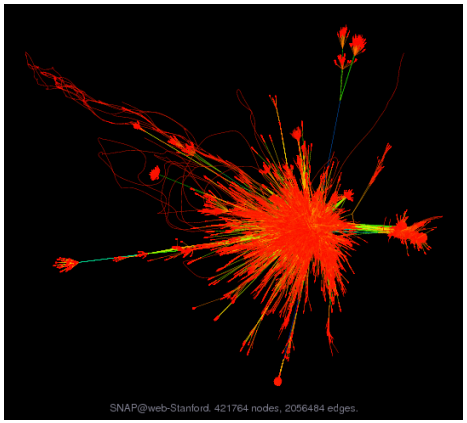
- 50 billion vertices, 1 trillion edges

Brain scale

- 100 billion vertices, 100 trillion edges



Gerhard et al, frontiers in neuroinformatics, 2011



Web graph from the SNAP database (http://snap.stanford.edu/data)

material adapted from Paul Burkhardt, Chris Waring

MAPREDUCE FOR PAGERANK

```
class Mapper
        method Map(id n, vertex N)
        p ← N.PAGERANK/|N.ADJACENCYLIST|
        EMIT(id n, vertex N)
        for all nodeid m in N.ADJACENCYLIST do
                EMIT(id m, value p)
class Reducer
        method REDUCE(id m, [p1, p2, …])
                M ← null, s ← 0
                for all p in [p1, p2, …] do
                        if ISVERTEX(p) then
                                M ← p
                        else
                                s ← s + p
                M.PAGERANK ← s * 0.85 + 0.15 / TOTALVERTICES
                EMIT(id m, vertex M)
```

PROBLEMS

The entire state of the graph is shuffled on every iteration

We only need to shuffle the new rank contributions, not the graph structure

Further, we have to control the iteration outside of MapReduce

PREGEL

Originally from Google

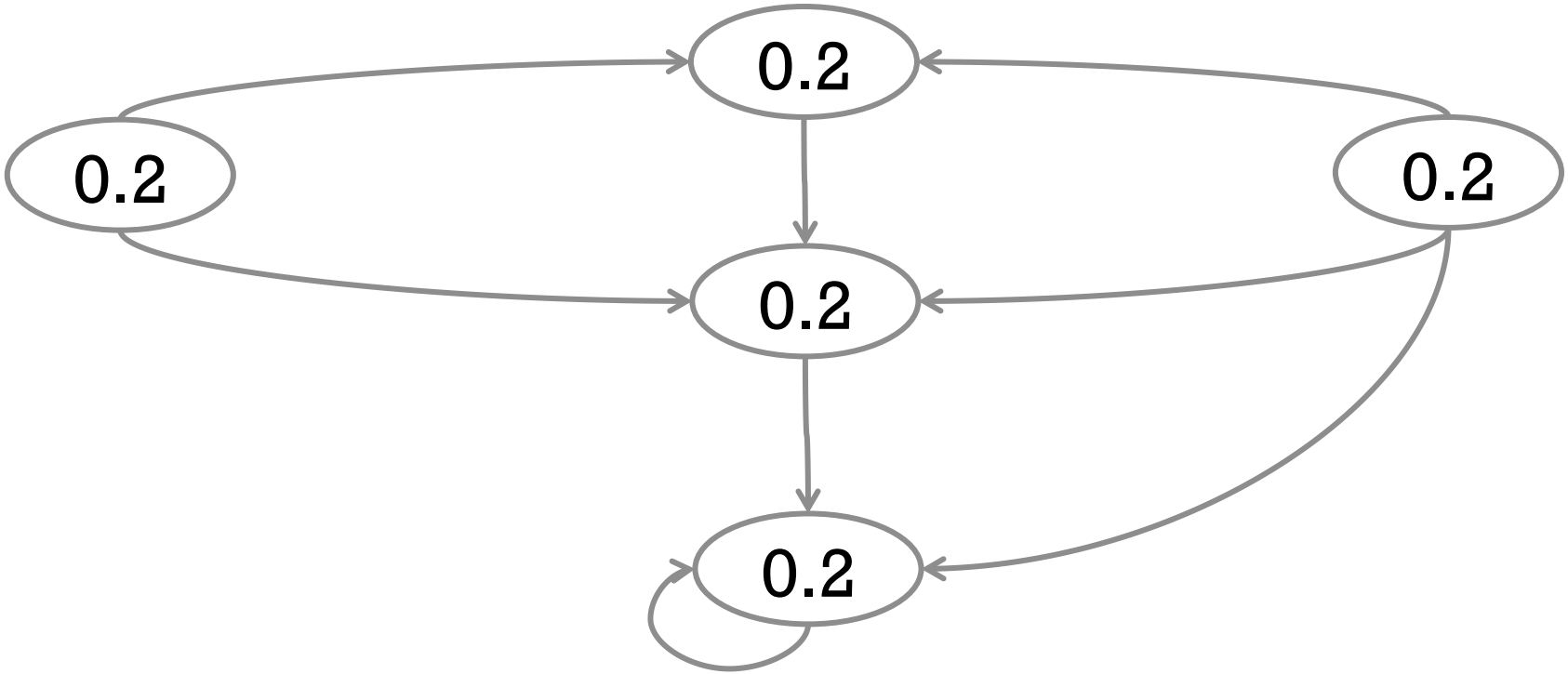
- Open source implementations
- Apache Giraph, Stanford GPS, Jpregel, Hama

Batch algorithms on large graphs

```
while any vertex is active or max iterations not reached:
    for each vertex:
        process messages from neighbors from previous
iteration                    <-- this loop is run in parallel
        send messages to neighbors
        set active flag appropriately
```

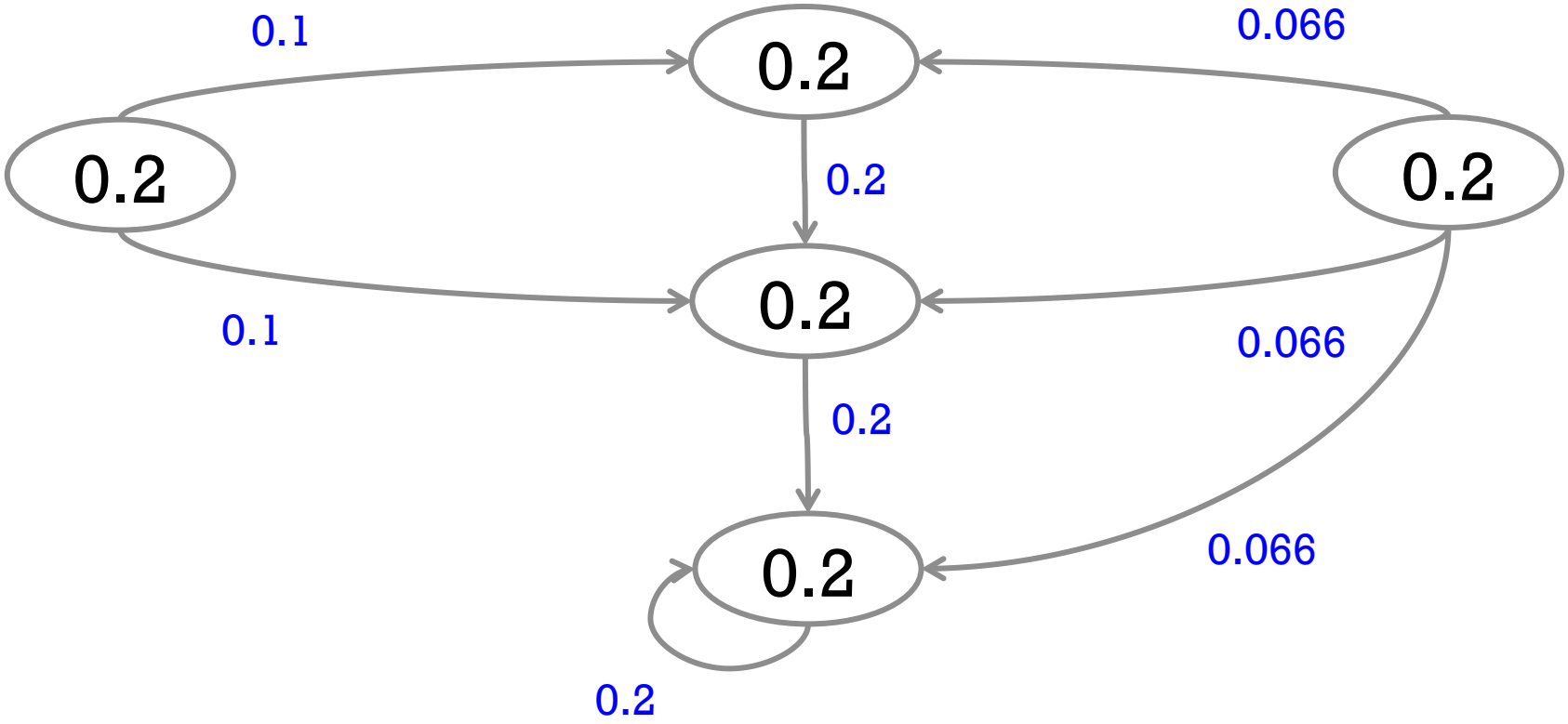
Malewicz et al. SIGMOD 10

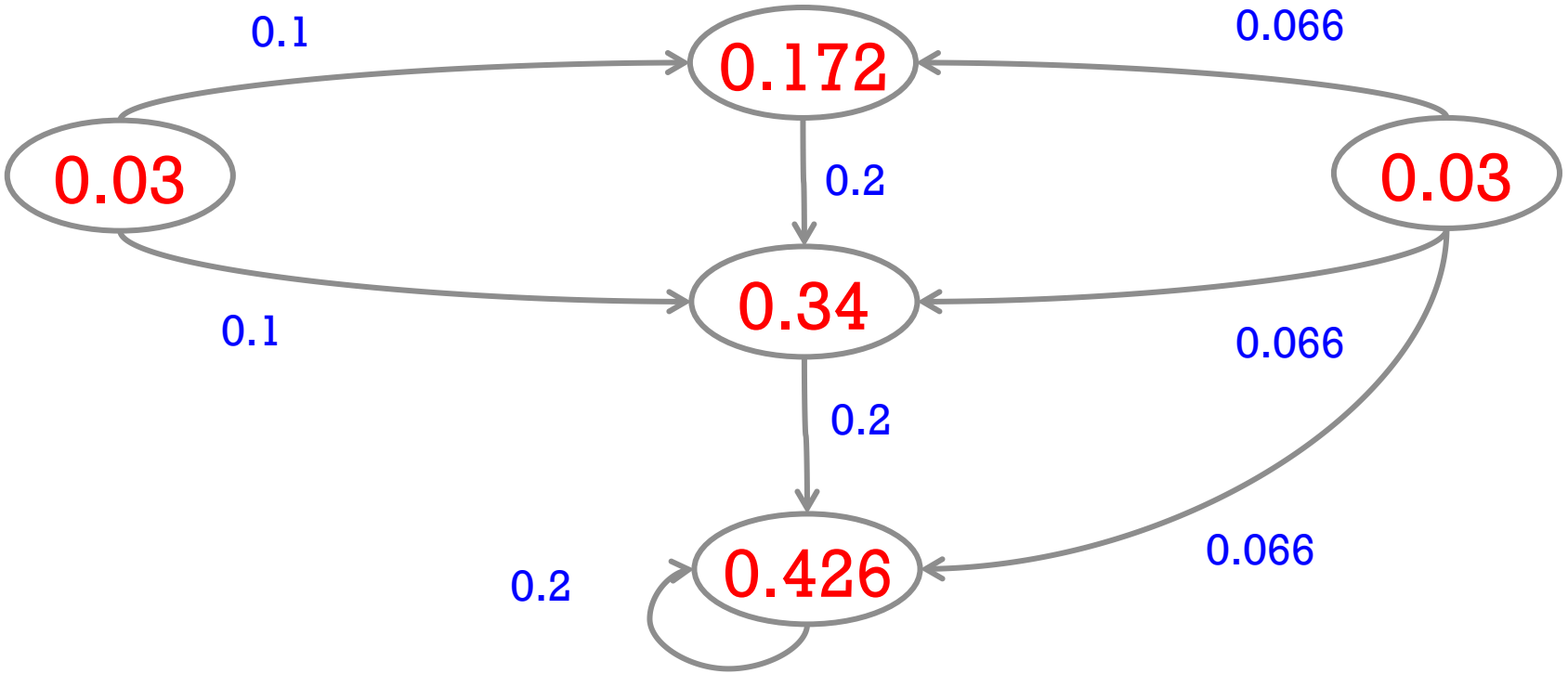
```
class PageRankVertex: public Vertex<double, void, double> {
    public:
    virtual void Compute(MessageIterator* msgs) {
        if (superstep() >= 1) {
            double sum = 0;
            for (; !msgs->Done(); msgs->Next())
                sum += msgs->Value();
            *MutableValue() = 0.15 / NumVertices() + 0.85 * sum;
        }
        if (superstep() < 30) {
            const int64 n = GetOutEdgeIterator().size();
            SendMessageToAllNeighbors(GetValue() / n);
        } else {
            VoteToHalt();
        }
    }
};
```



$$sum = sum(incoming\ values)$$

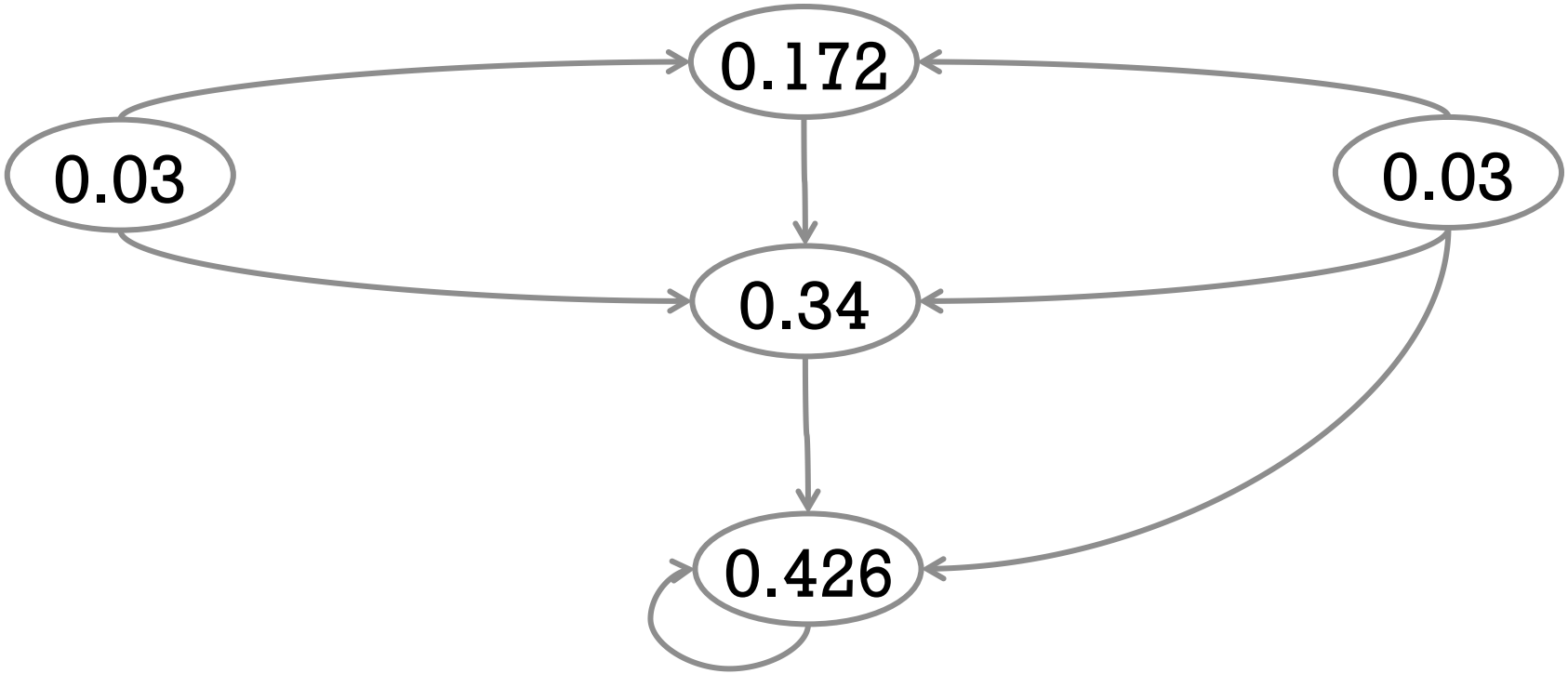
$$rank = \frac{0.15}{5 + 0.85 * sum}$$


0.1
0.2
0.066
0.2
0.2
0.2
0.1
0.066
0.2
0.2
0.066
0.2



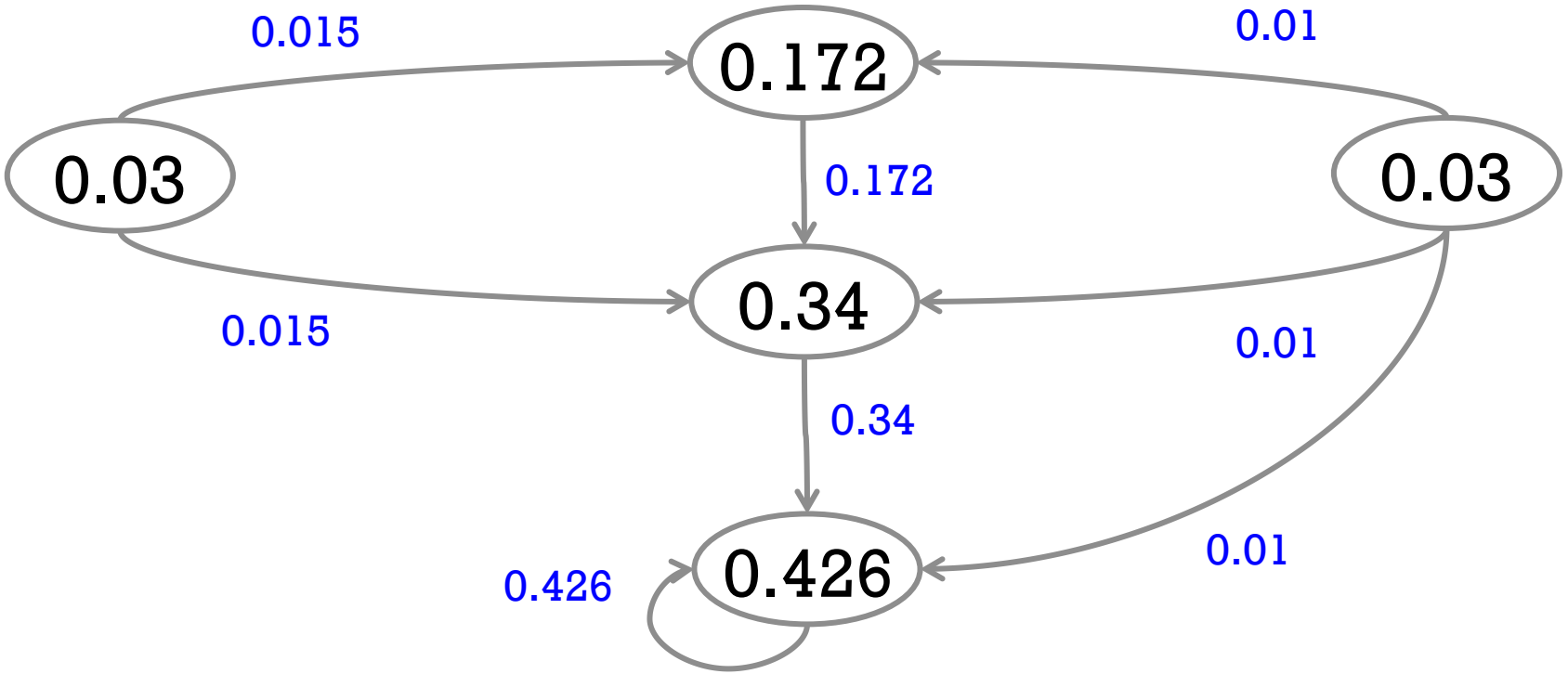
$$sum = sum(incoming\ values)$$

$$rank = \frac{0.15}{5 + 0.85 * sum}$$



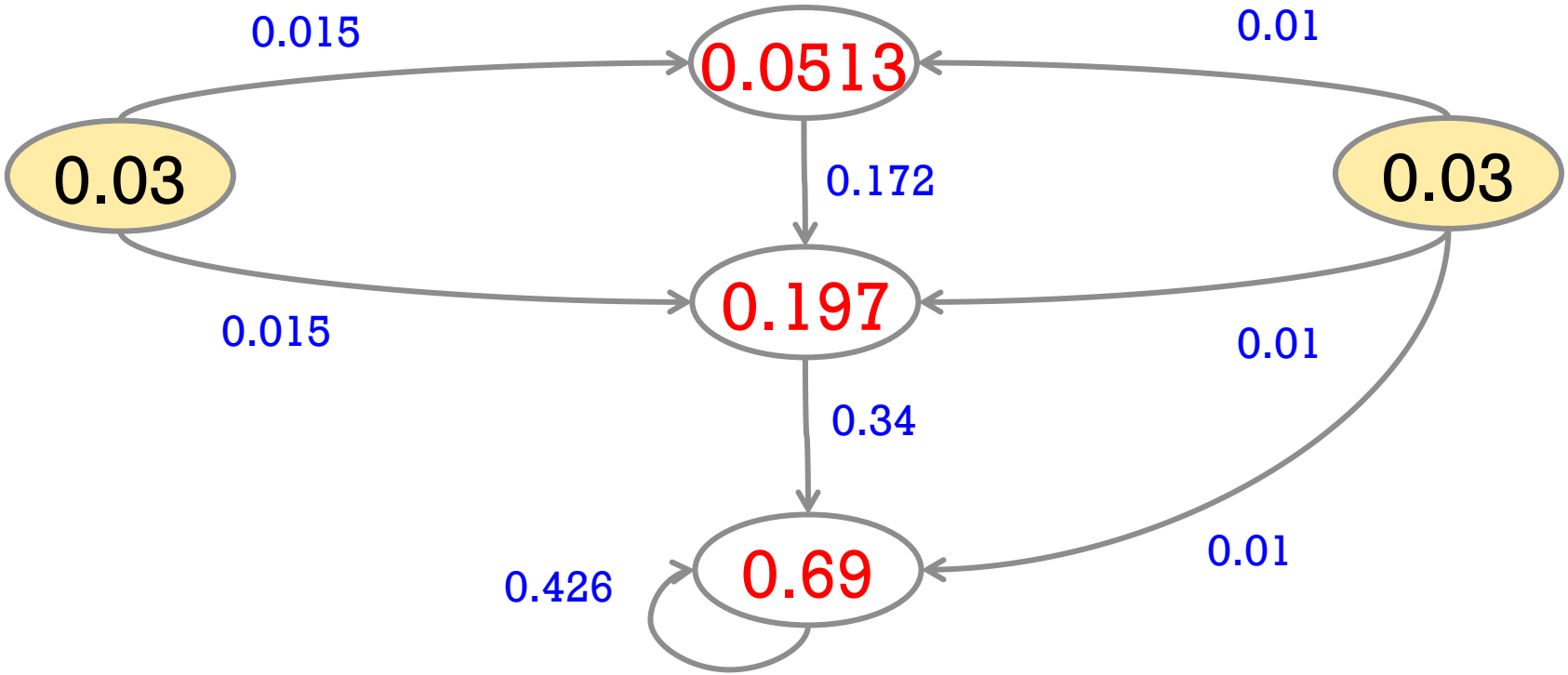
$$sum = sum(incoming\ values)$$

$$rank = \frac{0.15}{5 + 0.85 * sum}$$



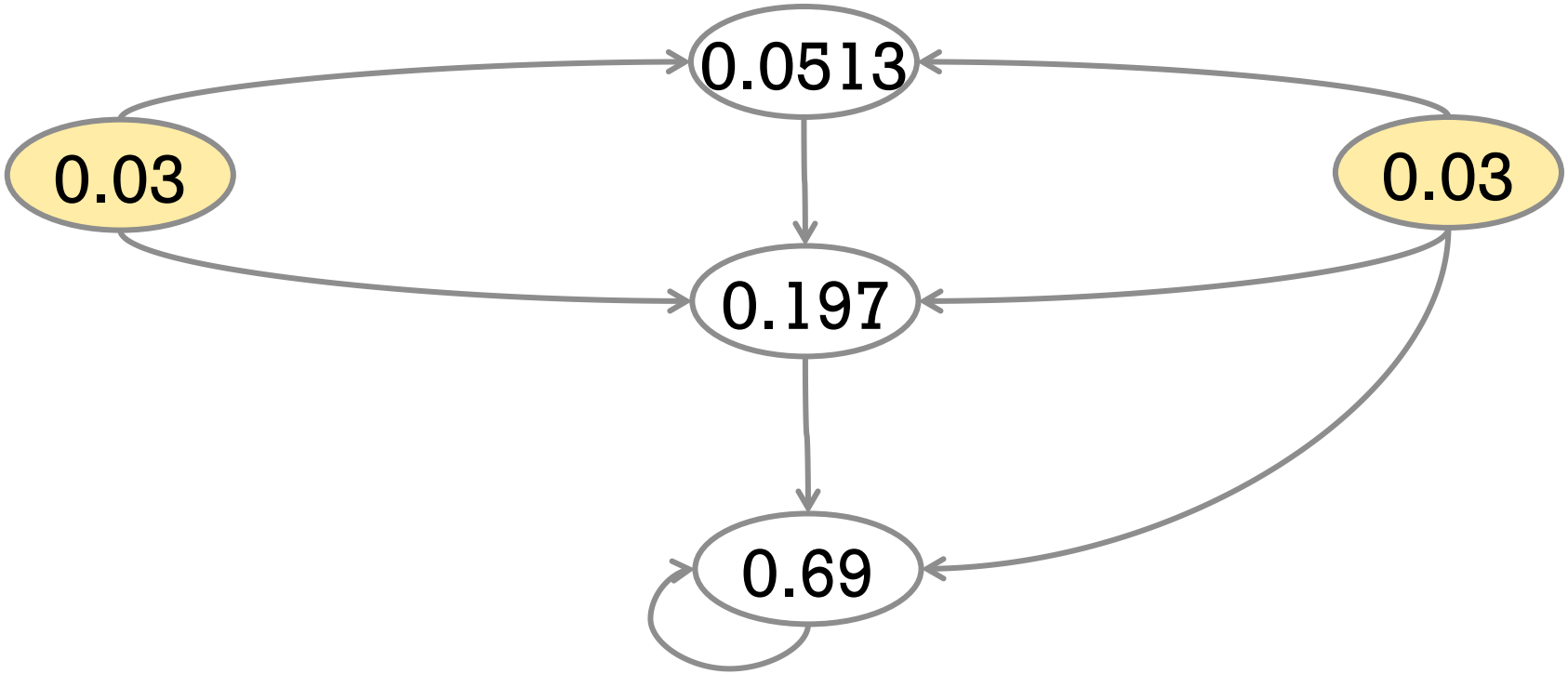
$$sum = sum(incoming\ values)$$

$$rank = \frac{0.15}{5 + 0.85 * sum}$$



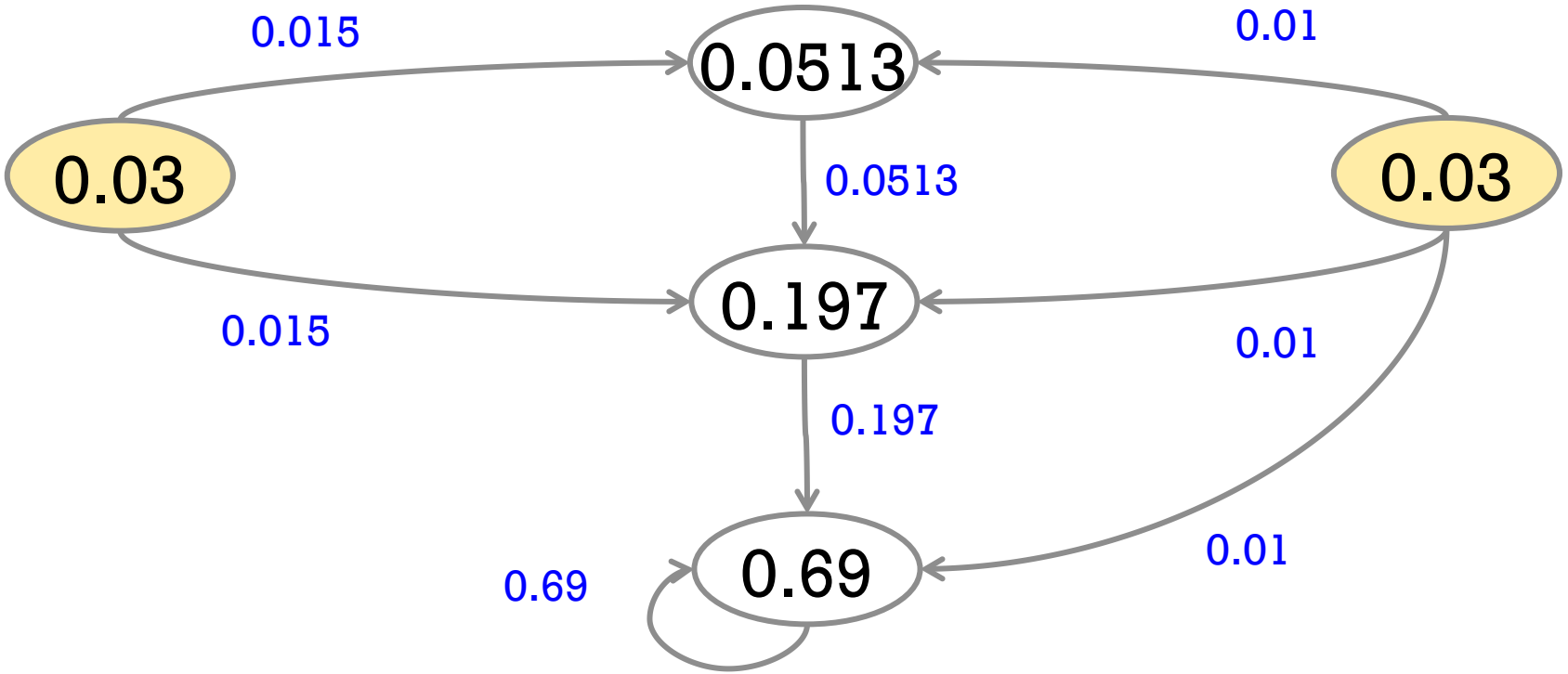
$$sum = sum(incoming\ values)$$

$$rank = \frac{0.15}{5 + 0.85 * sum}$$



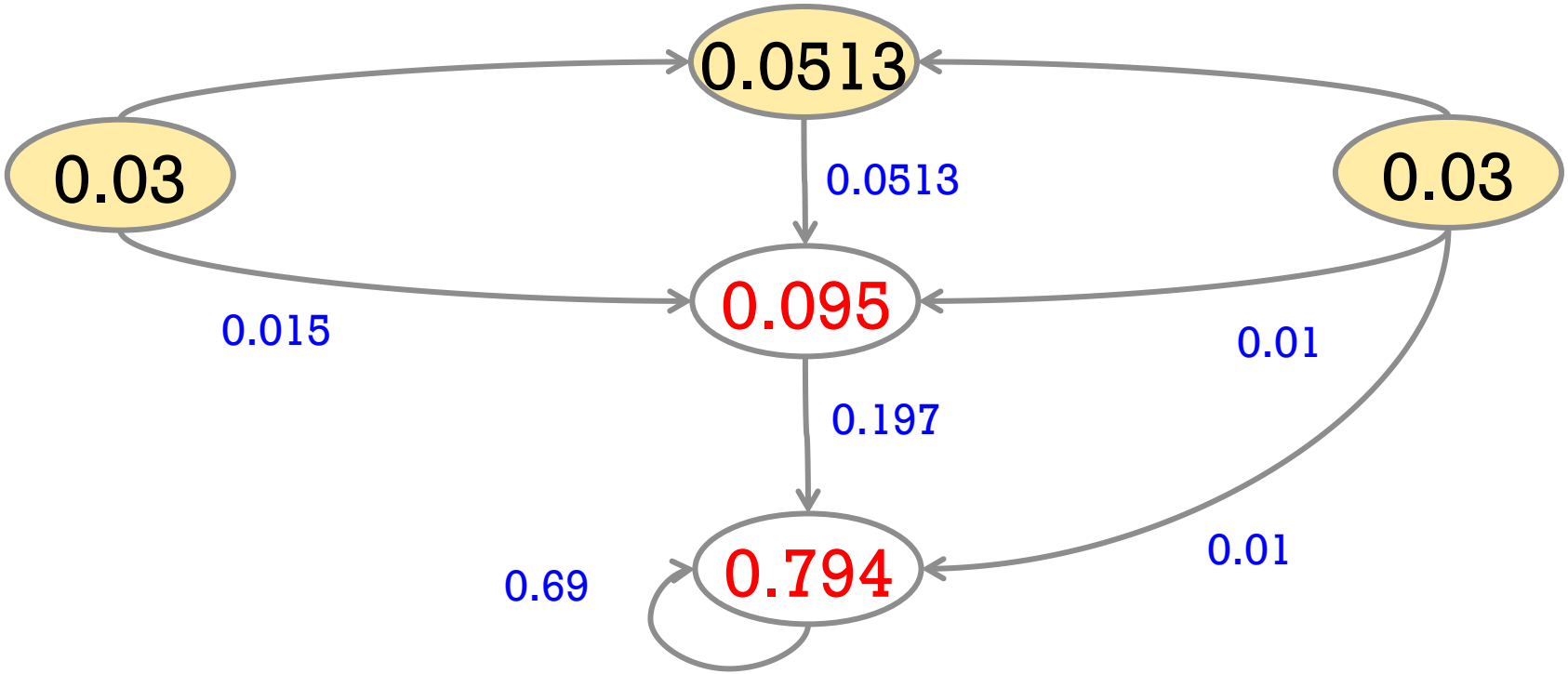
$$sum = sum(incoming\ values)$$

$$rank = \frac{0.15}{5 + 0.85 * sum}$$



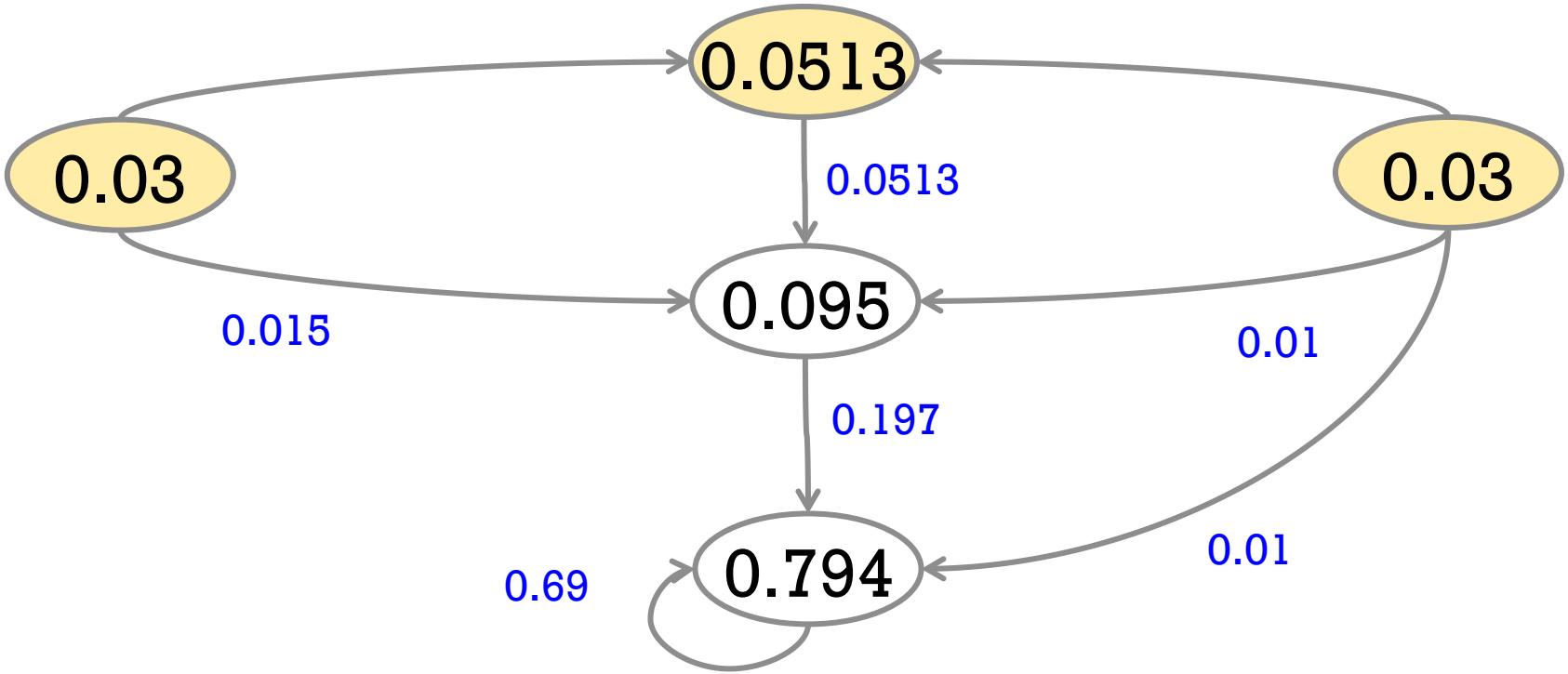
$$sum = sum(incoming\ values)$$

$$rank = \frac{0.15}{5 + 0.85 * sum}$$



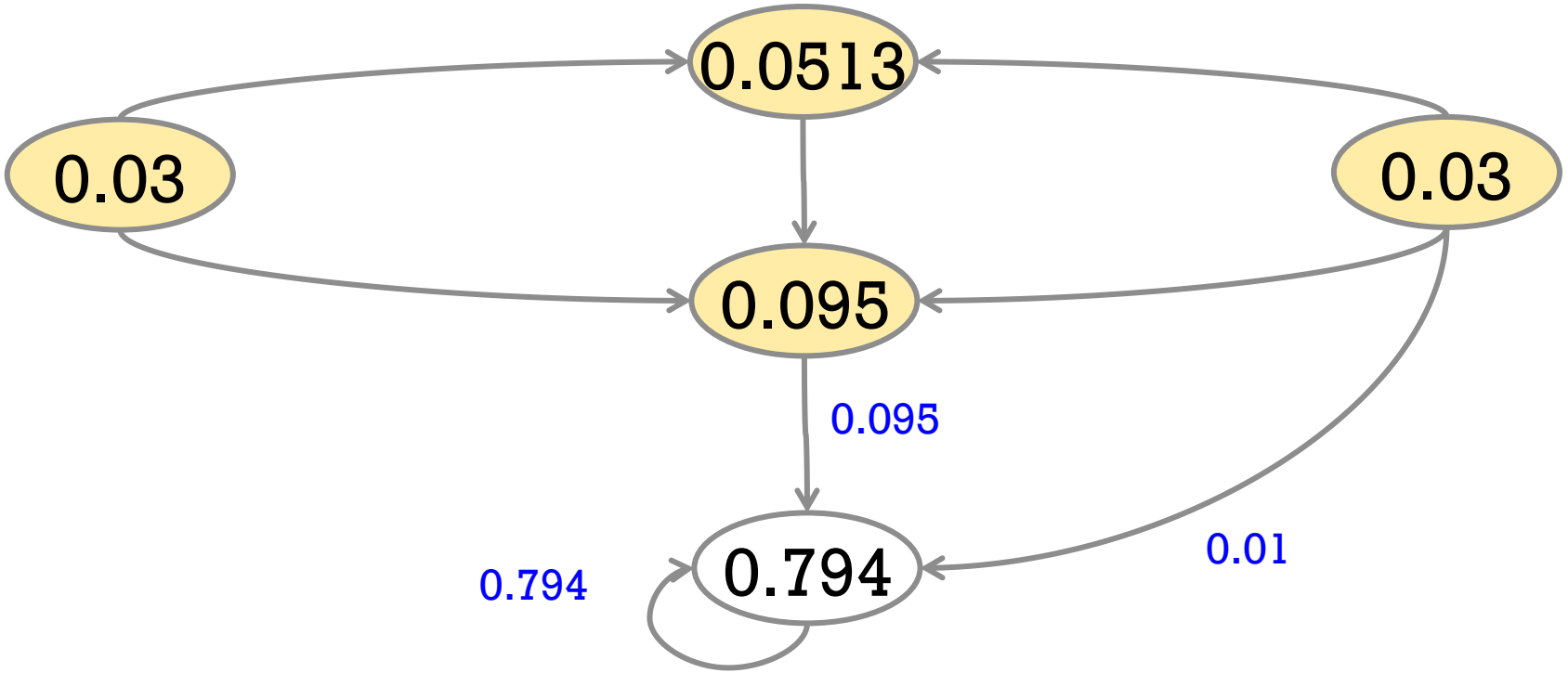
$$sum = sum(incoming\ values)$$

$$rank = \frac{0.15}{5 + 0.85 * sum}$$



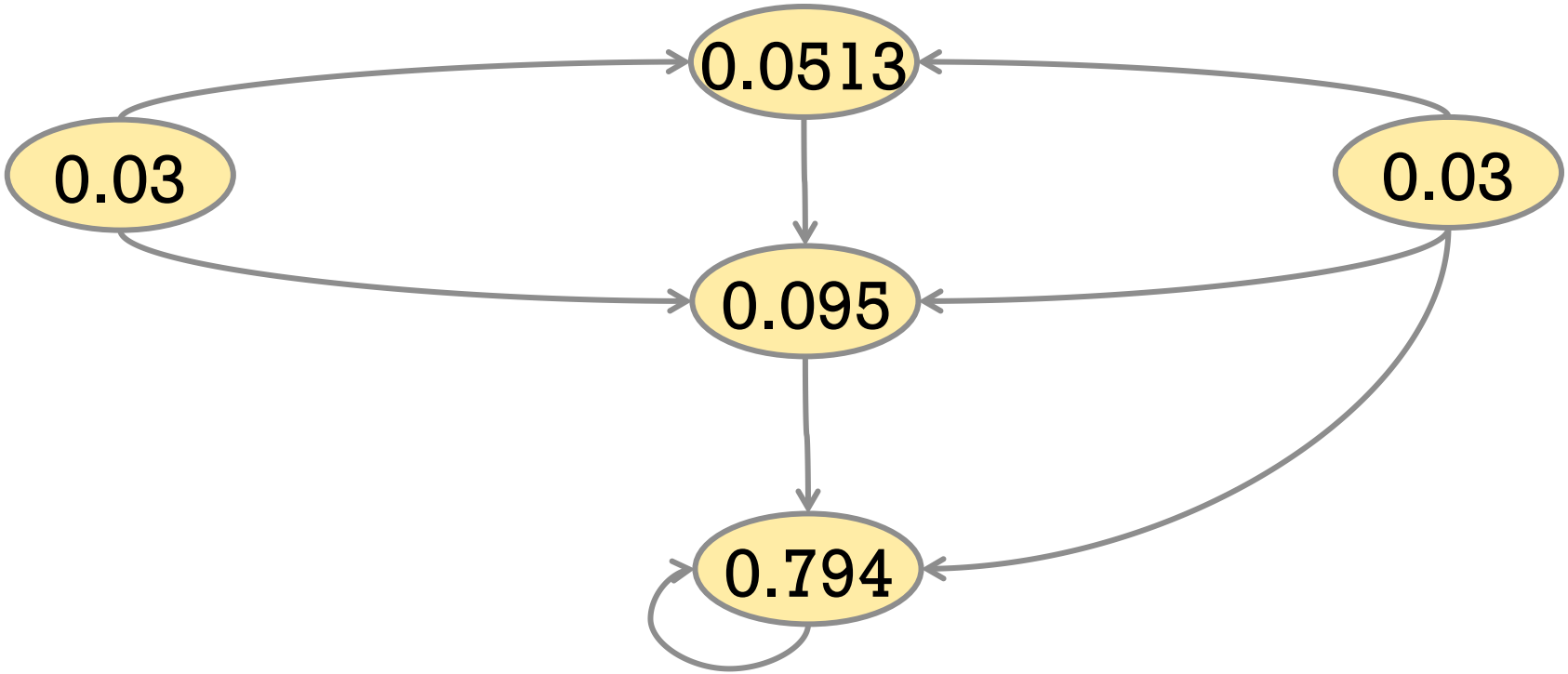
$$sum = sum(incoming\ values)$$

$$rank = \frac{0.15}{5 + 0.85 * sum}$$



$$sum = sum(incoming\ values)$$

$$rank = \frac{0.15}{5 + 0.85 * sum}$$



$$sum = sum(incoming\ values)$$

$$rank = \frac{0.15}{5 + 0.85 * sum}$$

SLIDES CAN BE FOUND AT:

TEACHINGDATASCIENCE.ORG