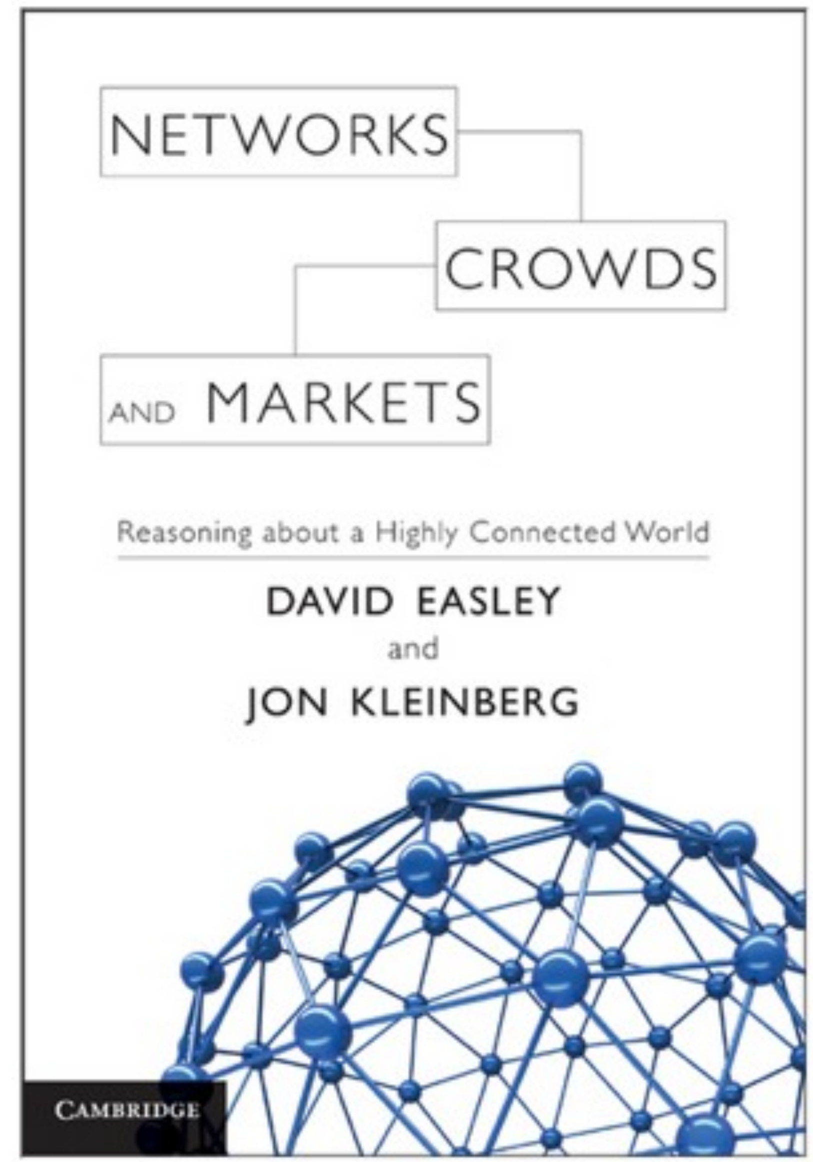
Structure of Social Networks

Outline

- Structure of social networks
- Applications of structural analysis



Social *networks*

Who

- Twitter
- Facebook
- Linked-in
- IMs
- Email
- Real life
- Address books
- ...

Twitter #numbers

140 characters

140 million active users

340 million tweets per day

> 15 billion connections

The Twitter Graph

Note: NOT (just) a social network

Asymmetric, follow relationship

VERY skewed graph

But very valuable "interest graph"

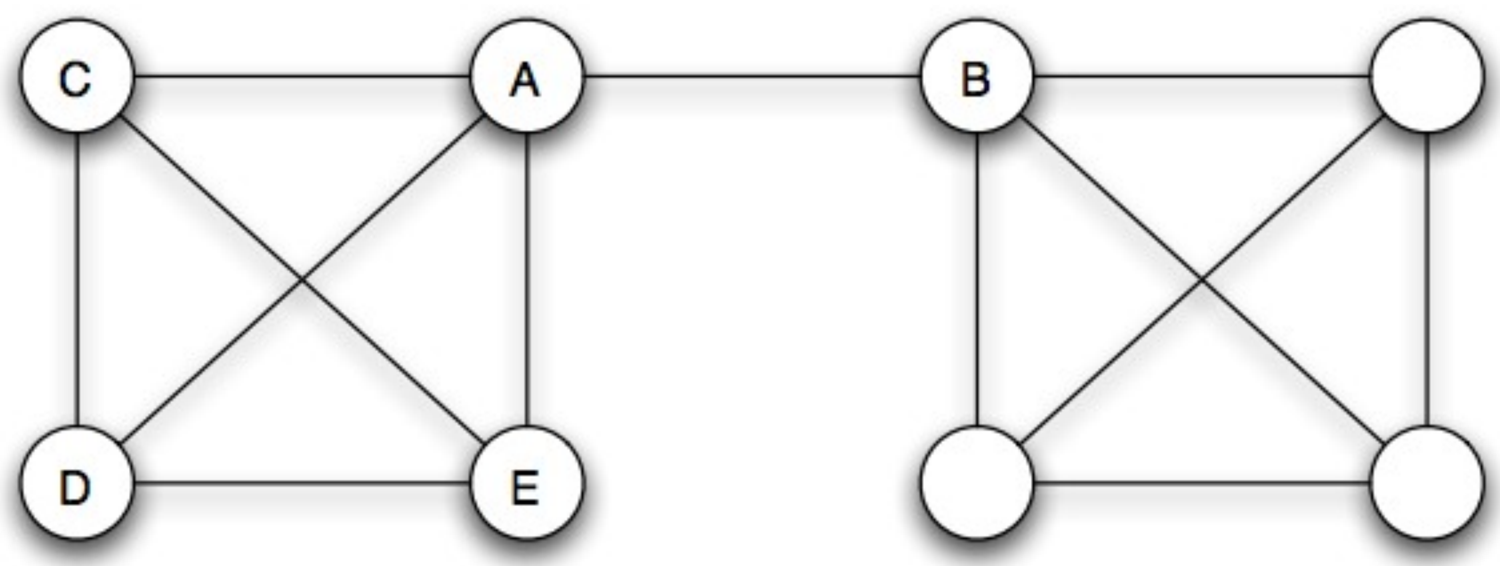
Part I: Network Structure

What can networks tell us?

- The strength of weak ties [Granovetter ’73]
  - How do people find new jobs?
  - Friends and acquaintances
  - Surprising fact: discovery is enabled by weak ties

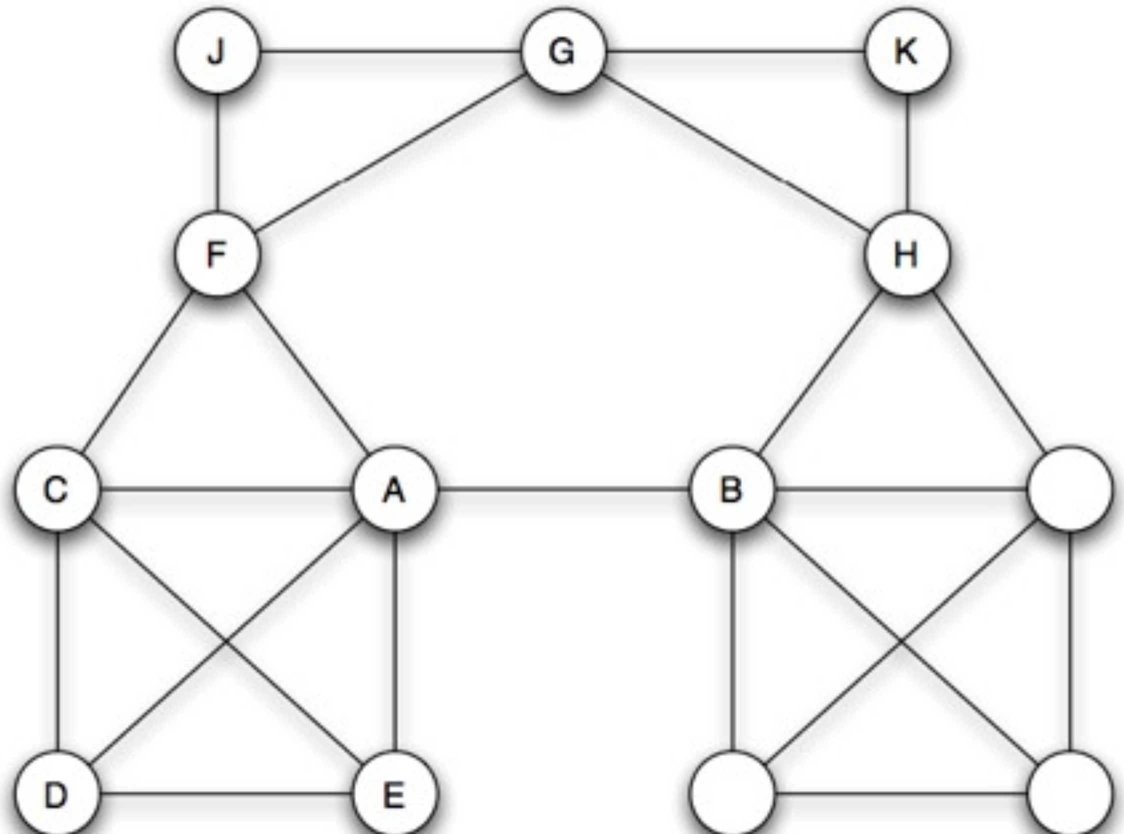
Strength of weak ties

- Definition: a *bridge* in a graph is an edge whose removal disconnects the endpoints.

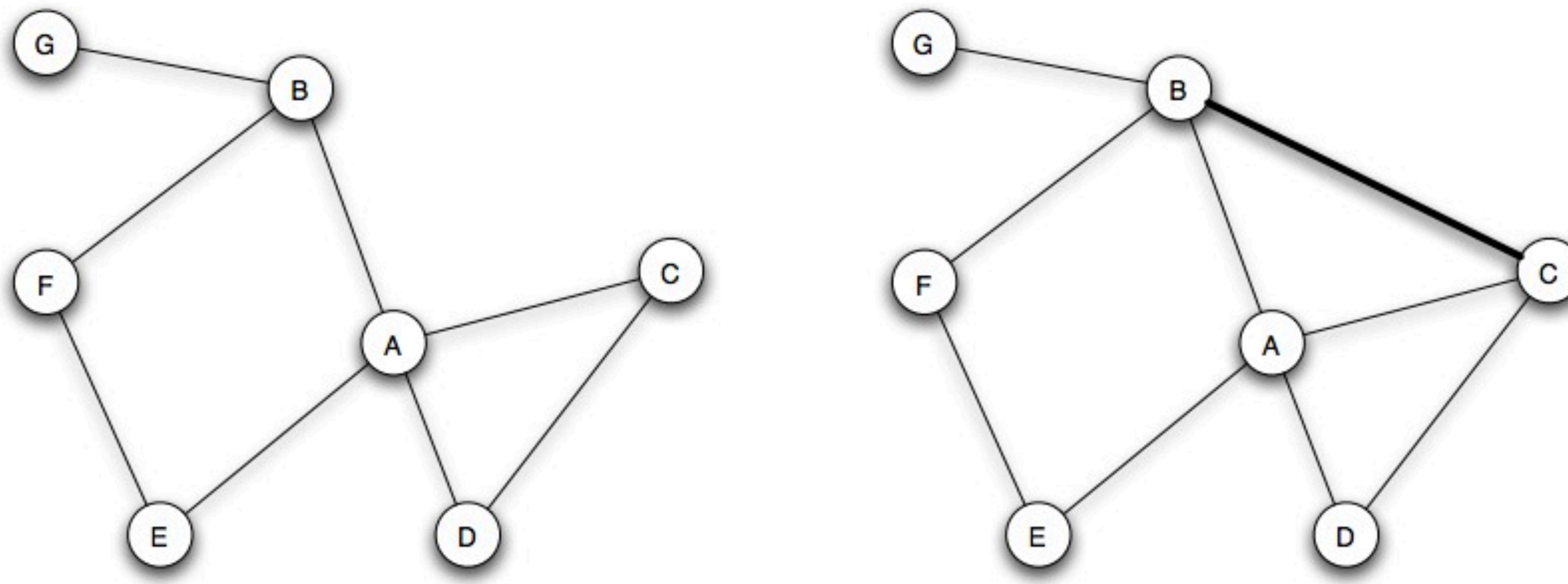


Strength of weak ties

- Definition: a *local bridge* in a graph is an edge whose endpoints have no common neighbor.

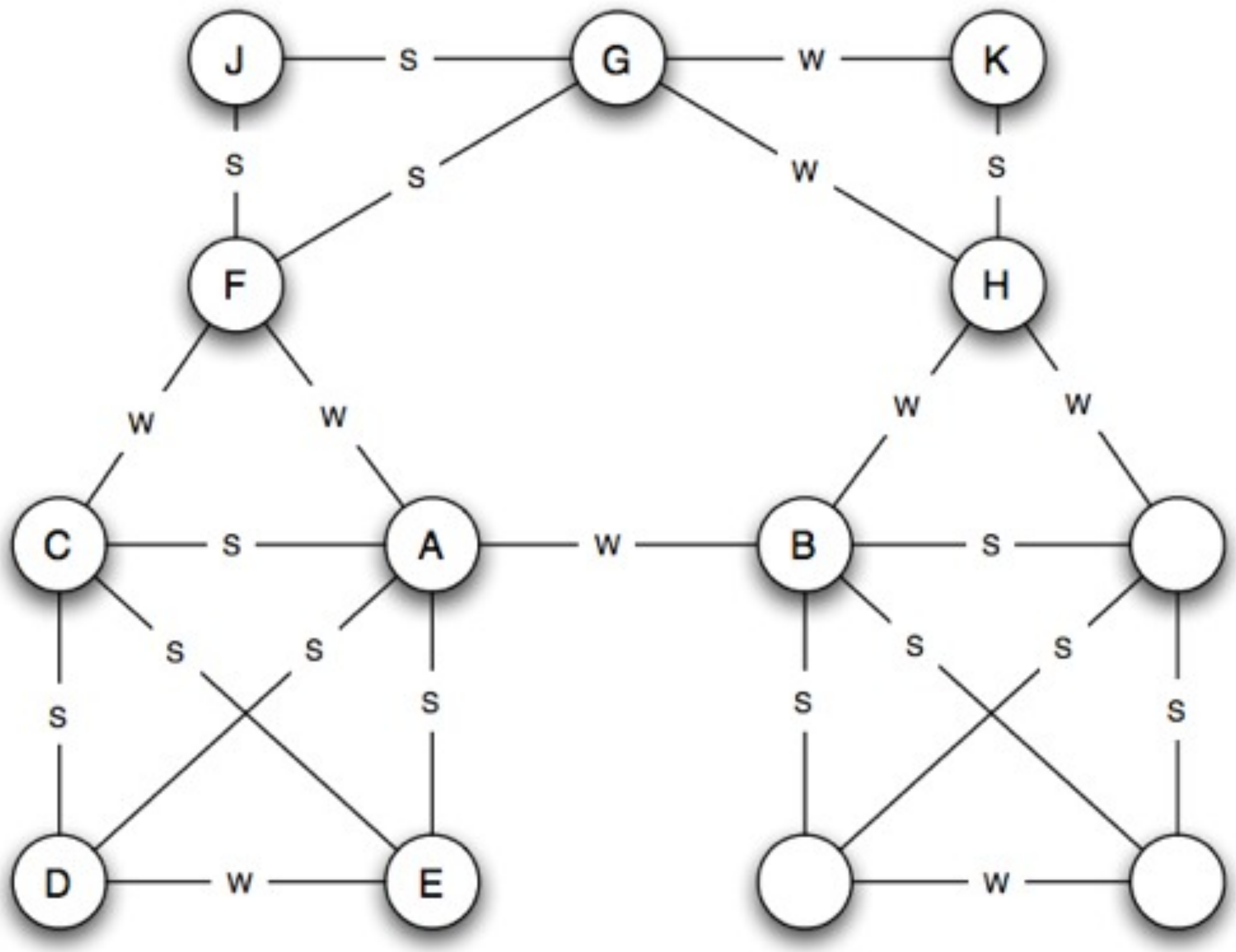


Triadic closure



Strong Triadic closure

Strong Triadic Closure Property: if the node has strong ties to two neighbors, then these neighbors must have at least a weak tie between them.



Strength of Weak Ties

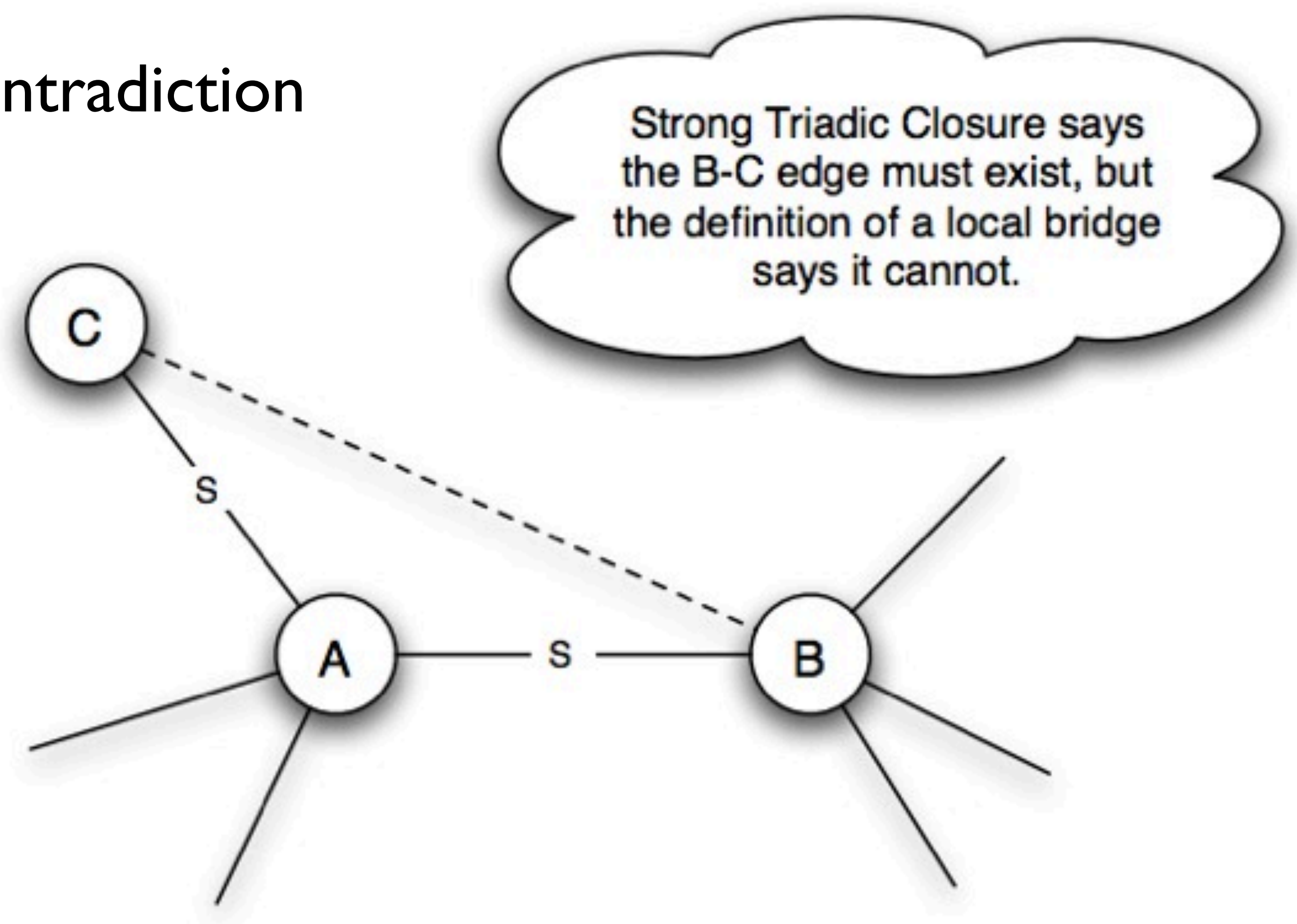
Claim: If a node A in a network satisfies the Strong Triadic Closure Property and is involved in at least two strong ties, then any local bridge it is involved in must be a weak tie.

Consequence: all local bridges are weak ties!

Strength of Weak Ties

Claim: If a node A in a network satisfies the Strong Triadic Closure Property and is involved in at least two strong ties, then any local bridge it is involved in must be a weak tie.

Proof: by contradiction

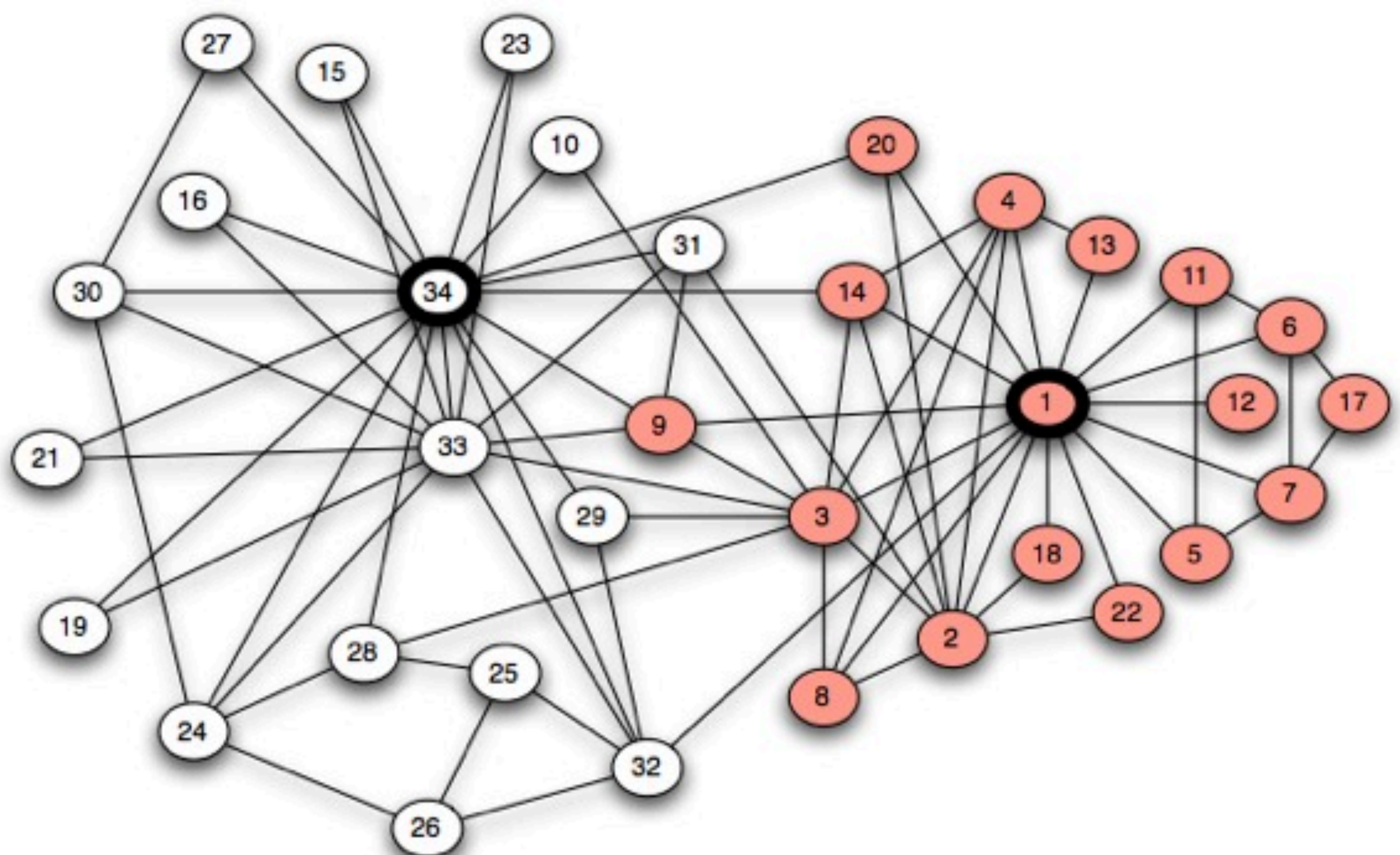


Strength of Weak Ties

- Discovery is enabled by weak ties
  - Surprising strength of weak ties!
- Simple structural model explains this cleanly
- Similar observations for Twitter/Facebook

More network insights

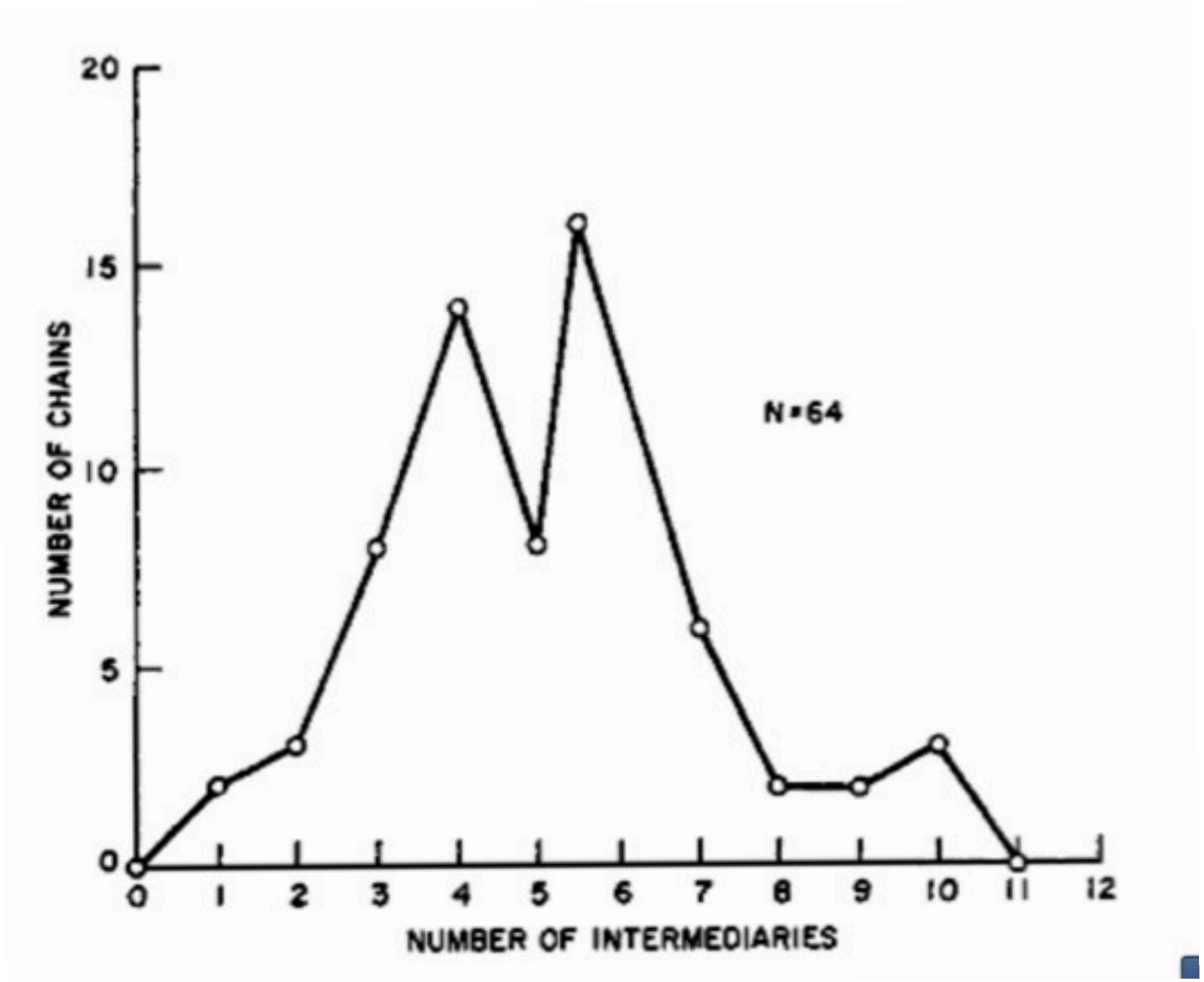
- Zachary's karate club and graph partitioning



Six degrees of separation

- Milgram's experiment:
    - How are people connected?
    - Letters given to people in Omaha
    - Target is a person in Boston
    - Rule: can only forward letters to people *you know* (<== social network)
    - How many hops did it take?

Milgram's experiment



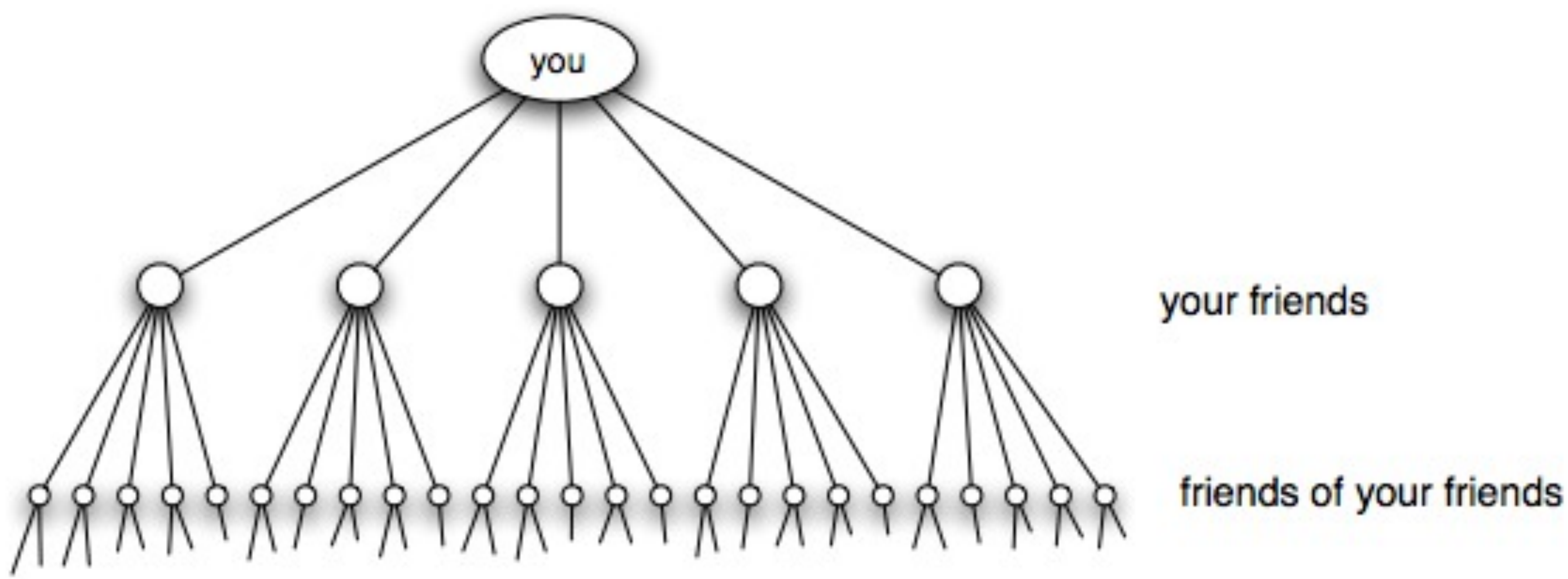
Six degrees of separation

- Bacon number game

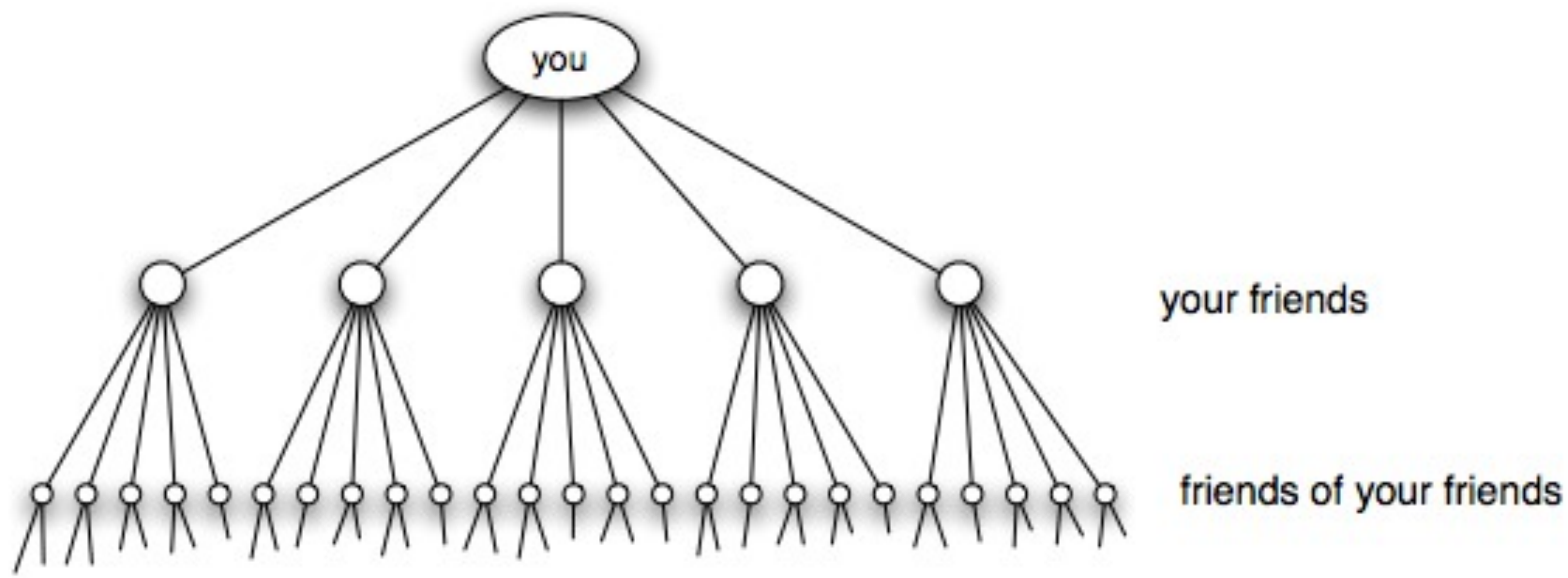
- Short paths exist, and abound!

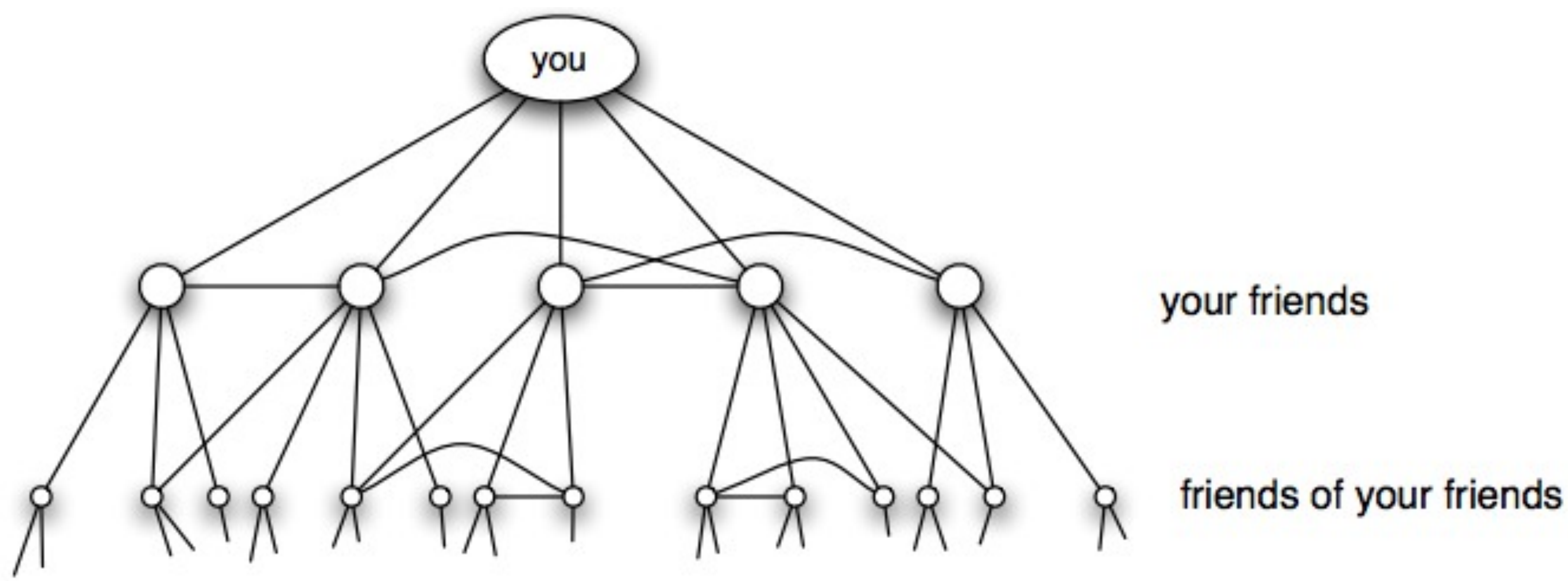
How?

- The world seems to be small
- But how does this happen?



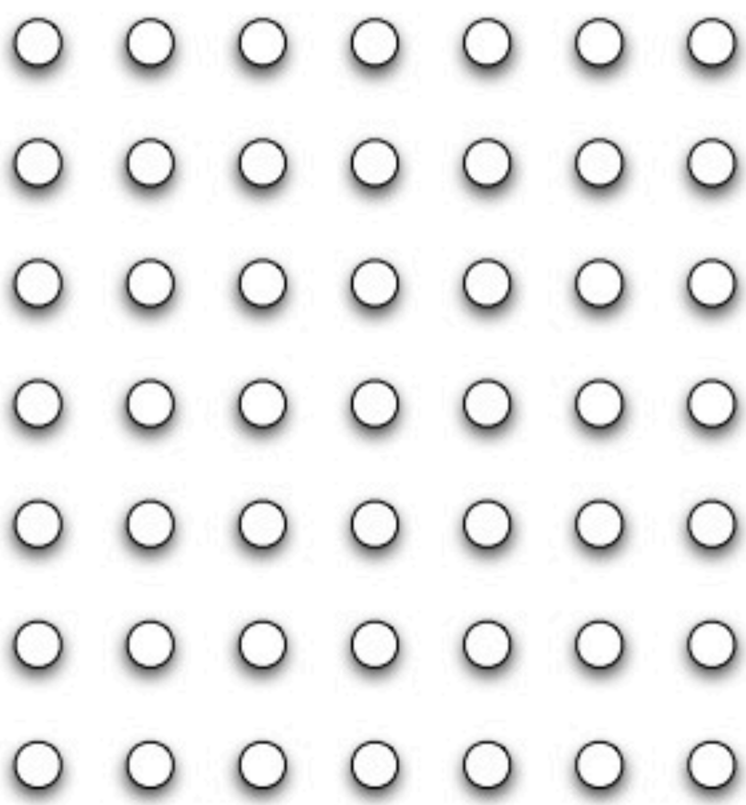
How?

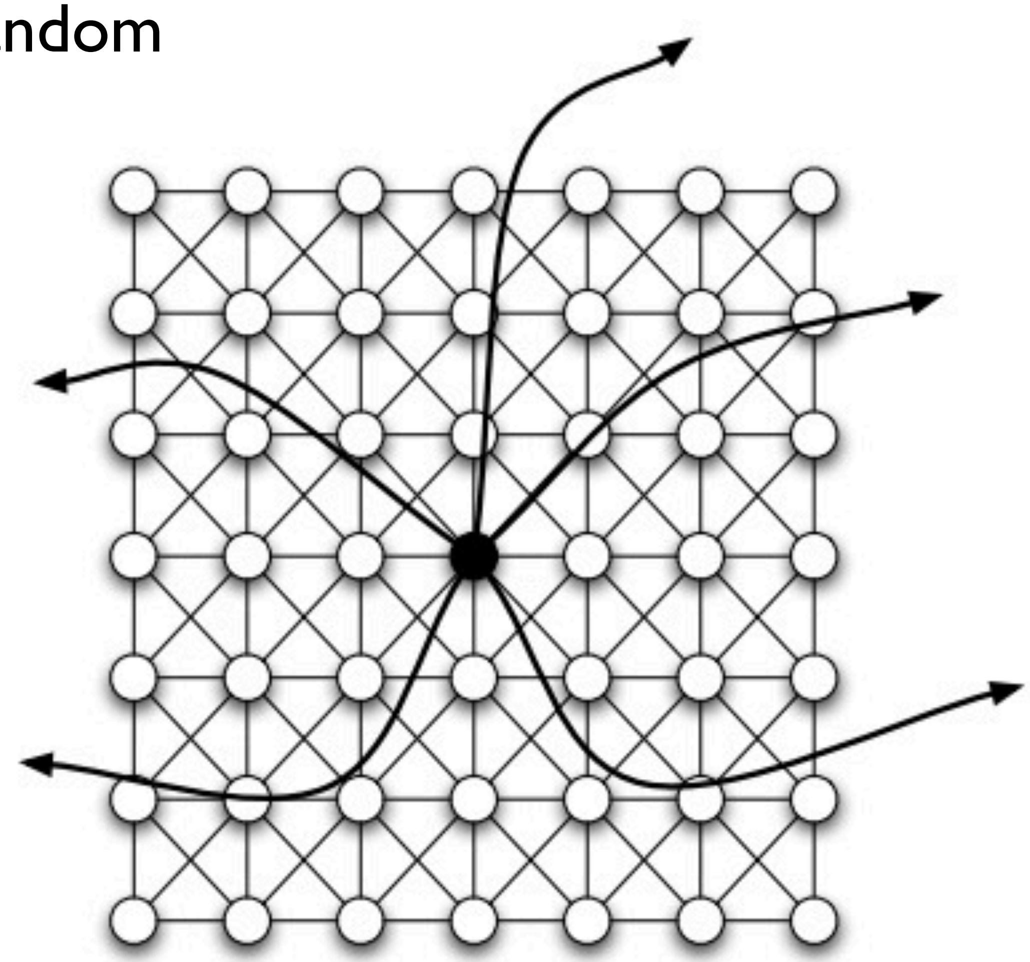

you
your friends
friends of your friends


you
your friends
friends of your friends

Watts-Strogatz model

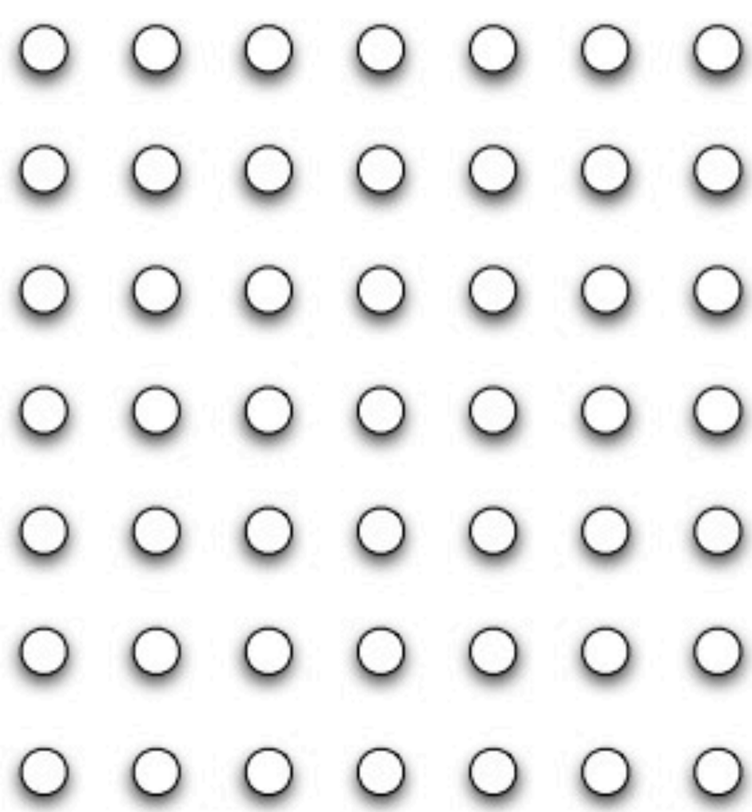
- Each node forms 2 links:
  - All nodes within r hops
  - k nodes picked uniformly at random

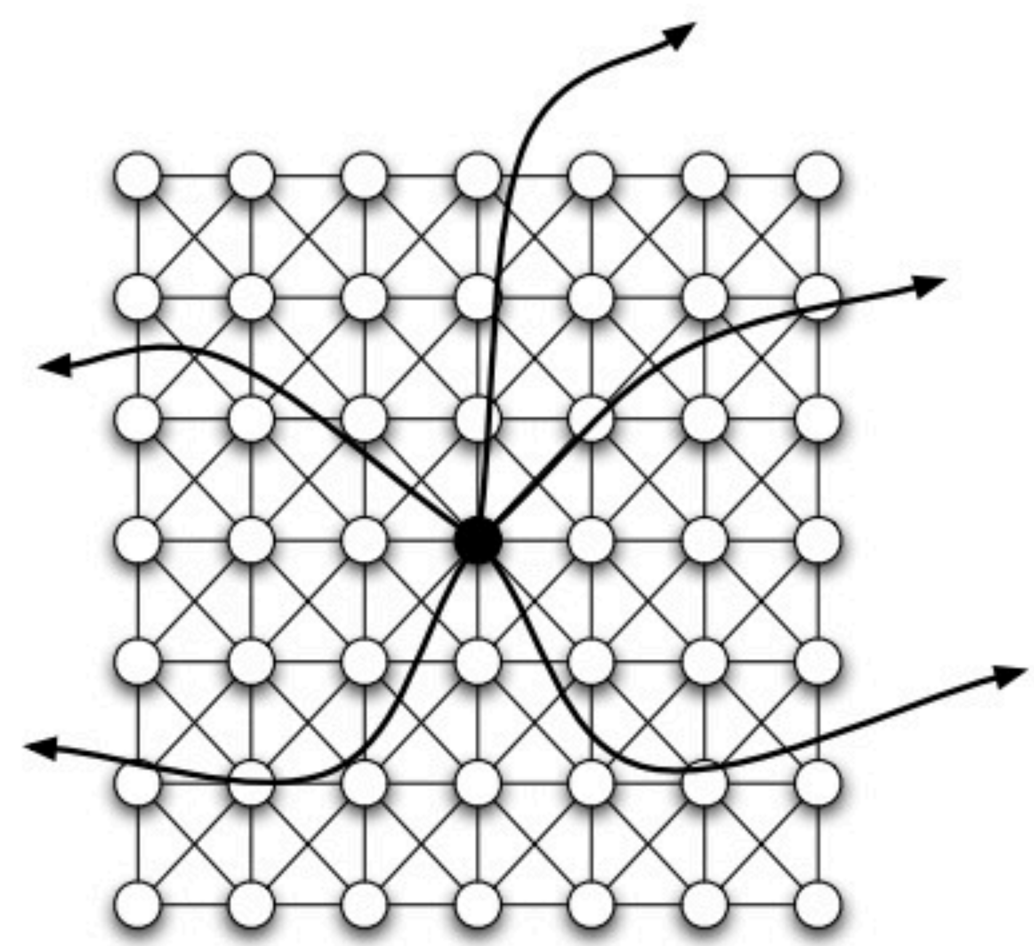




Watts-Strogatz model

- Property: short paths exist between everyone!
- Intuition: random links exponentiate due to lack of triangles

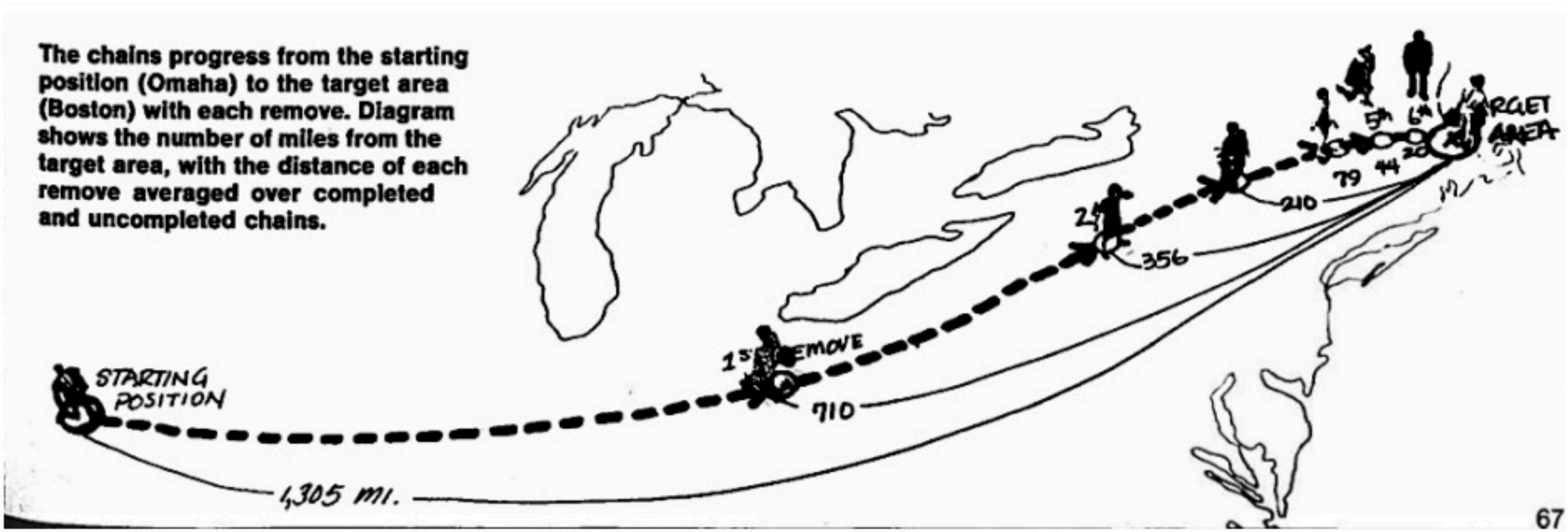




Decentralized search

- Is Milgram's experiment now explained?
  - Watts-Strogatz: short paths exist
  - But how can we possibly navigate?

Milgram's experiment



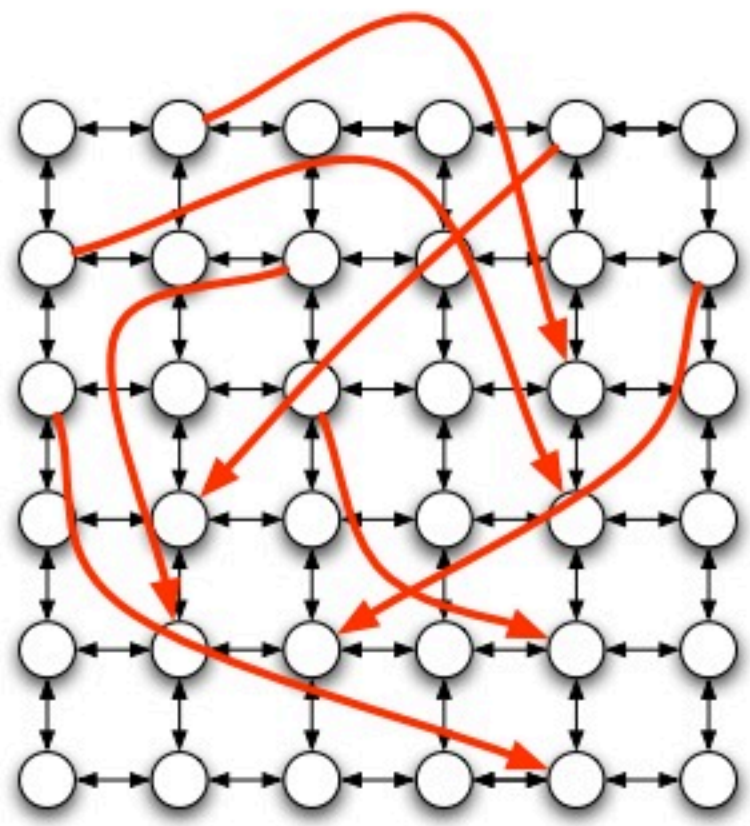
The chains progress from the starting position (Omaha) to the target area (Boston) with each remove. Diagram shows the number of miles from the target area, with the distance of each remove averaged over completed and uncompleted chains.

Watts-Strogatz revisited

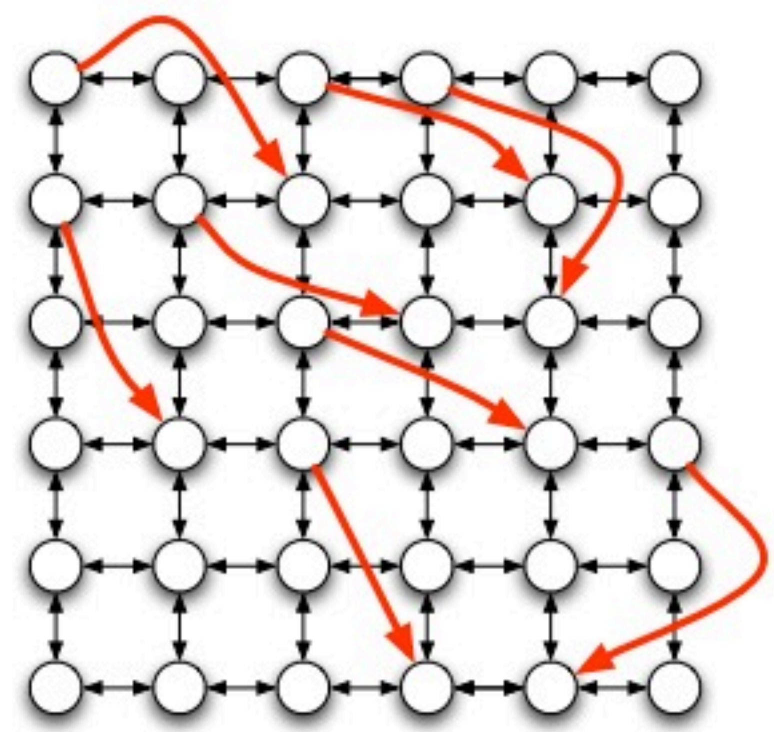
- Navigate using only local information
- Impossible to navigate in this world!
  - So does *not* explain Milgram's experiment
  - Intuition: random links are *too* random

Kleinberg's model

- Slight change: given exponent q, for nodes u, v create a link with probability $\mathrm{d}(u,v)^{-q}$

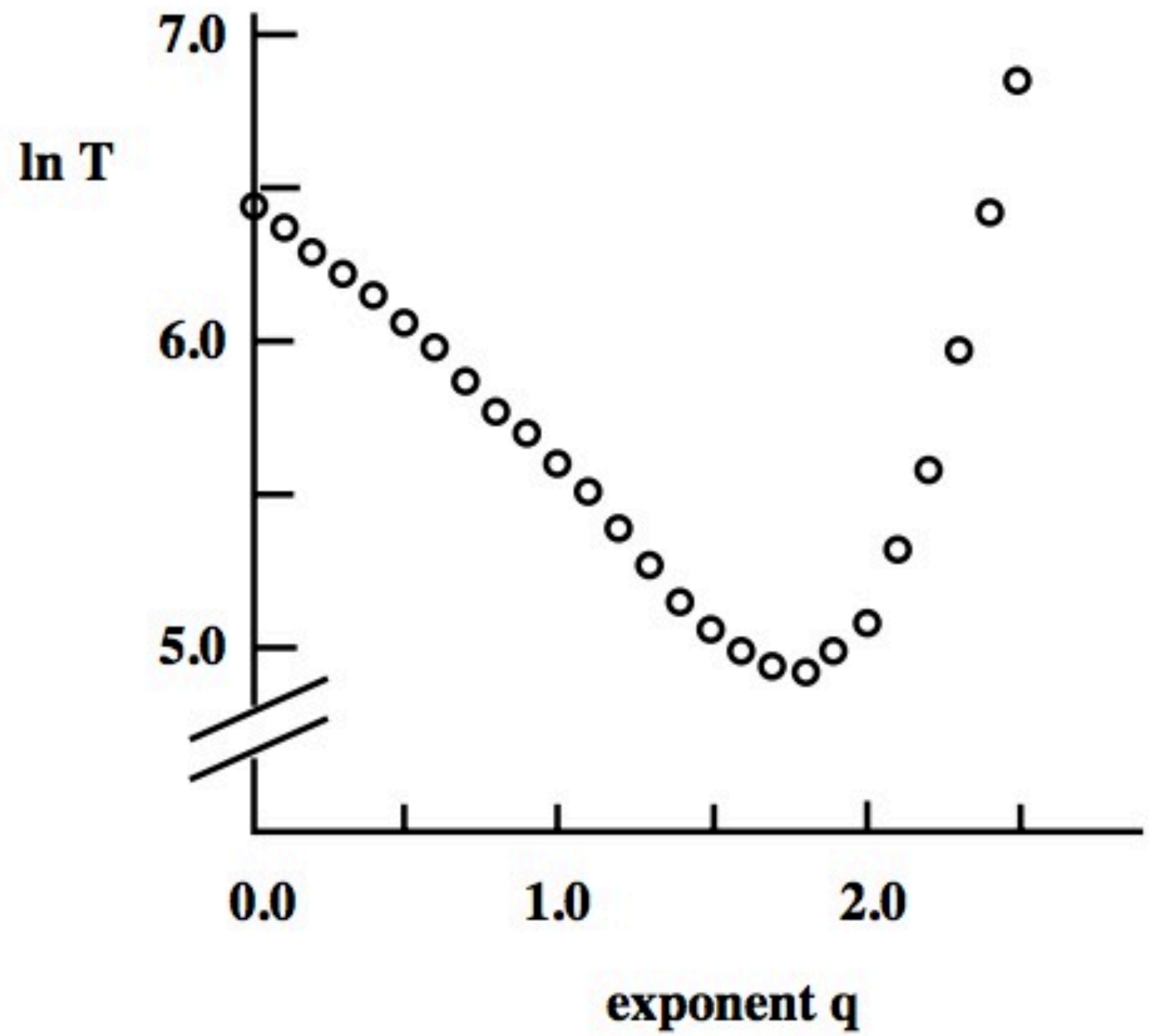


(a) *A small clustering exponent*



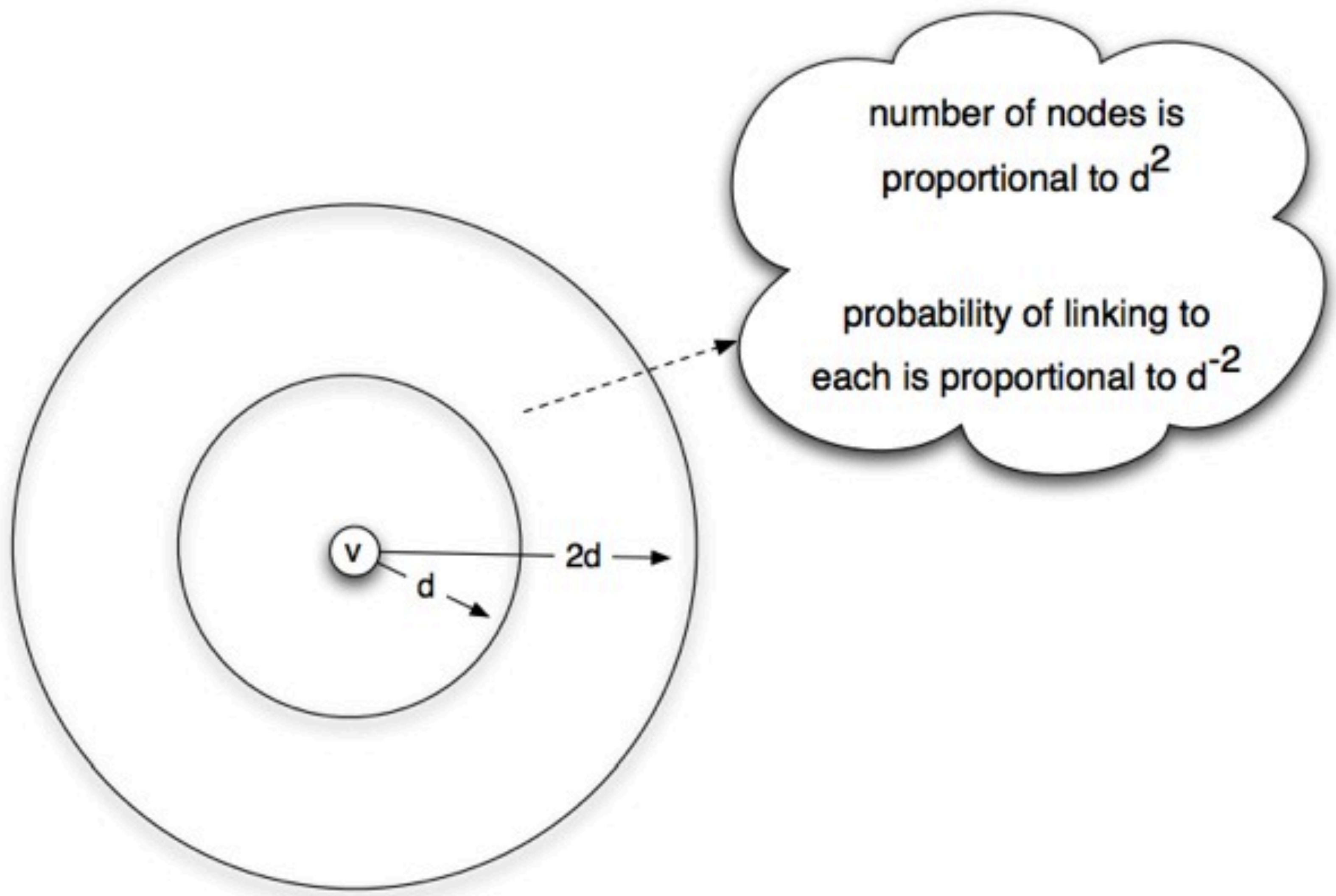
(b) *A large clustering exponent*

Kleinberg's model



Kleinberg's model

- Some intuition:



Network structure

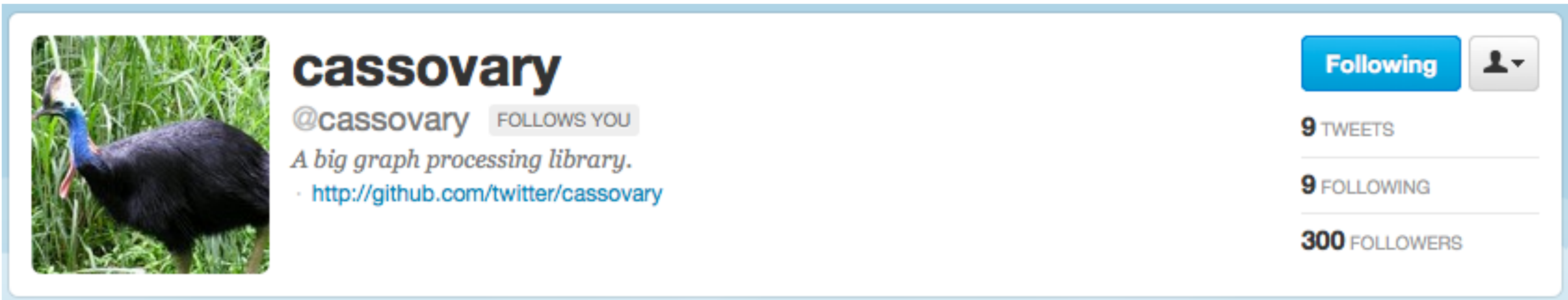
- Understanding structure affords deep insights
- Interplay between sociology and graph theory

Part II: Applications

Part II: Applications

- Recommendations

Graph Engine: Cassovary



In-memory computation

No compression!

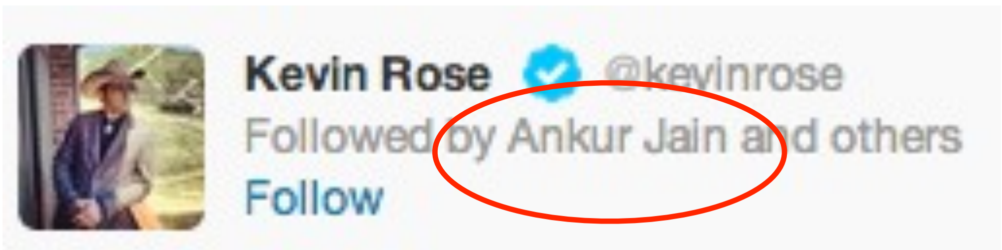
Adjacency list format

Open source: https://github.com/twitter/cassovary

Link Prediction

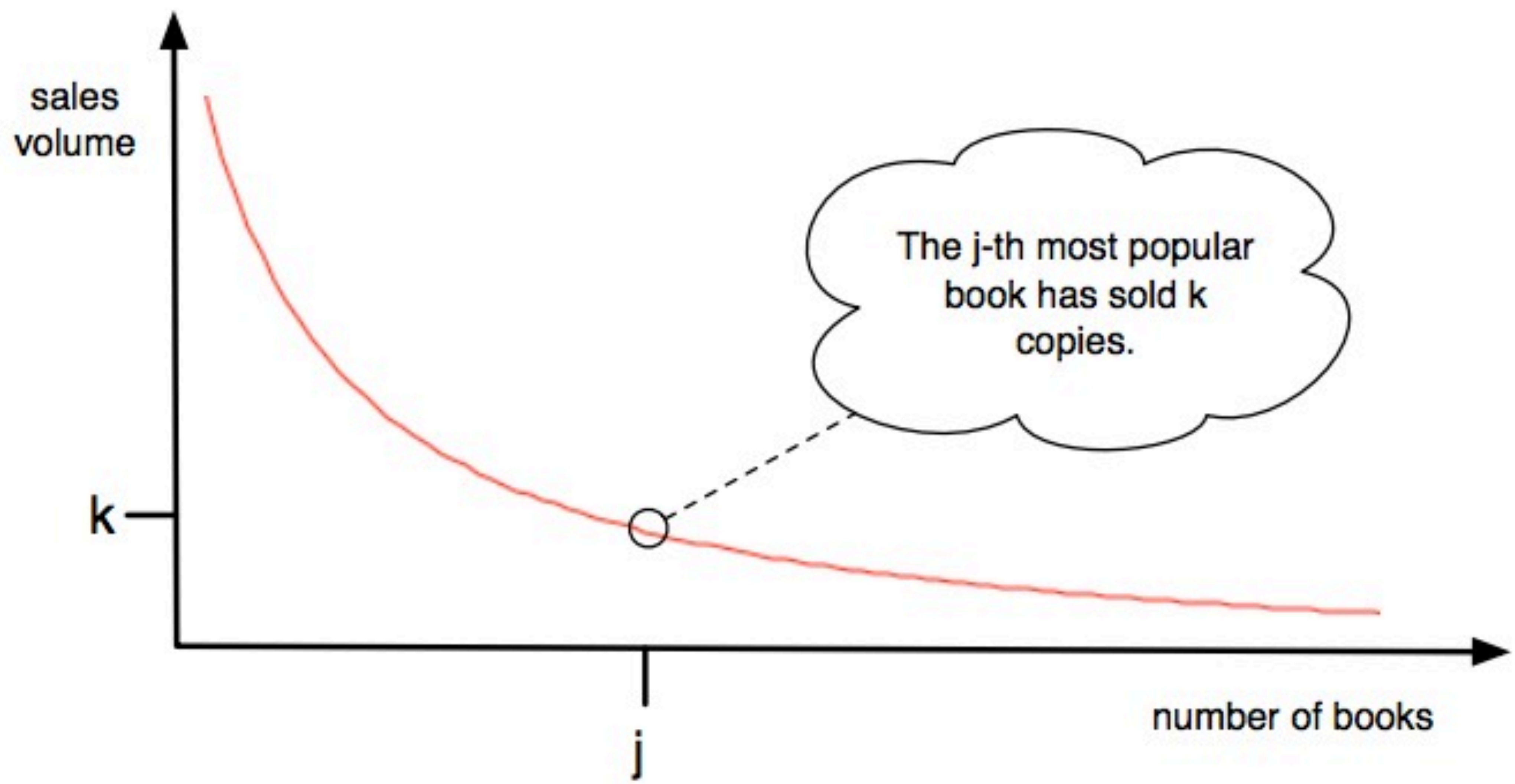
- General algorithms:
  - Most popular in a country
  - Popular movie stars, pop stars, etc

- Personalized algorithms
  - Triadic closure
  - Personalized pagerank
  - SALSA



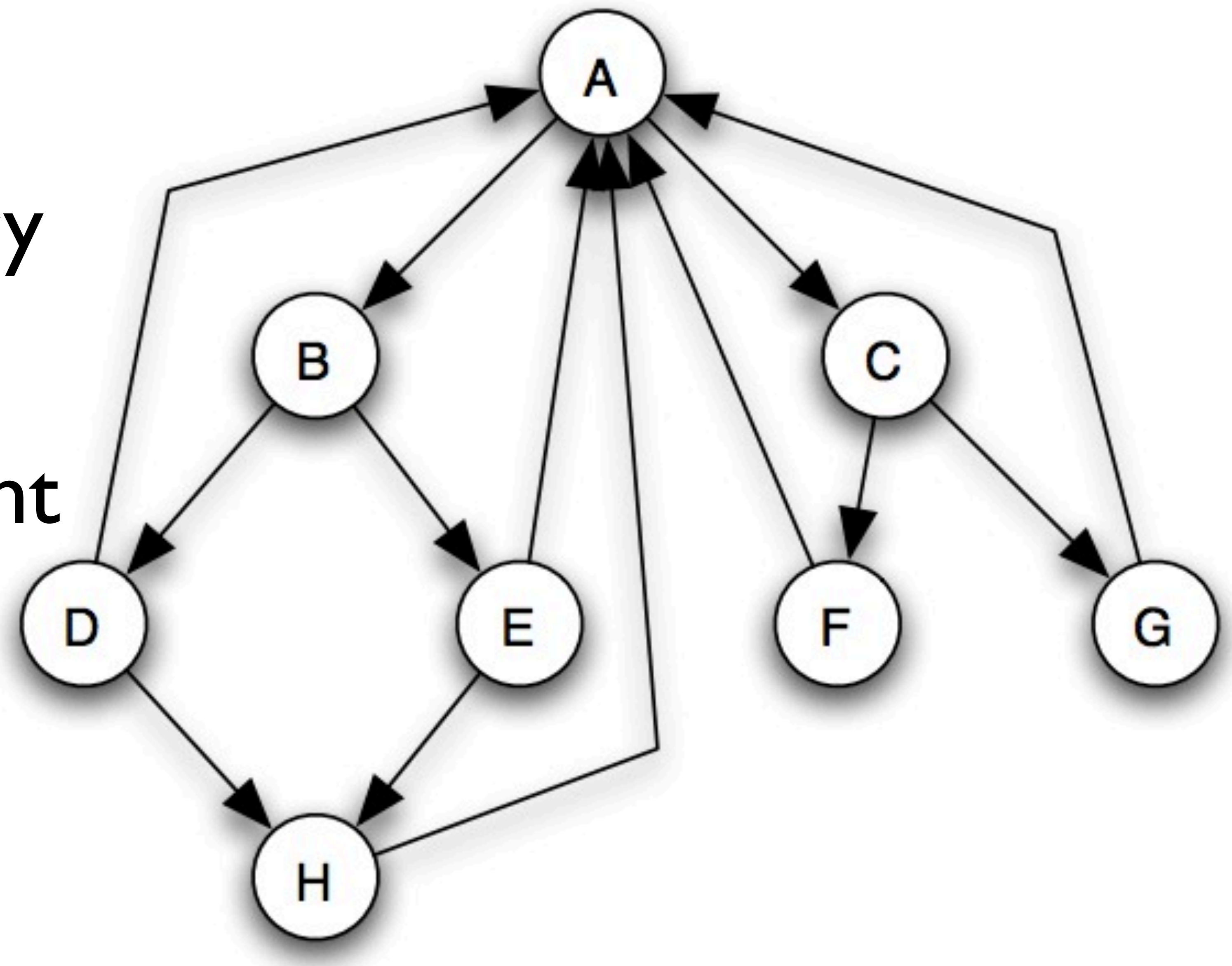
- Data: Social graph + usage (relationship strength, interests)

The Long Tail

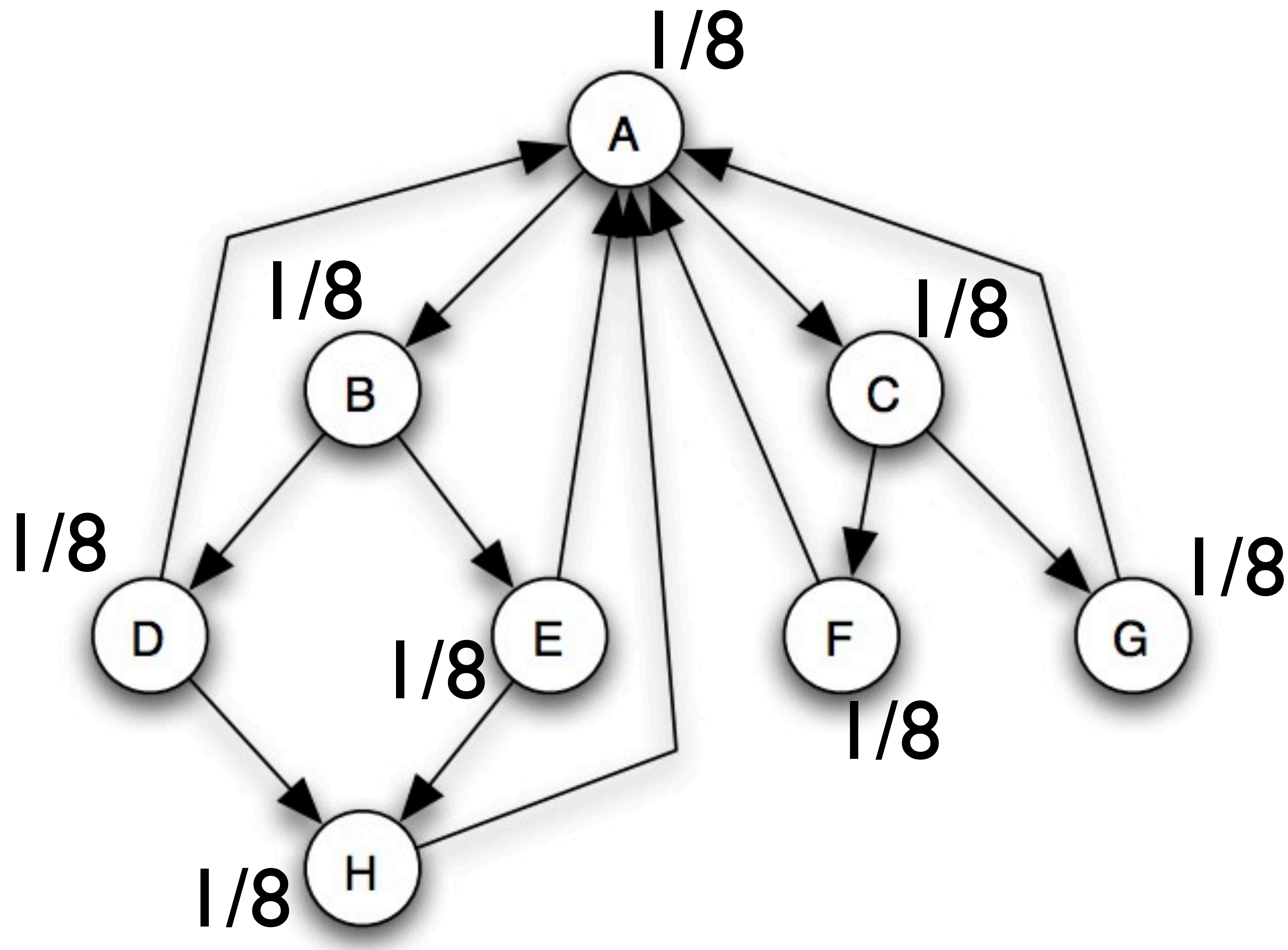


Pagerank

- Start with equal weights on every node
- Distribute weight equally among outgoing edges

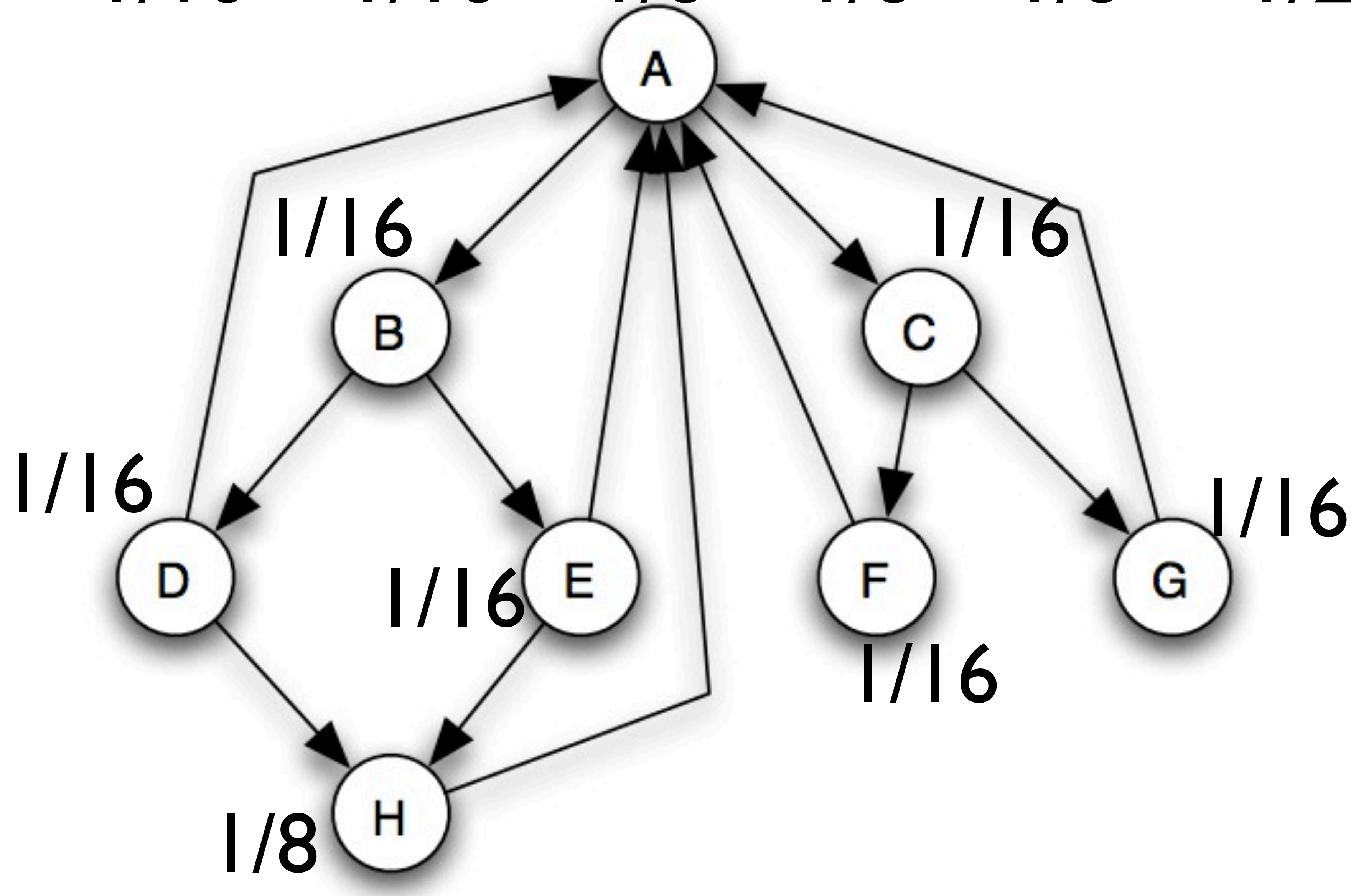


Pagerank

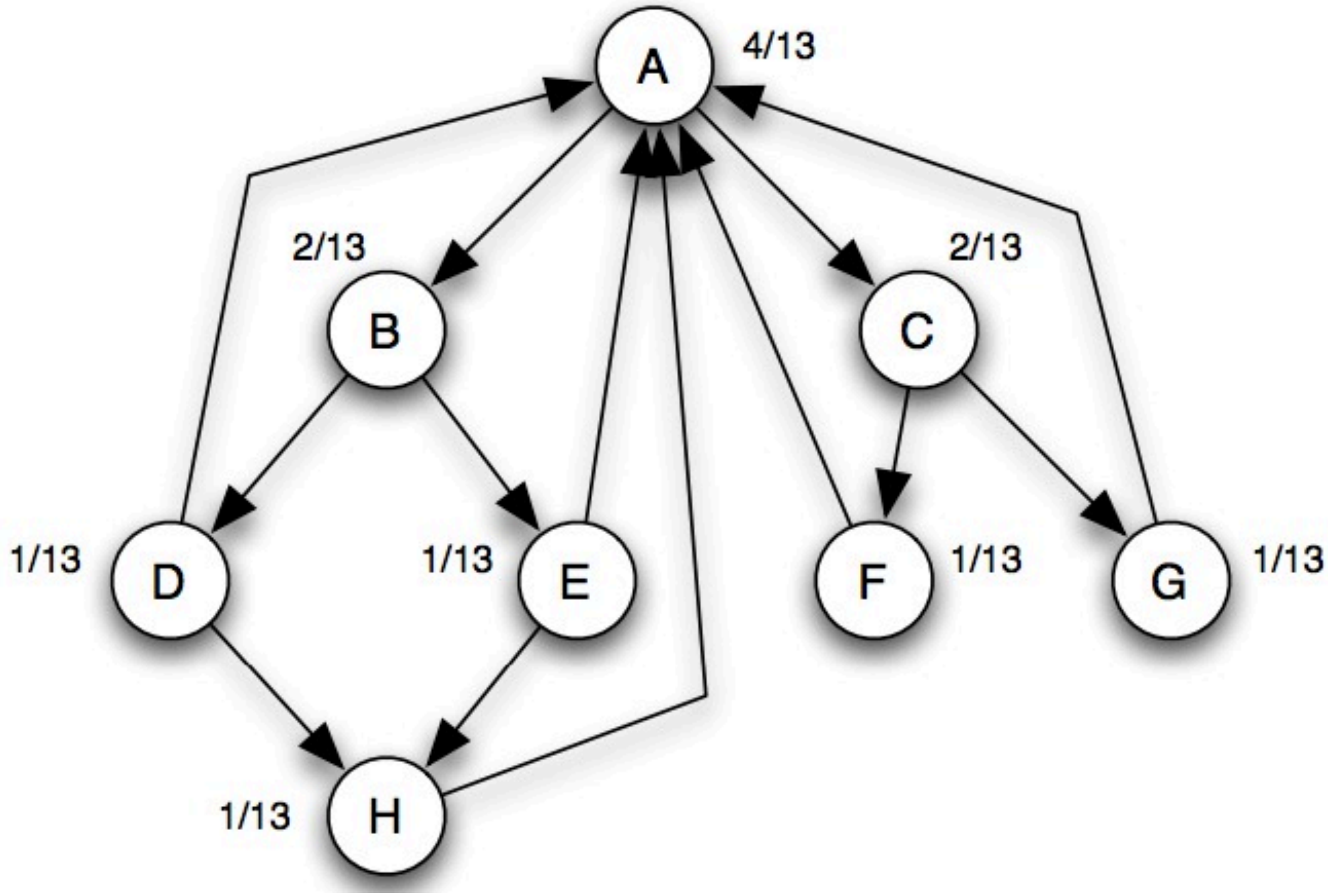


Pagerank

$1/16 + 1/16 + 1/8 + 1/8 + 1/8 = 1/2$

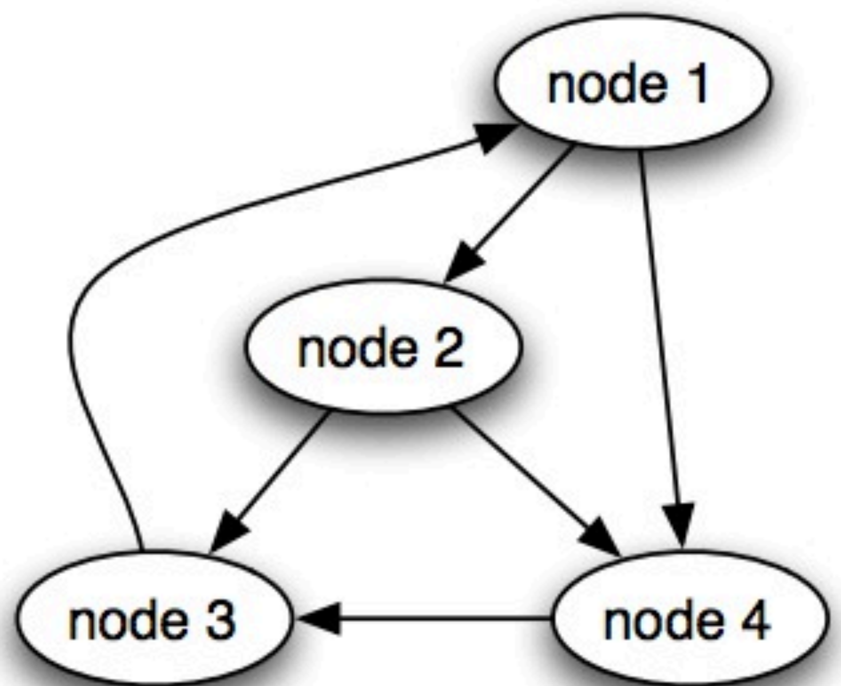


Pagerank



Pagerank

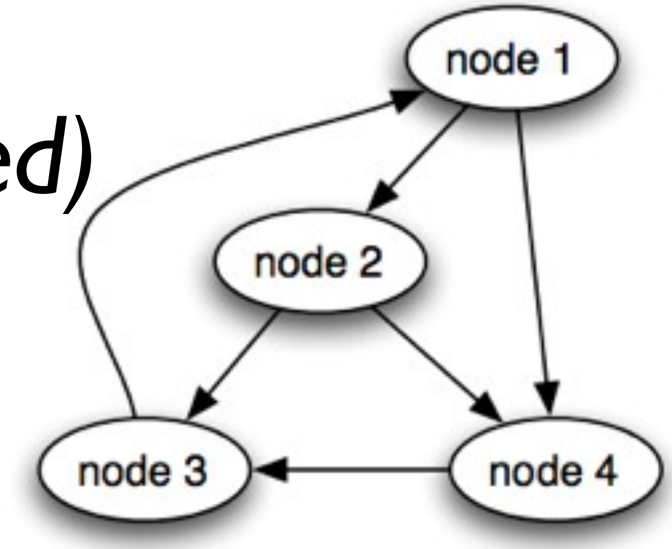
- Alternate matrix formulation:
  - Adjacency matrix *A*



$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Pagerank

- Alternate matrix formulation:
  - Adjacency matrix *A (normalized)*



$$\begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1/4 & 1/8 & 3/8 & 1/4 \end{bmatrix}$$

Pagerank

- Matrix view:

$$\pi^t \cdot A = \pi^{t+1}$$

$$\pi^0 \cdot A^t = \pi^t$$

- Convergence via Eigenvectors:

$$\bar{\pi} \cdot A^t = \bar{\pi}$$

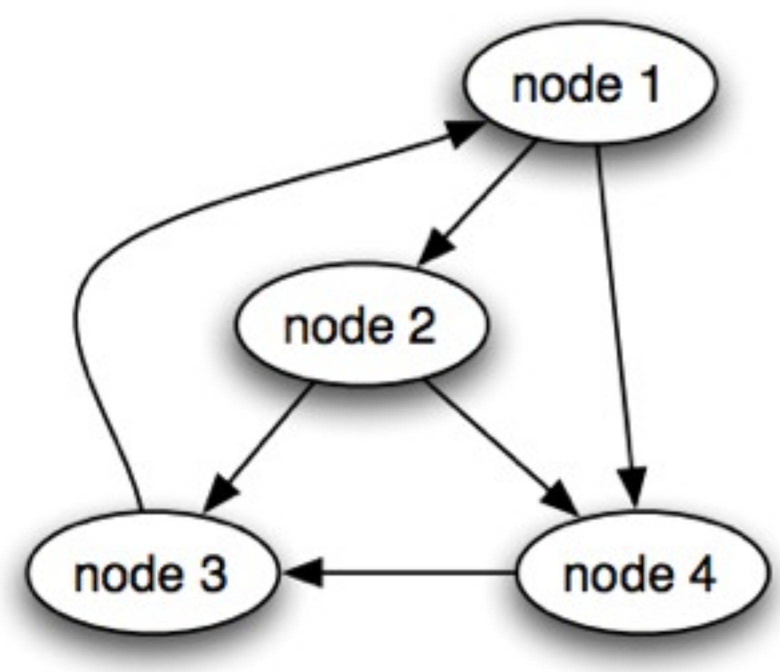
Personalized-pagerank

- A personalized version of pagerank
  - Reset to source *u* with probability λ

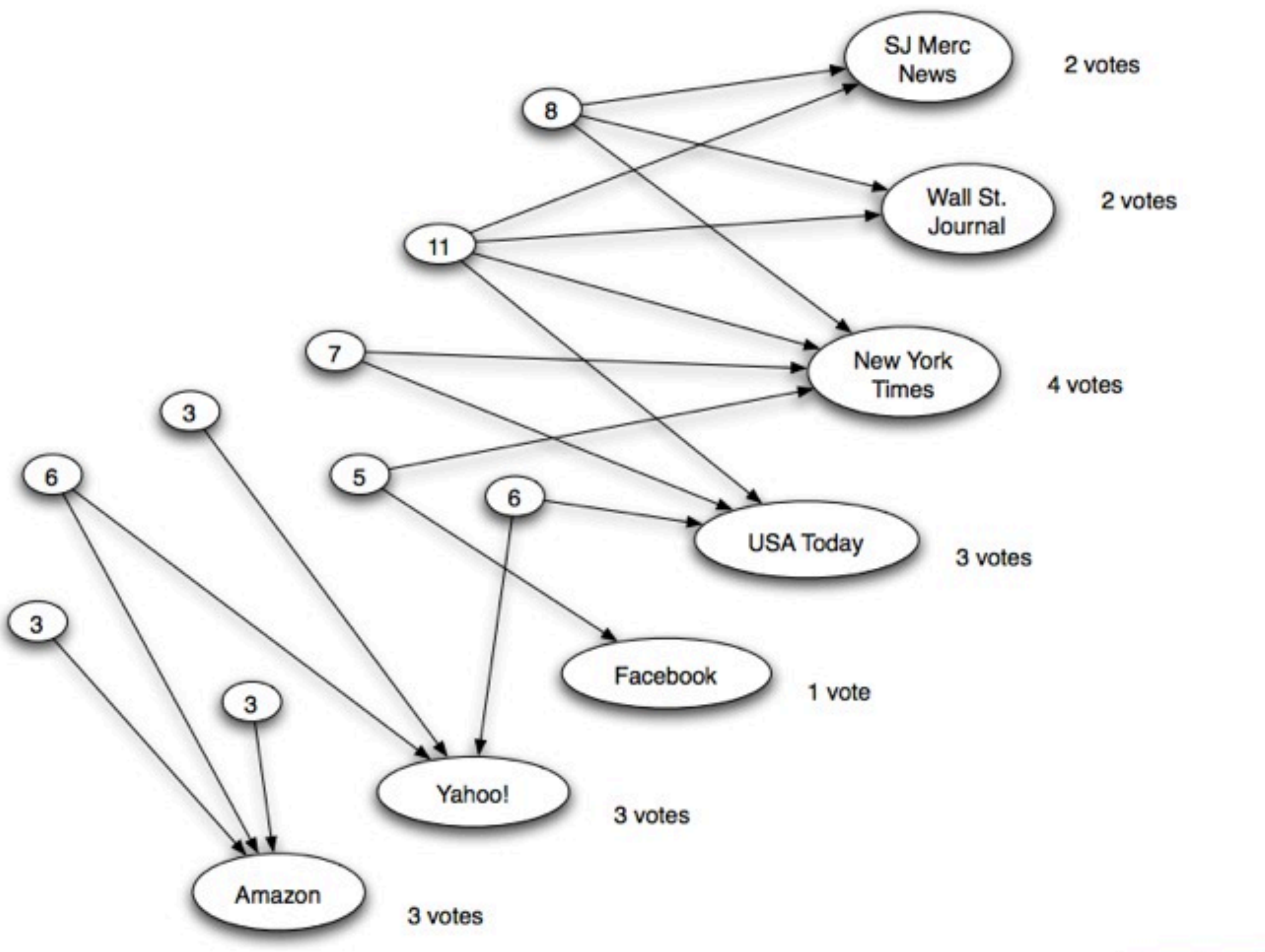
$$\lambda \pi^t \cdot A + (1 - \lambda)\pi_u = \pi^{t+1}$$

Personalized Pagerank

- Equivalent random walk view
  - Easier to simulate via Monte-carlo



HITS and SALSA



More link prediction

- Pagerank -> personalized pagerank
- HITS -> SALSA
- Many other variants
  - Similar users
  - Simrank

Scale

- A cautionary tale about scaling
  - Doing random walks
  - Weighted random walks
- Map-Reduce

Uniform Random Walk

The key sampling routine:

- For a user, find a random following/follower, i.e. pick uniformly at random from $[u_1, u_2, \ldots u_n]$

Budget per call: 20ms / 10K steps = 2000 ns /step

Main memory reference: 100 ns

Needs a call to random(), so just within budget

Weighted Random Walk

But what if there are weights:

- For a user, find a random following/follower from $[u_1, u_2, \ldots u_n]$ with weights $[w_1, w_2, \ldots w_n]$

n could be very large!

Weighted Random Walk

Many simple ideas don't work:

- Pre-process + binary search:

$$[w_1 > w_2 > \ldots > w_n]$$

Pay $O(\log n)$ for every step

Cost for Obama: 20 comparisons + lookups

Recall: lookup is 100 ns, so out of budget :(

Weighted Random Walk

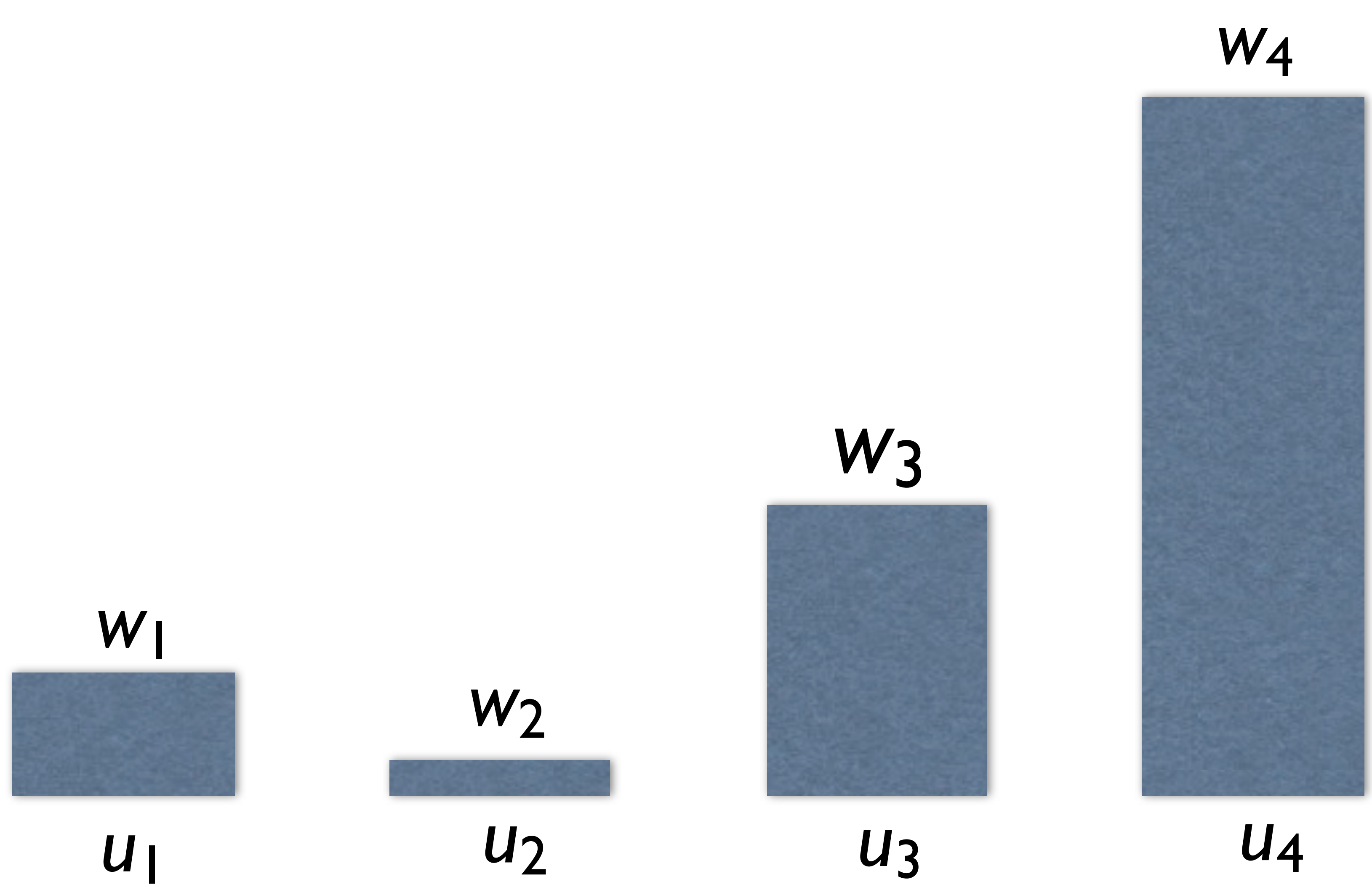
We can do a lot better!

In fact, we can do it in $O(1)$ lookups

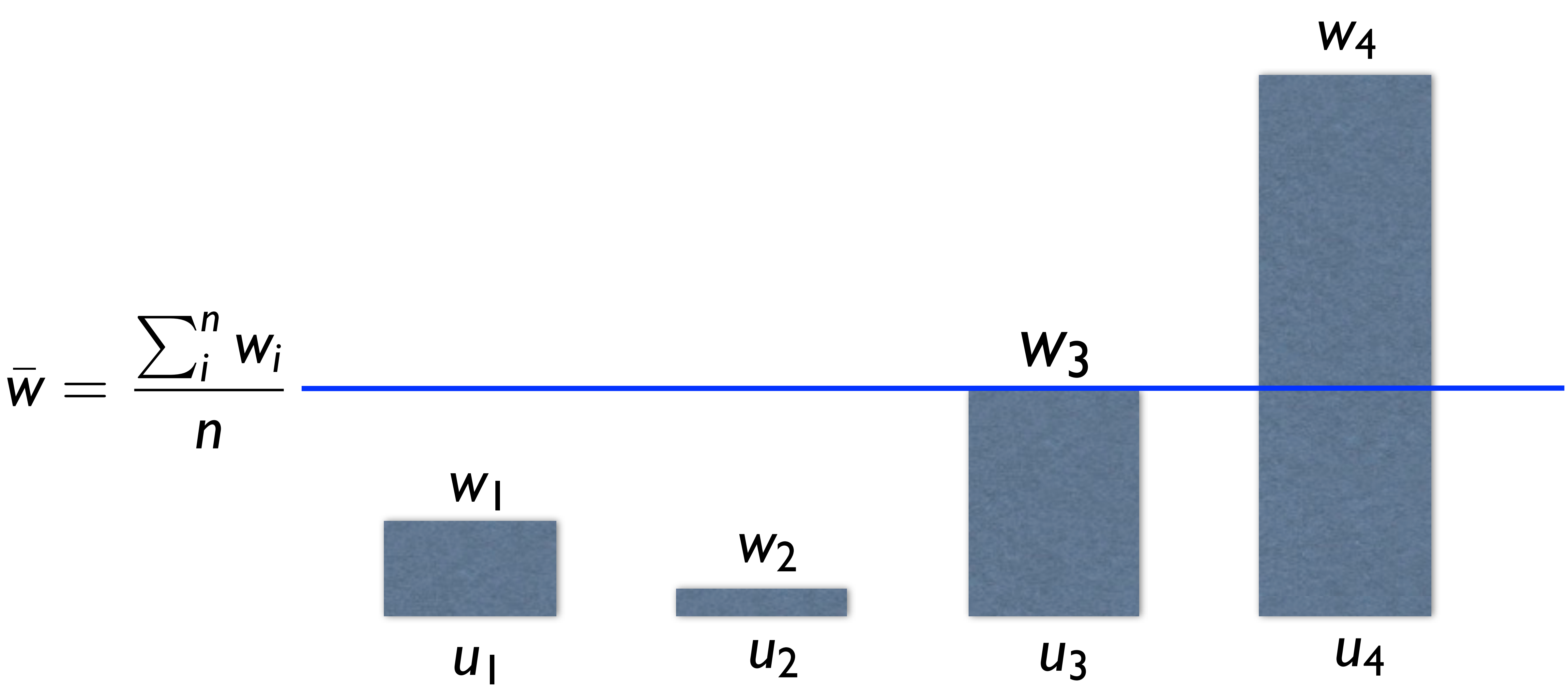
2 random calls + 2 memory lookups!

Walker, Alastair J. (1974a), Fast Generation of Uniformly Distributed Pseudorandom Numbers With Floating Point Representation, Electronics Letters, 10, 553-554.

The Walker Method



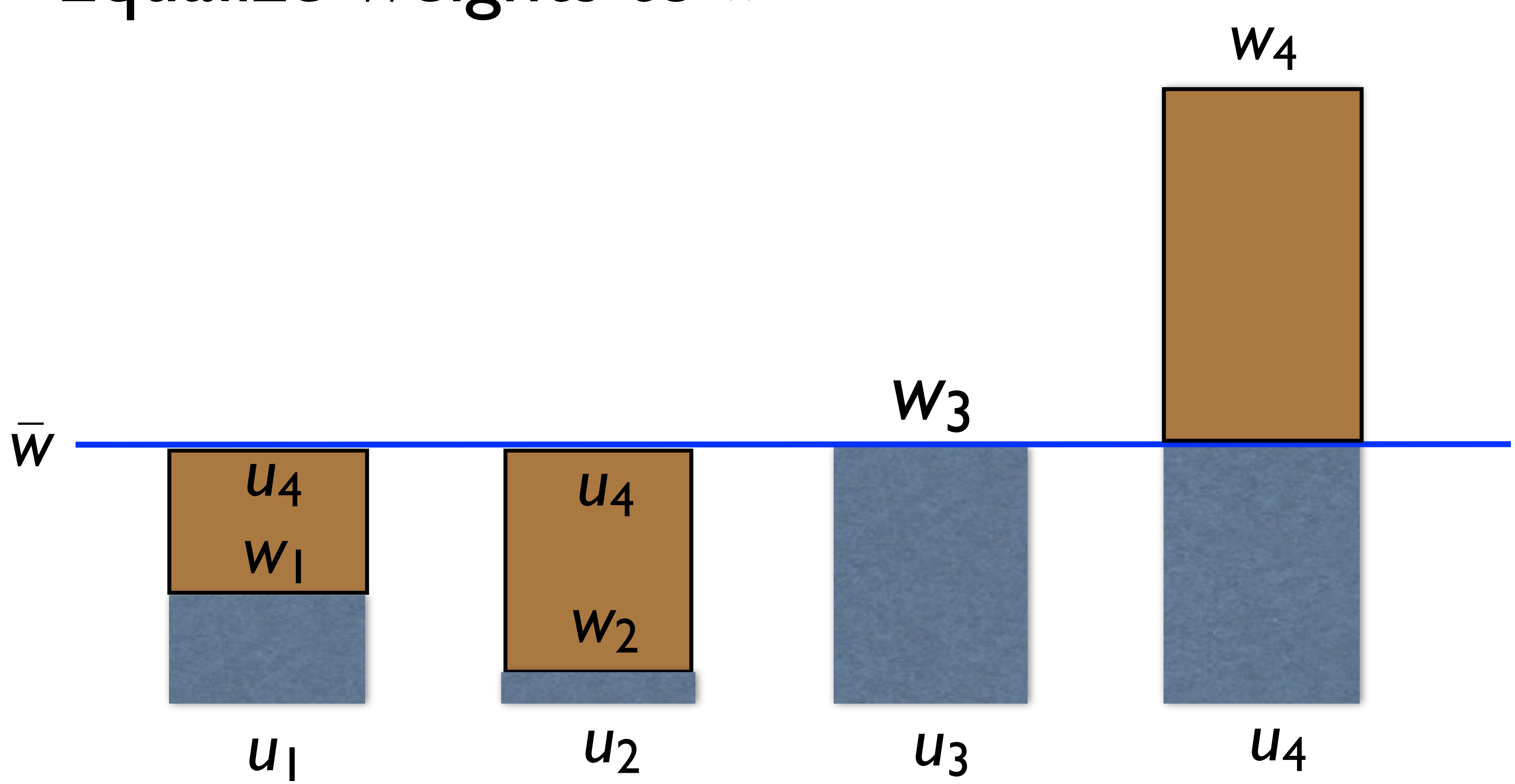
The Walker Method



The Walker Method

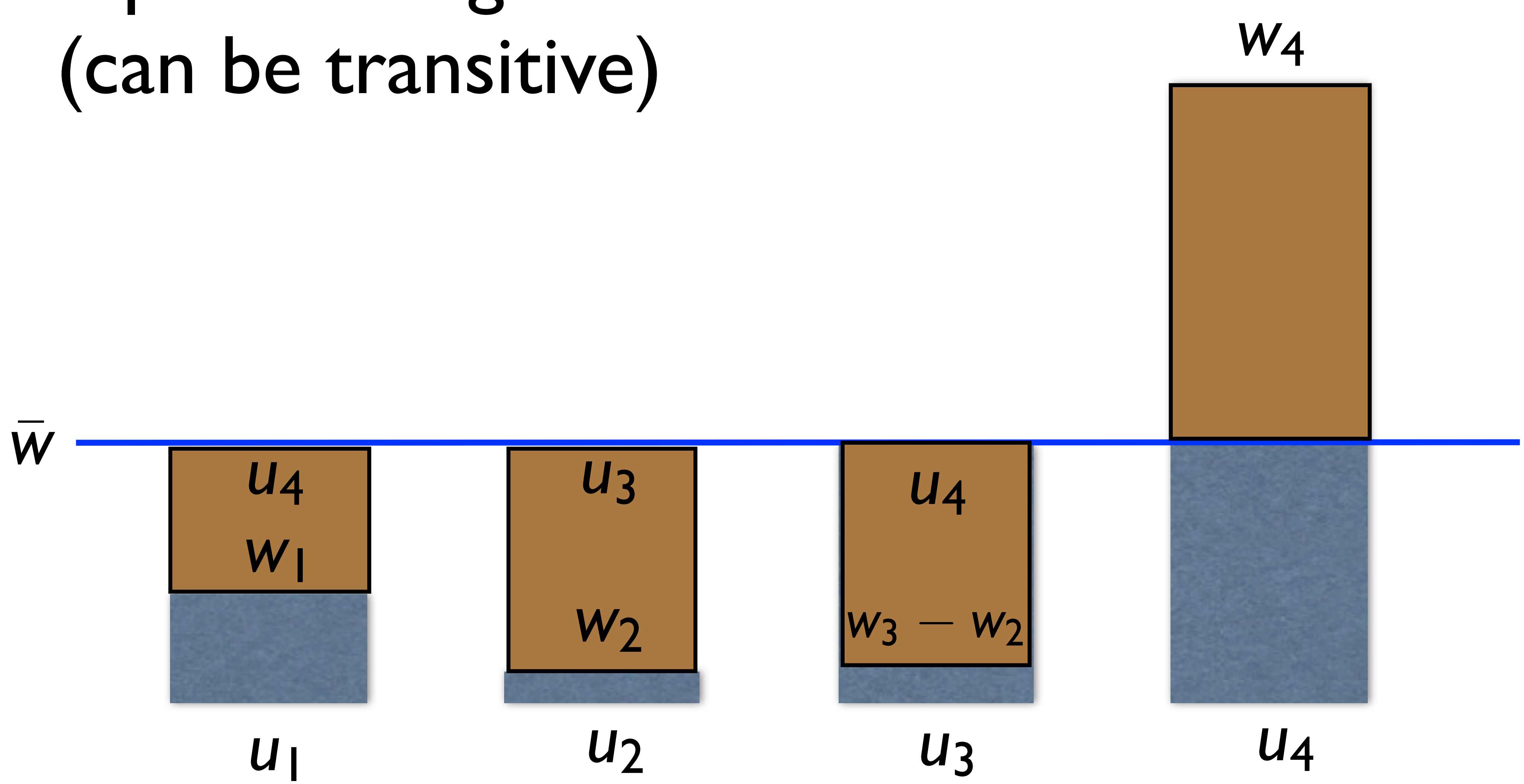
Central idea:

Equalize weights to $\bar{w}$



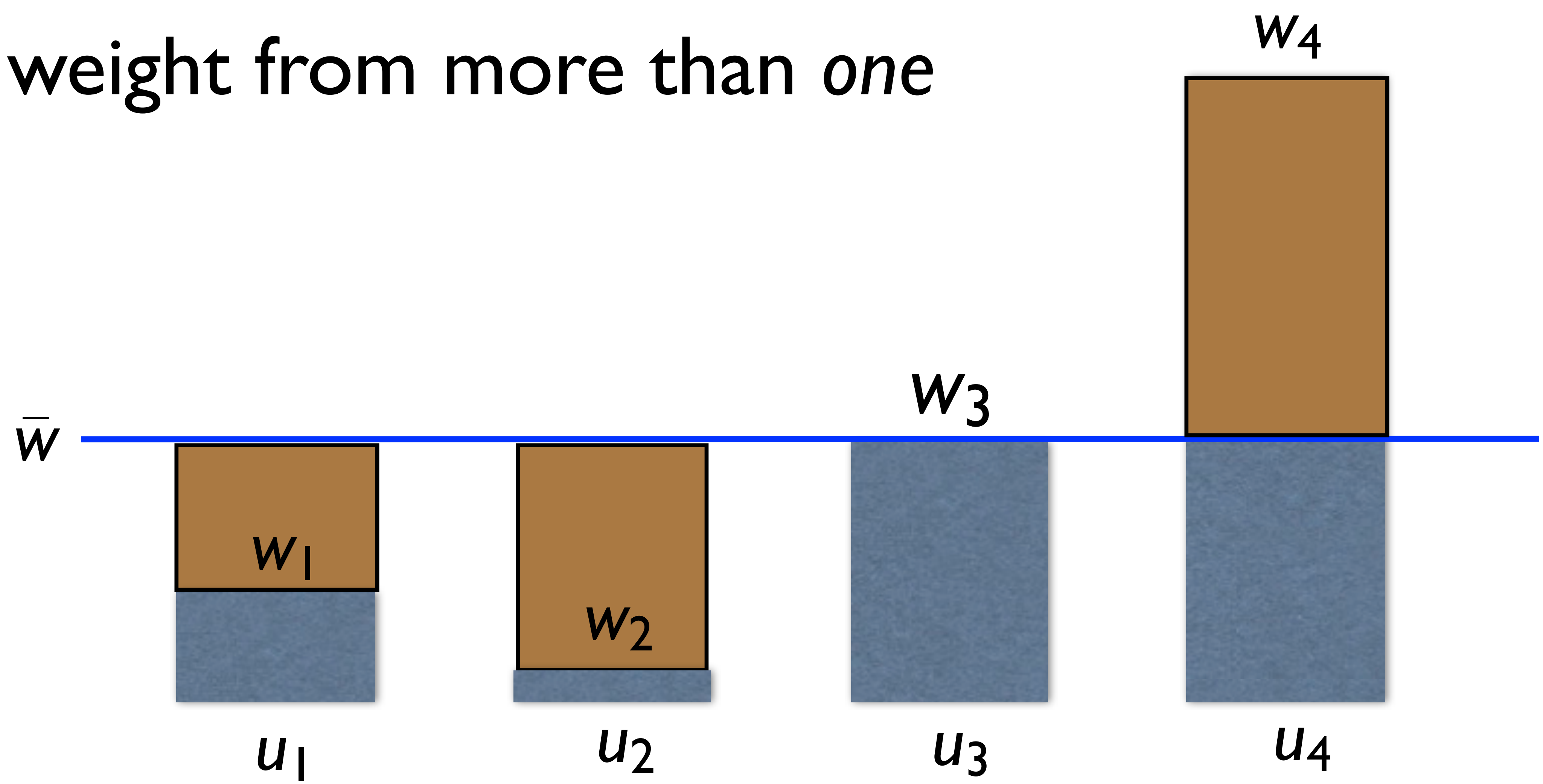
The Walker Method

Central idea:
Equalize weights to $\bar{w}$
(can be transitive)



The Walker Method

Key observation:
Never need to borrow
weight from more than *one*



The Walker Method

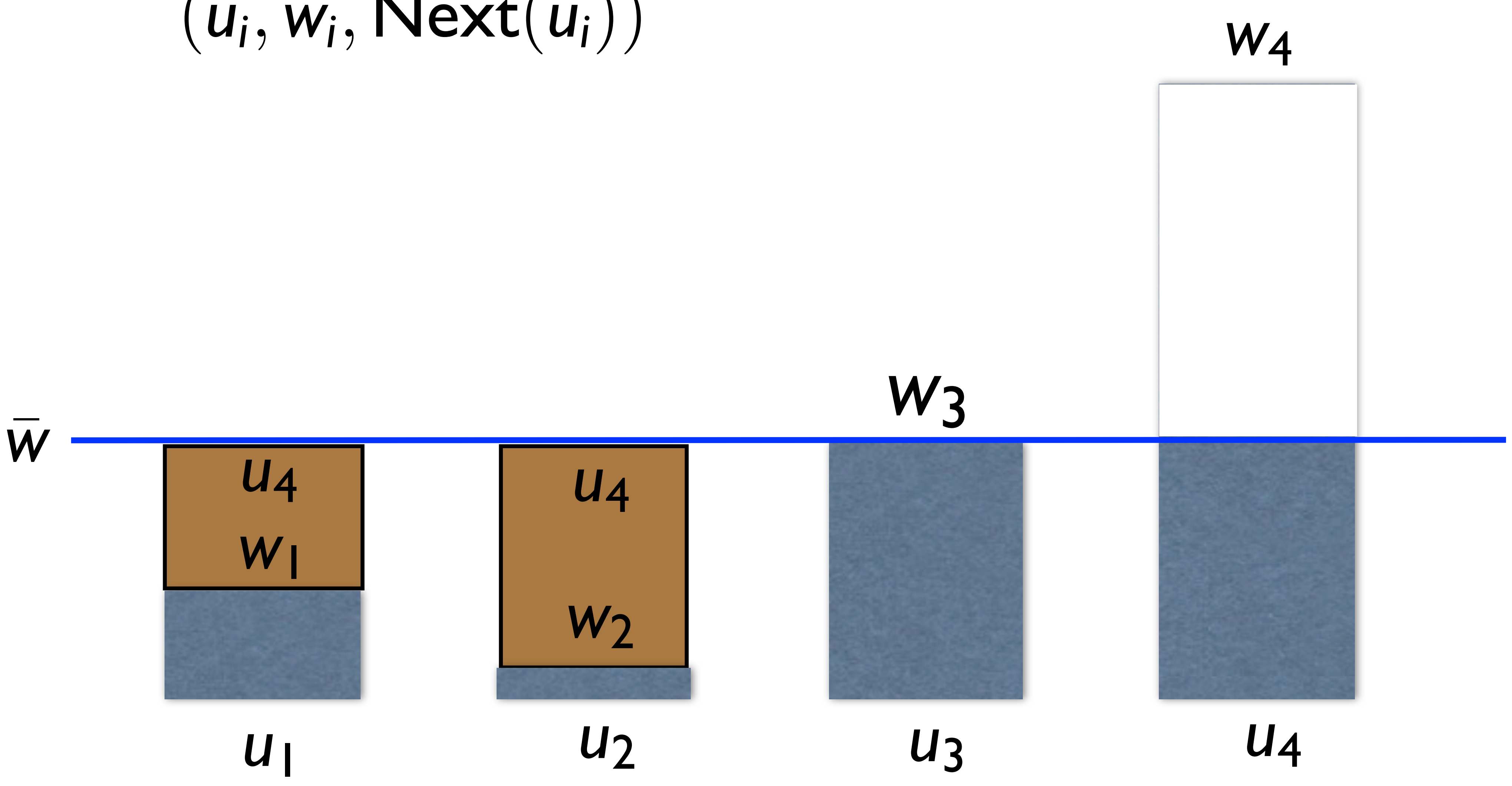
Lemma: Can equalize weights among any set of *n* elements such that each bin has at most two elements.

Proof by induction:
- Easy for two
- Suppose true for any set of *n* elements
- Pick an element below average and transfer weight from someone above average (now the latter can be below average)
- Now we are back to the *n* case!

The Walker Method

For each *i*, we have:

$$\left(u_i, w_i, \mathrm{Next}(u_i)\right)$$



The Walker Method

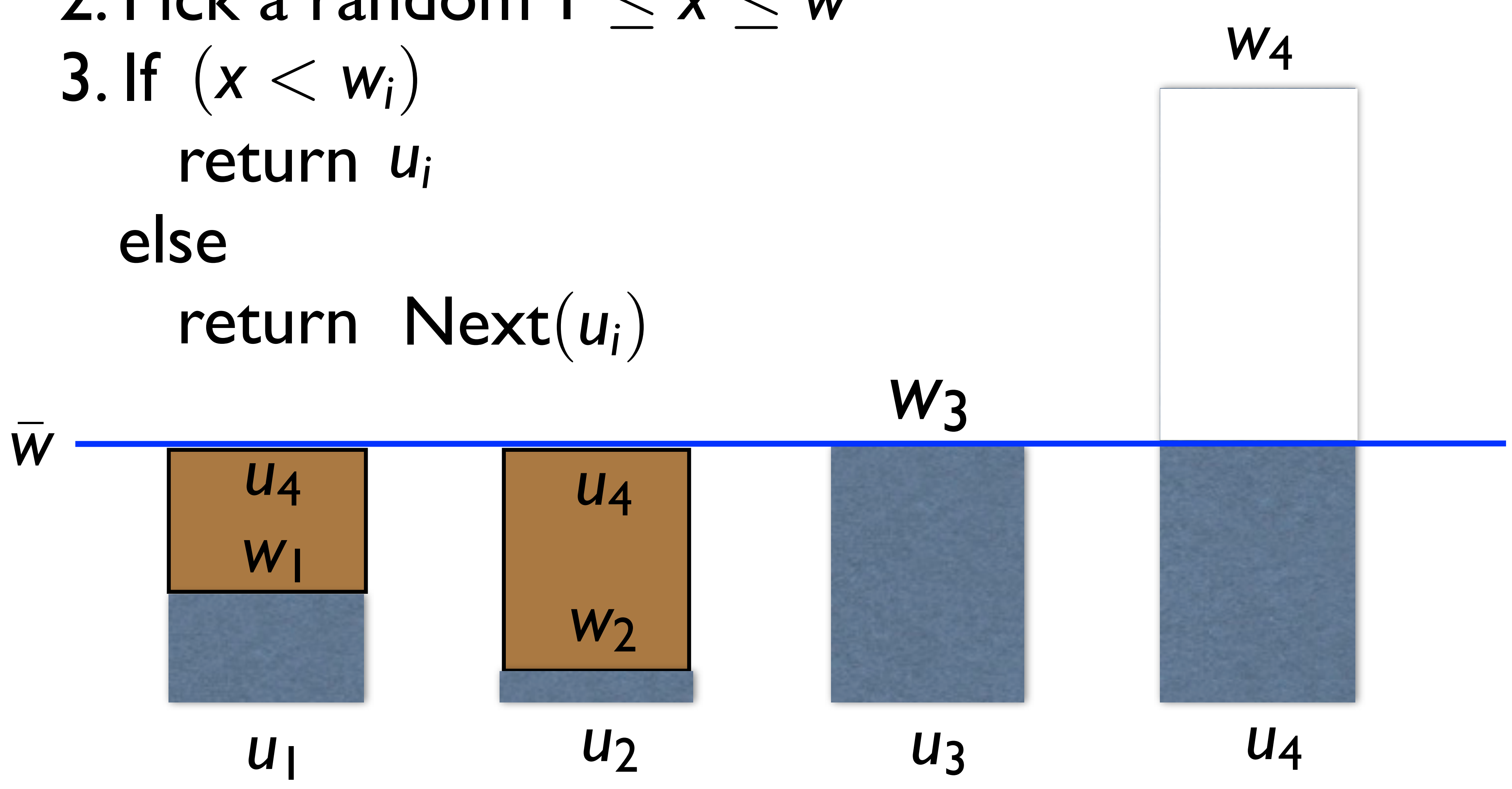
Algorithm:

1. Go to a random u_i
2. Pick a random $1 \leq x \leq \bar{w}$
3. If $(x < w_i)$

    return u_i

  else

    return $\text{Next}(u_i)$



The Walker Method

2 random calls + 2 memory lookups!

Are we done?

Engineering: $(u_i, w_i, \text{Next}(u_i))$

- (Int, Double, Int) = (4+8+4)*15*10^9 ~ 240GB
- Oops!

Summary

- Network structure is information rich
- Lots of practical applications
- Many algorithmic challenges

Thanks for listening!