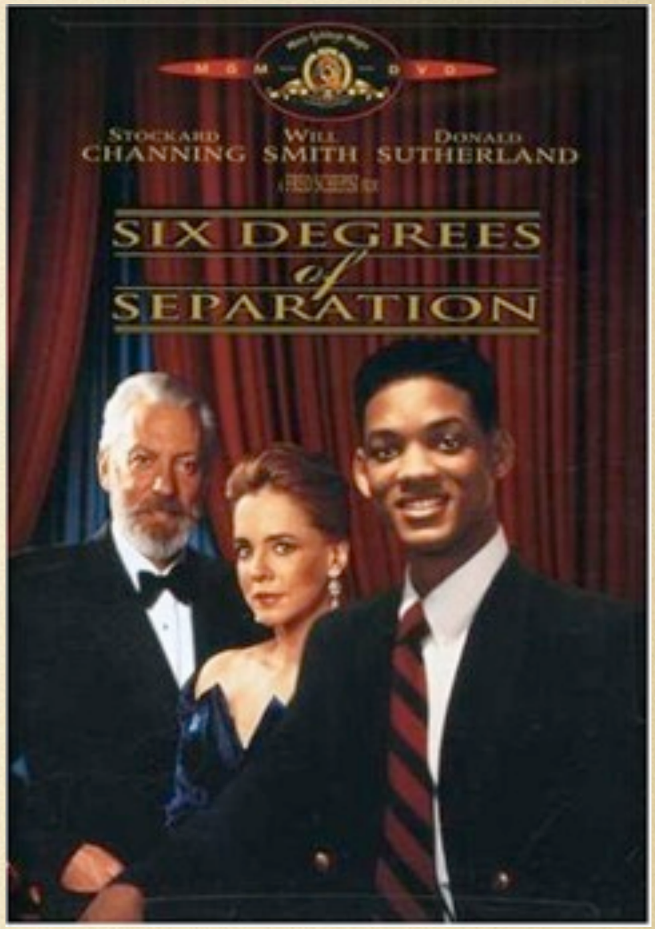
Distributed Algorithms

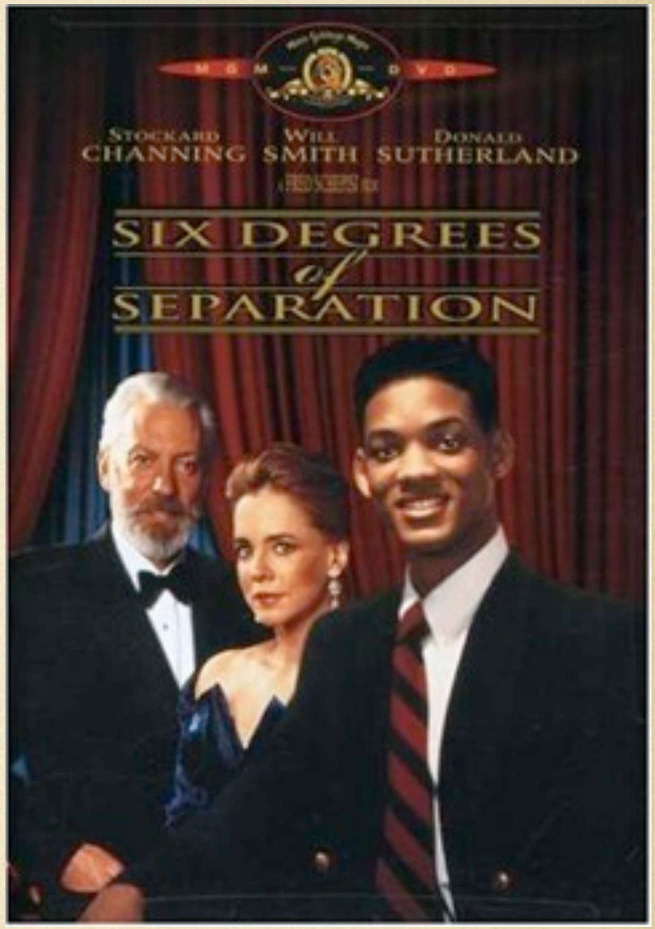
Jared Saia

6 Degrees



Ouisa Kitteridge: *“I read somewhere that everybody on this planet is separated by only six other people. Six degrees of separation between us and everyone else on this planet. The President of the United States, a gondolier in Venice, just fill in the names. I find it extremely comforting that we're so close. I also find it like Chinese water torture, that we're so close because you have to find the right six people to make the right connection.”*

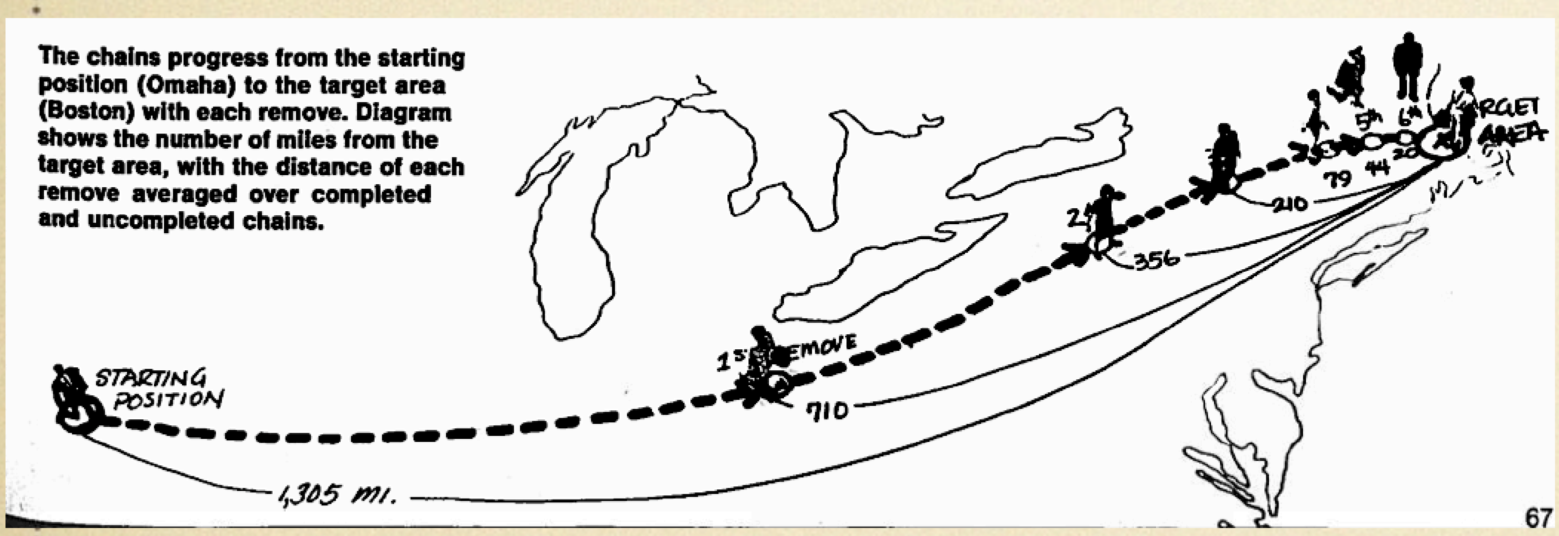
6 Degrees



Ouisa Kitteridge: *"I read somewhere that everybody on this planet is separated by only six other people. Six degrees of separation between us and everyone else on this planet. The President of the United States, a gondolier in Venice, just fill in the names. I find it extremely comforting that we're so close. I also find it like Chinese water torture, that we're so close because you have to find the right six people to make the right connection."*

Tess: *"He offered you parts in Cats? I thought you hated Cats. You said it was an all time low in a lifetime of theatre going. You said, "Aeschylus did not invent the theatre to have it end up a bunch of chorus kids in cat suits prancing around wondering which of them will go to kitty-cat heaven."*

Milgram's Experiment

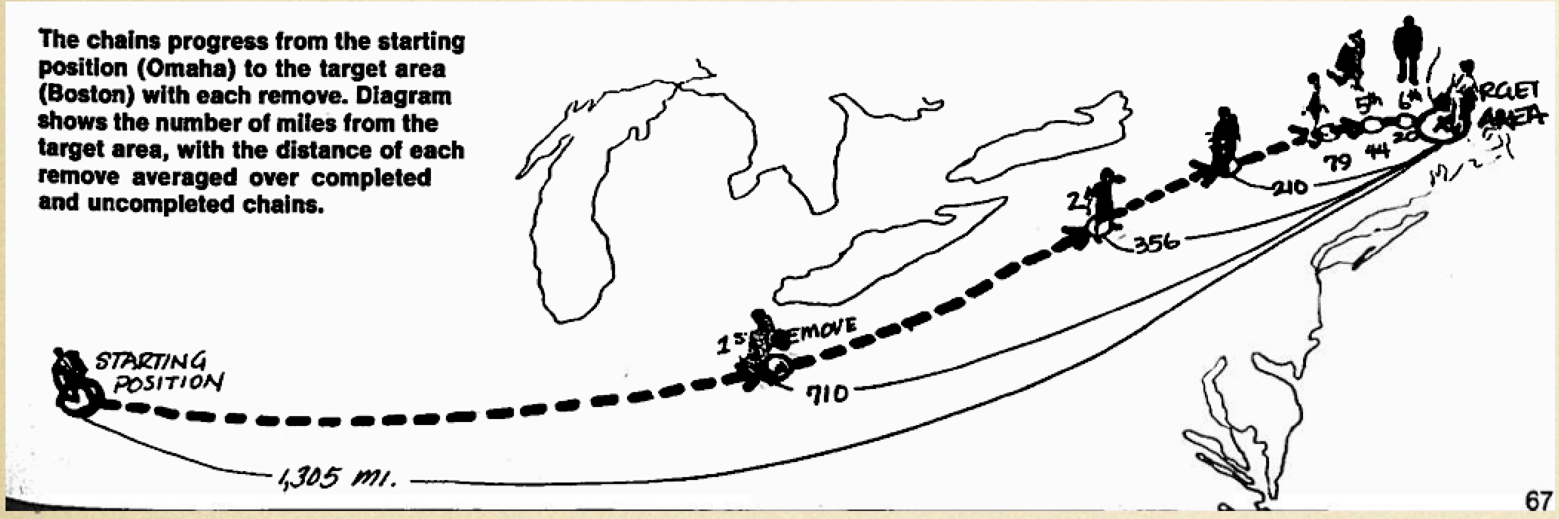


Start: 160 random people in Omaha

Target: 1 stock broker in Boston

Rule: Only send to a friend or acquaintance

Milgram's Experiment



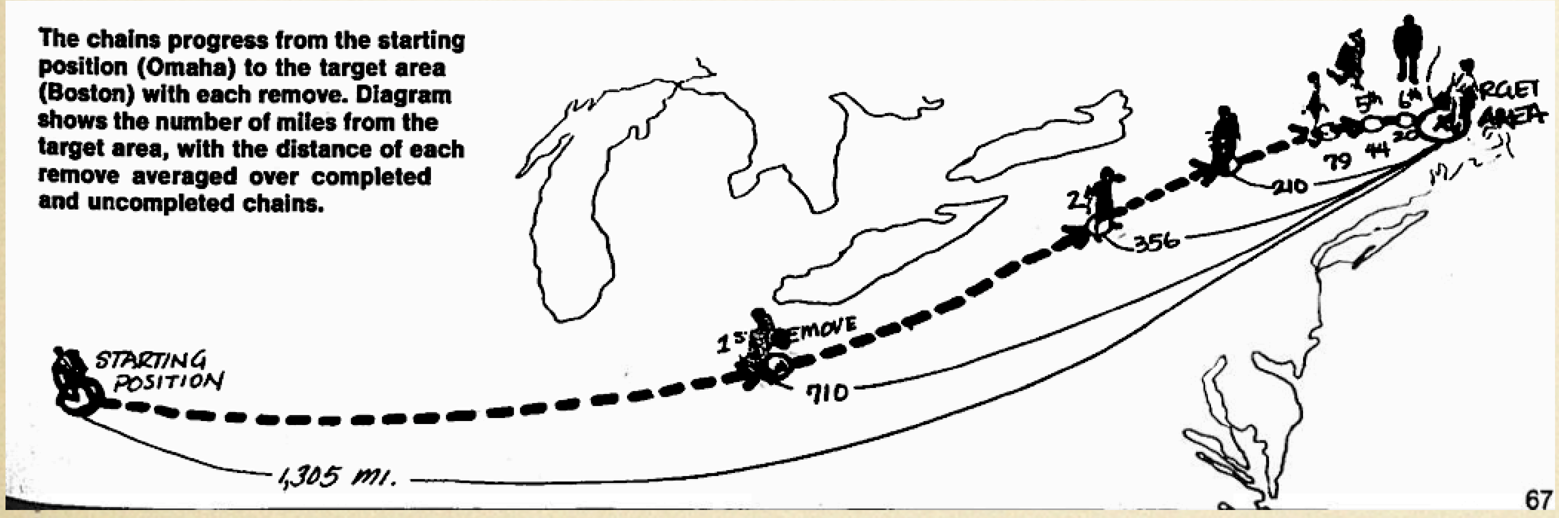
Start: 160 random people in Omaha

Target: 1 stock broker in Boston

Rule: Only send to a friend or acquaintance

Result: takes 6 hops on average to get to target

Milgram's Experiment



Start: 160 random people in Omaha

Target: 1 stock broker in Boston

Rule: Only send to a friend or acquaintance

Result: takes 6 hops on average to get to target

Recent: ~6 hops to route via email (Watts, '01)

Social Network Properties

- 1) Shortest paths are small

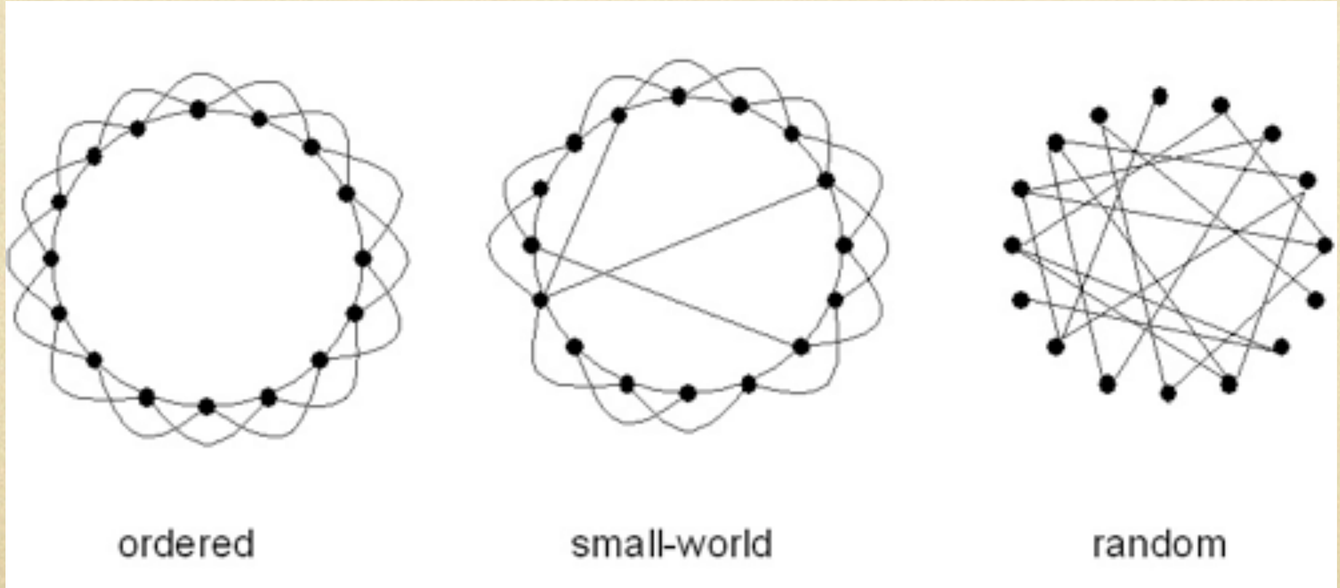
  “Six degrees of separation ... I find it extremely comforting that we're so close.”

- 2) Local Clusters

  “Keep your friends close and your enemies closer” - Machiavelli

Watts-Strogatz Model

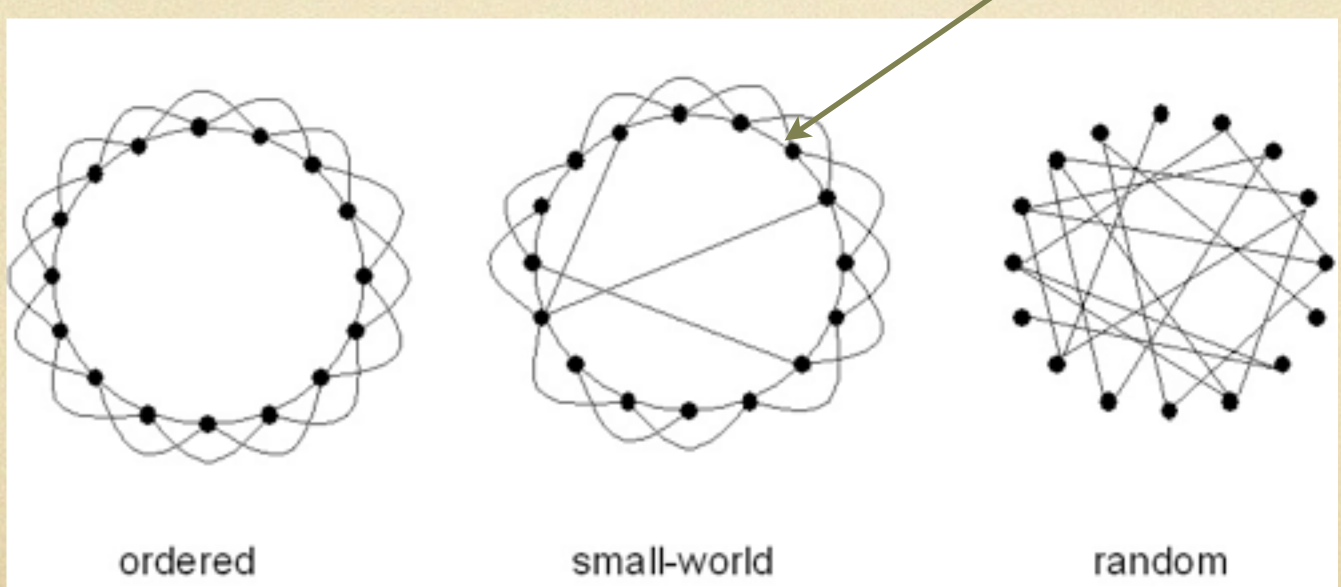
- "Small World" model ensures both:
  - Short paths (logarithmic)
  - Many clusters



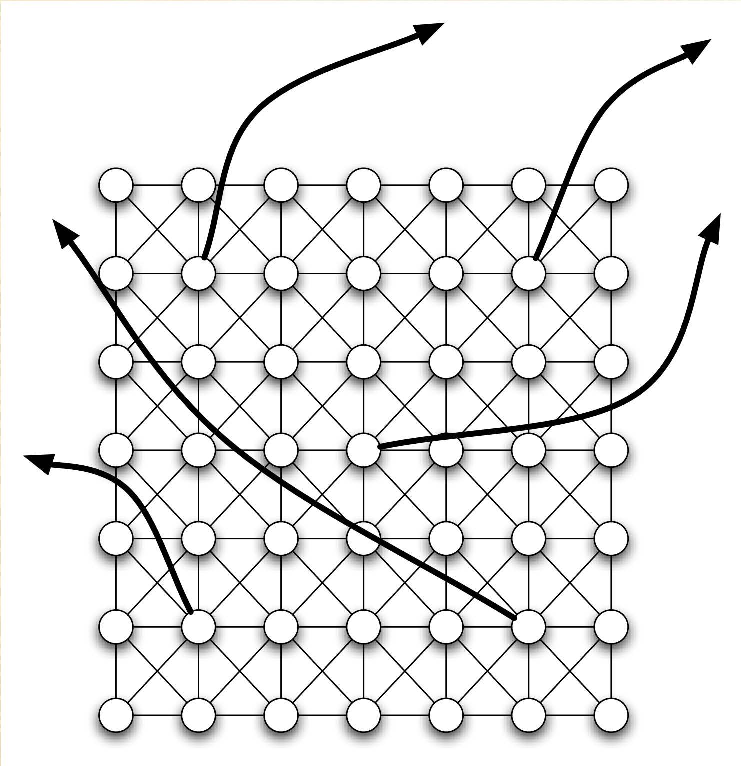
Watts-Strogatz Model

- "Small World" model ensures both:
  - Short paths (logarithmic)
  - Many clusters

Small World is ordered + random

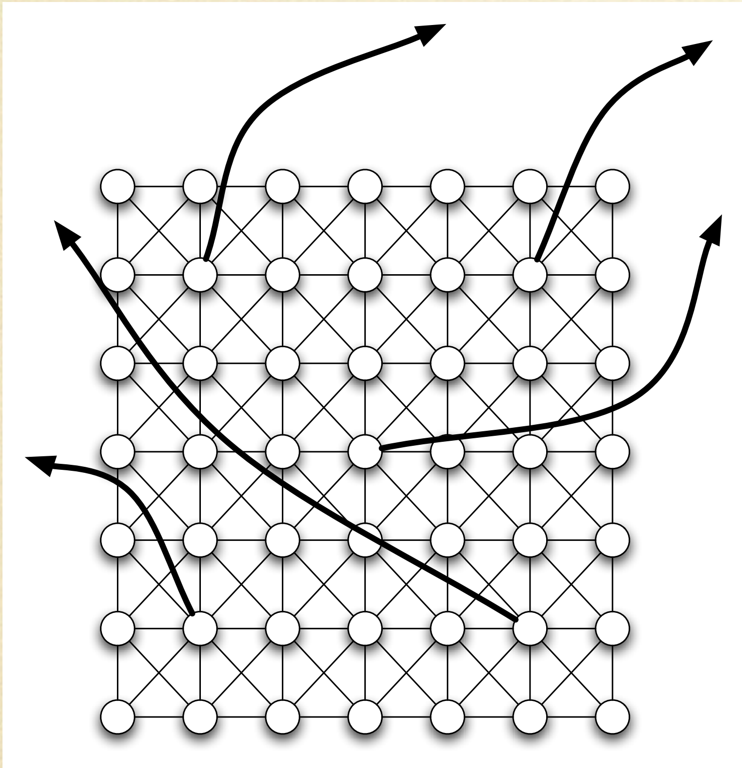


Watts-Strogatz



1) **ordered** links: neighbors in grid
2) **random** links: to random node in grid
Each node has one random link

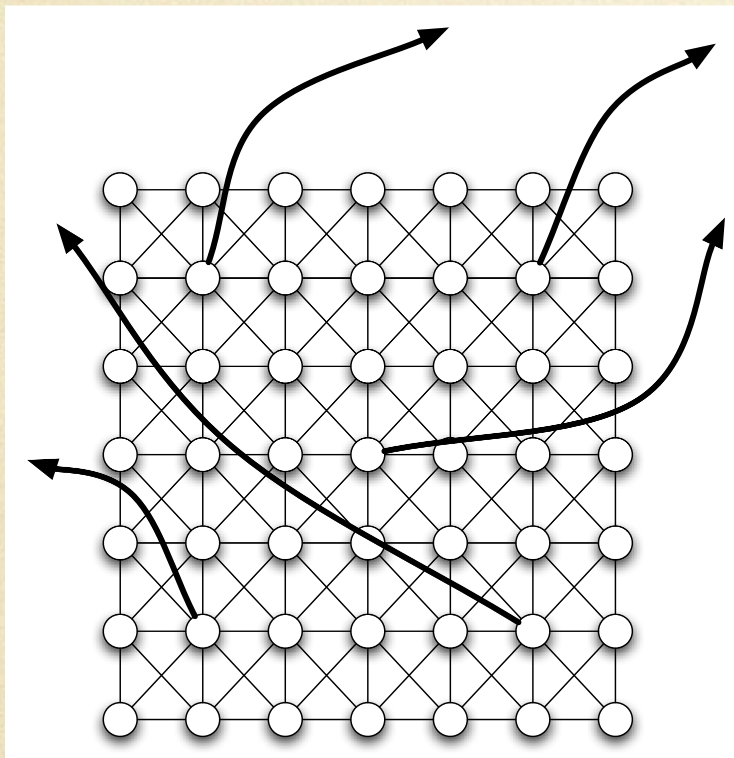
Watts-Strogatz



1) **ordered** links: neighbors in grid
2) **random** links: to random node in grid
Each node has one random link

Clear that: 1) Many local clusters;
Can show: 2) All distances at most logarithmic.

Watts-Strogatz



1) **ordered** links: neighbors in grid
2) **random** links: to random node in grid
Each node has one random link

Clear that: 1) Many local clusters;
Can show: 2) All distances at most logarithmic.

Node selected uniformly at random

A Problem

*"Six degrees of separation ... I find it extremely comforting that we're so close... I also find it like Chinese water torture, that we're so close because **you have to find the right six people to make the right connection.**"*

Knowing there **exist** six people is very different than **finding** those six people

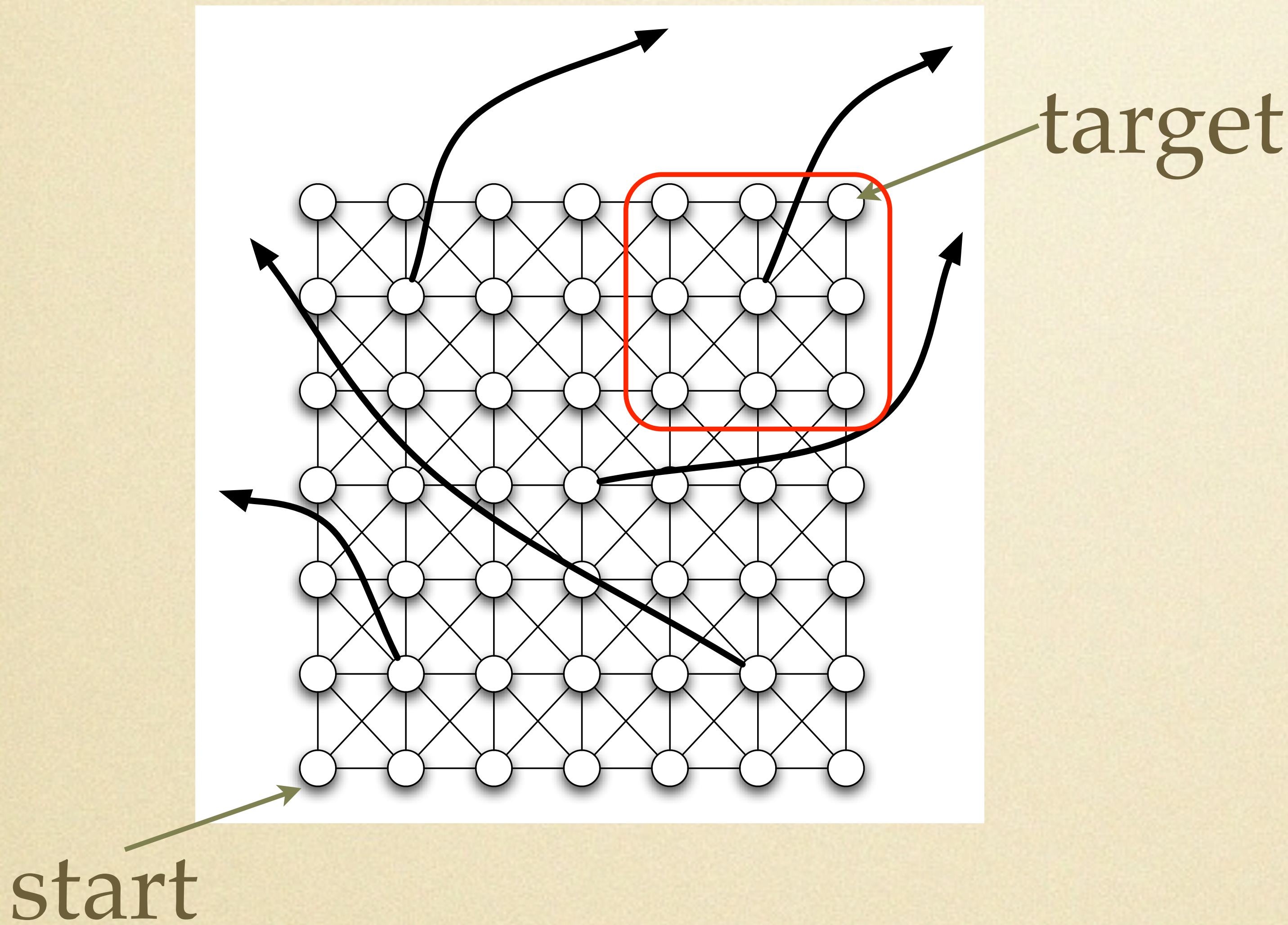
A Problem

*"Six degrees of separation ... I find it extremely comforting that we're so close... I also find it like Chinese water torture, that we're so close because **you have to find the right six people to make the right connection.**"*

Knowing there **exist** six people is very different than **finding** those six people

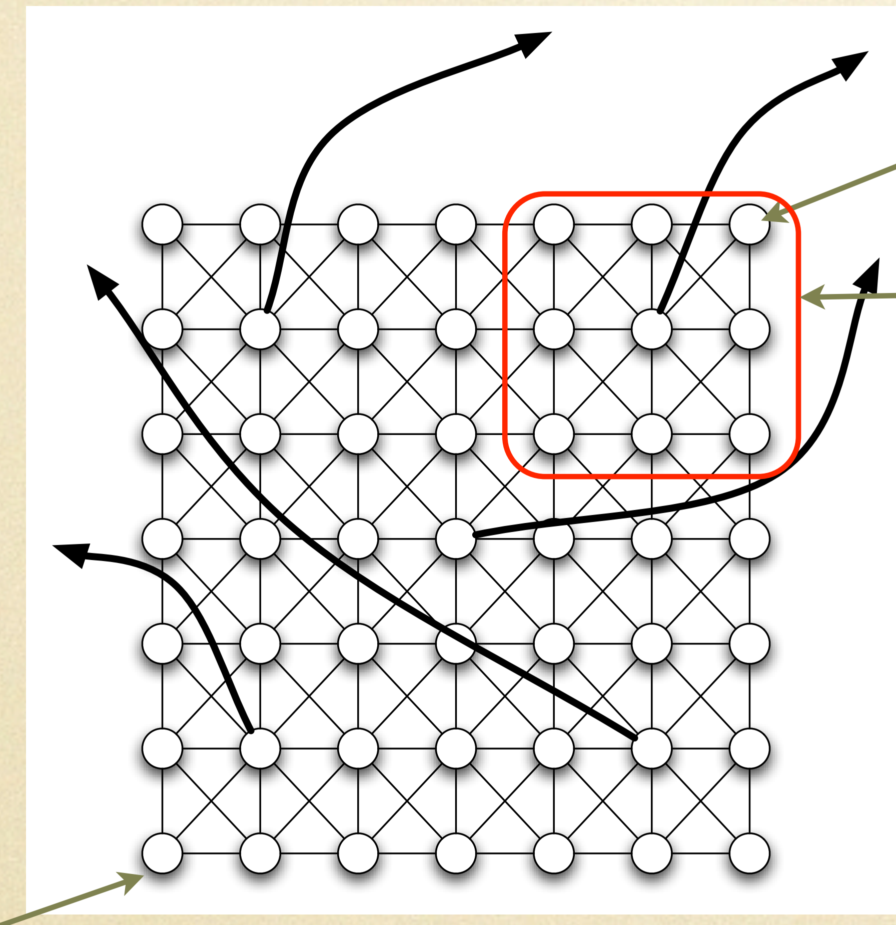
In fact, Watts-Strogatz is wrong! It doesn't account for **finding** the six people.

A Problem



n nodes in grid

A Problem



target

Region containing $\sqrt{n}$ nodes

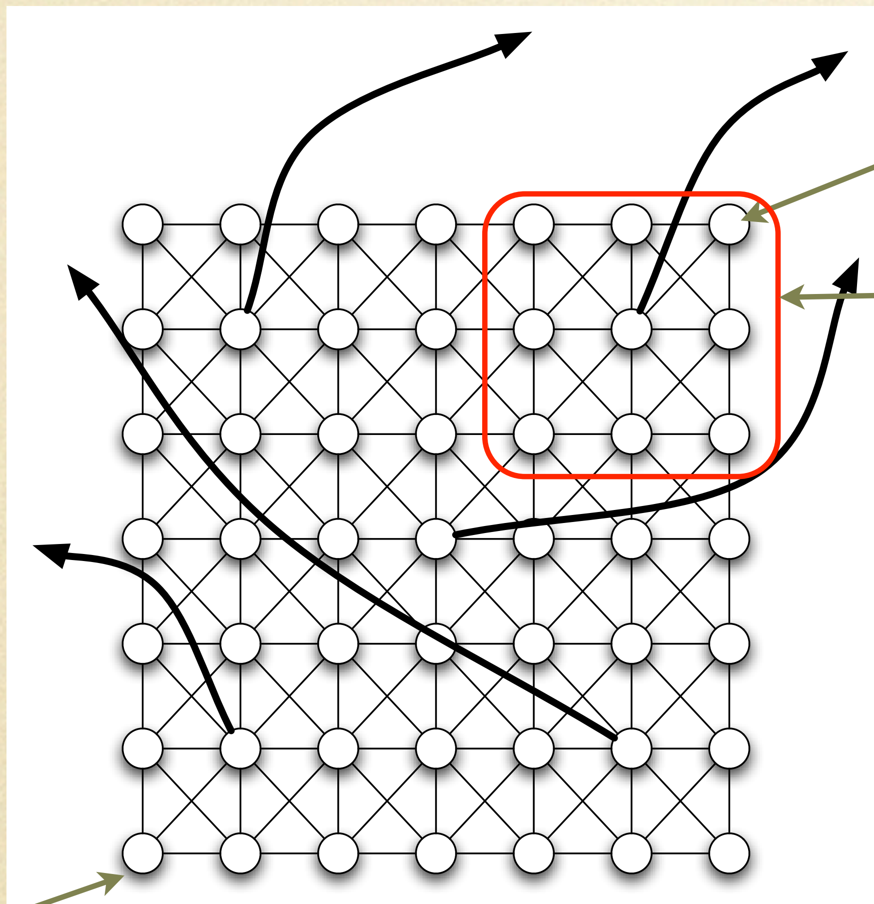
n nodes in grid

start

A Problem

n nodes in grid

target



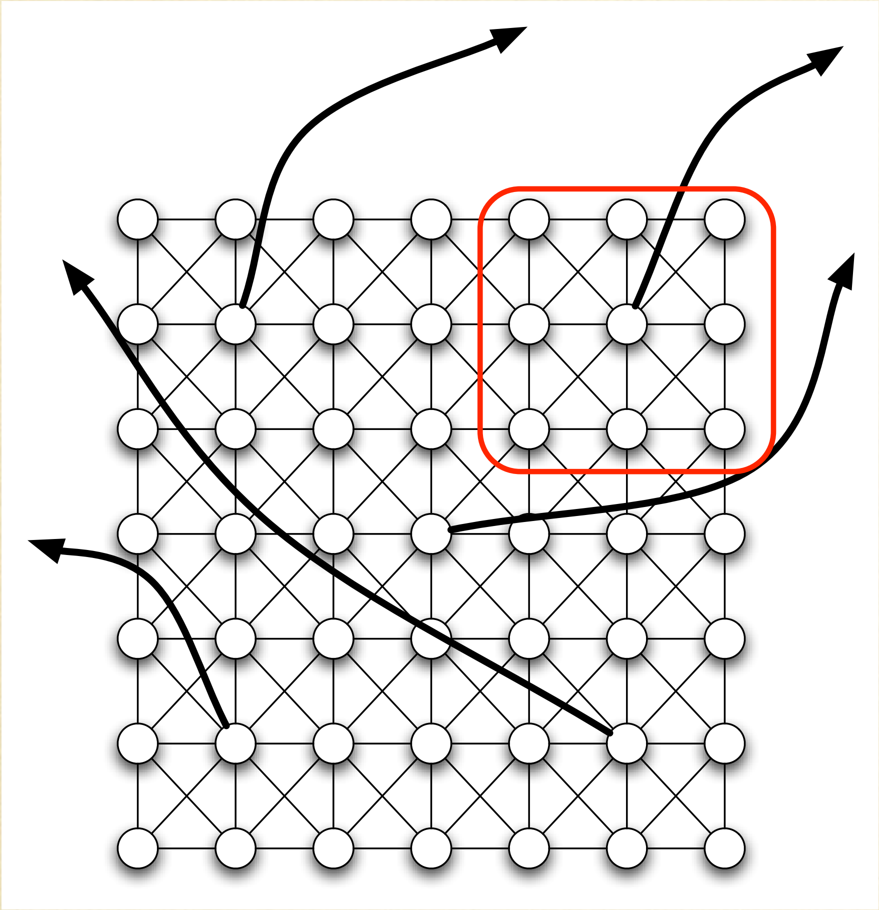
Region containing

√n nodes

start

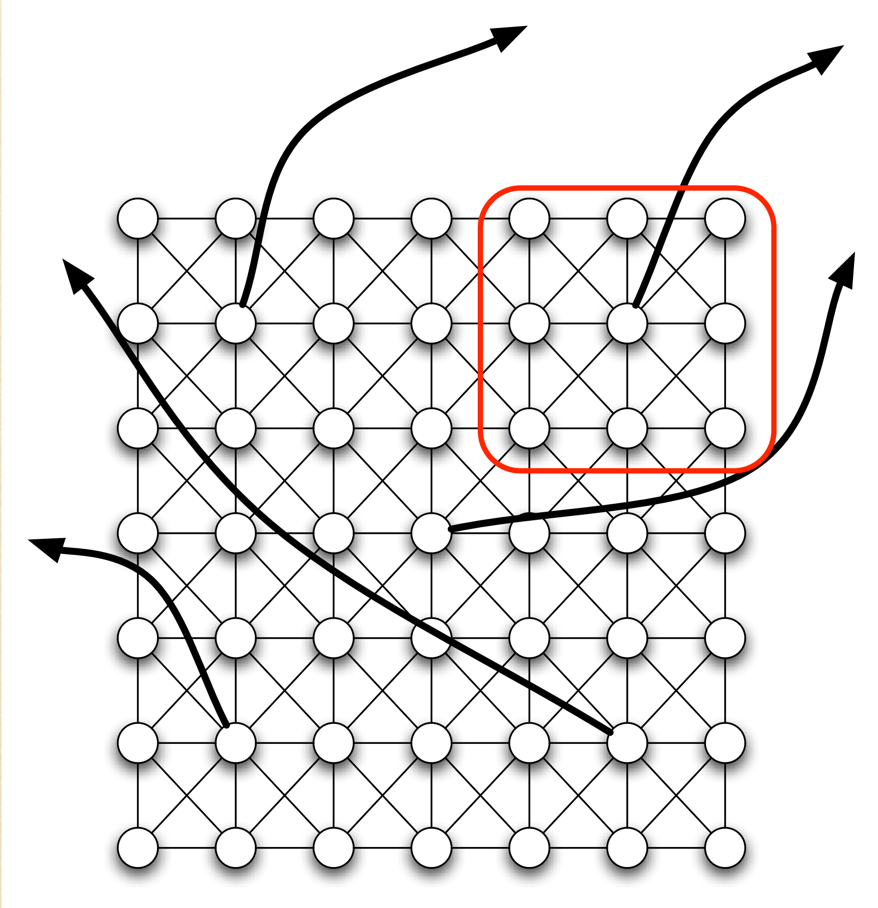
Q: What is expected time to get to this red square?

A Problem



Q: What is expected time to get to this red square?

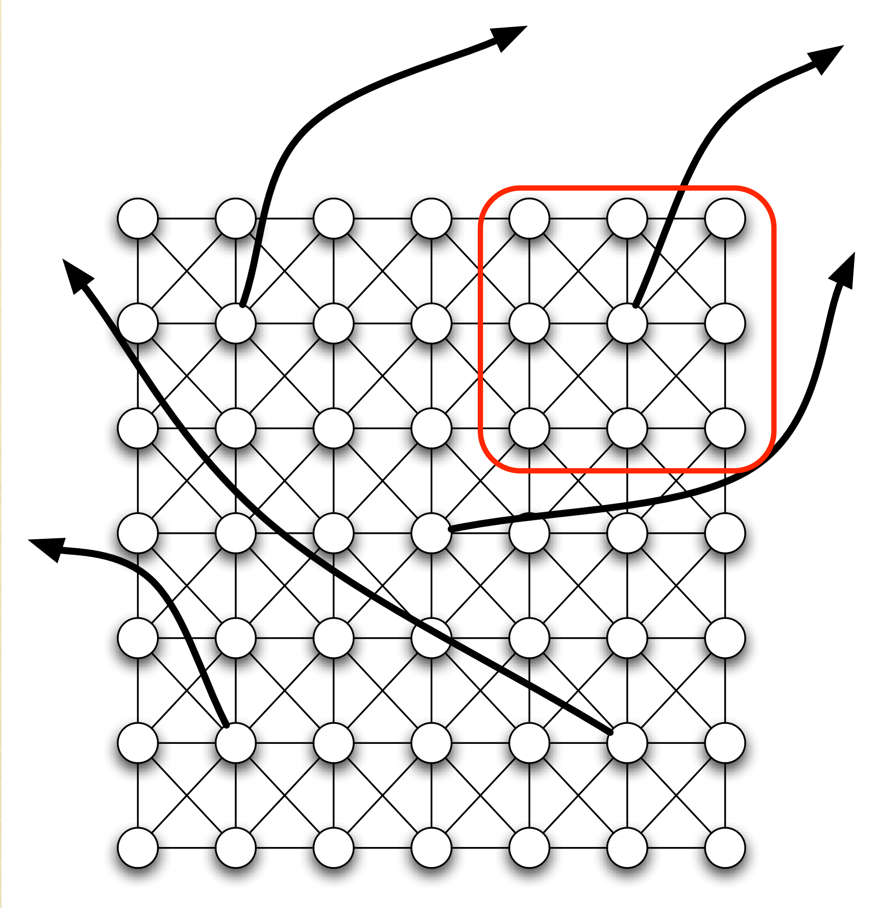
A Problem



Q: What is expected time to get to this red square?

Using short links alone requires √n hops

A Problem

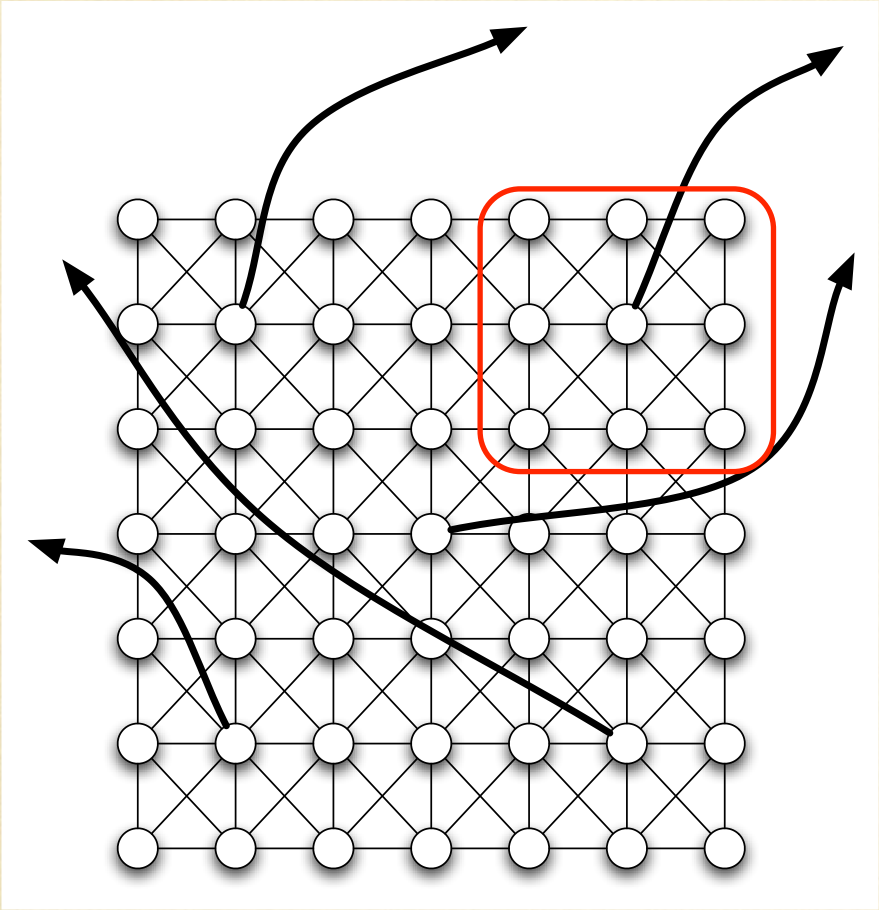


Q: What is expected time to get to this red square?

Using short links alone requires √n hops

A long link has prob. 1/√n of falling in red square

A Problem



Q: What is expected time to get to this red square?

Using short links alone requires $\sqrt{n}$ hops

A long link has prob. $1/\sqrt{n}$ of falling in red square

Expect to have to visit $\sqrt{n}$ nodes before finding a long link which falls in red square!

A Problem

Expect to have to visit $\sqrt{n}$ nodes before finding a long link which falls in red square!

A Problem

Expect to have to visit $\sqrt{n}$ nodes before finding a long link which falls in red square!

307 million people in the United States

A Problem

Expect to have to visit $\sqrt{n}$ nodes before finding a long link which falls in red square!

307 million people in the United States

$\sqrt{307}$ million is about 17,500

A Problem

Expect to have to visit $\sqrt{n}$ nodes before finding a long link which falls in red square!

307 million people in the United States

$\sqrt{307}$ million is about 17,500

Need much quicker routing!!!

Kleinberg Model

1) **ordered** links: neighbors in grid
2) **random** links: to random node in grid
Each node has one random link

Watts-Strogatz:
Node selected uniformly at random

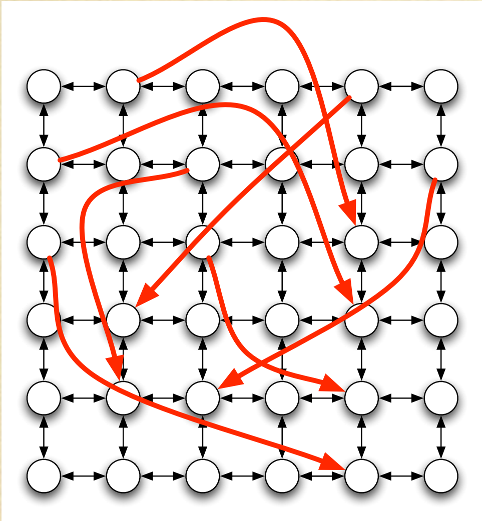
Kleinberg Model

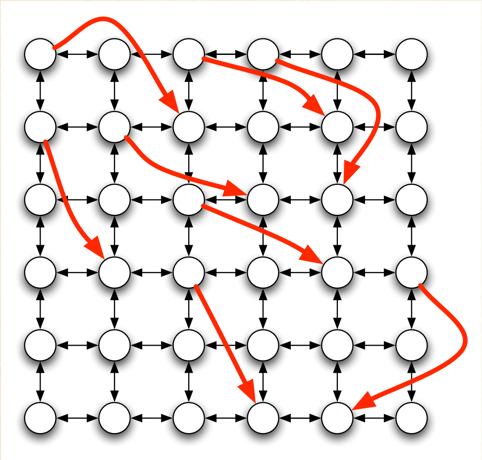
1) **ordered** links: neighbors in grid
2) **random** links: to random node in grid
Each node has one random link

Watts-Strogatz: Node selected uniformly at random

Kleinberg: Node x selected with probability $\propto 1/(\text{distance to } x)^2$

Kleinberg Model





1) **ordered** links: neighbors in grid
2) **random** links: to random node in grid
Each node has one random link

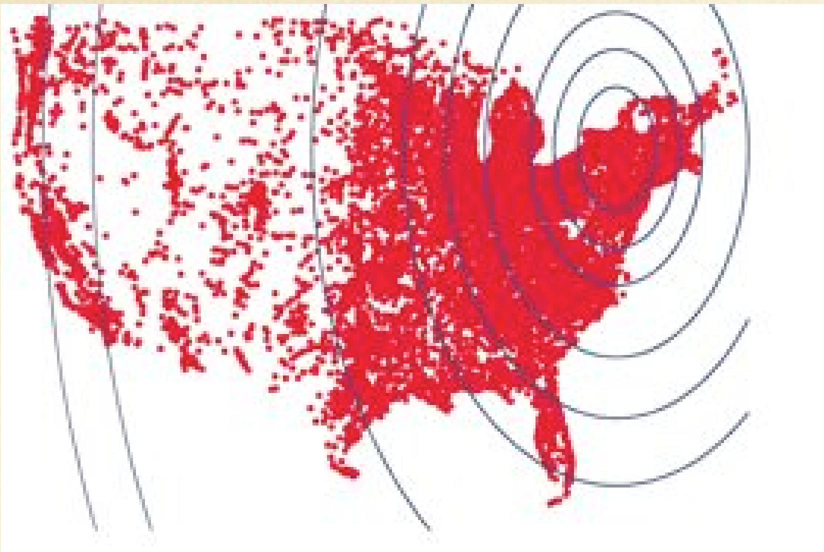
Watts-Strogatz: Node selected uniformly at random

Kleinberg: Node x selected with probability $\propto 1/(\text{distance to x})^2$

Kleinberg

- Result: In Kleinberg model, can route from any start node to any goal node in essentially $\log^2 n$ hops!

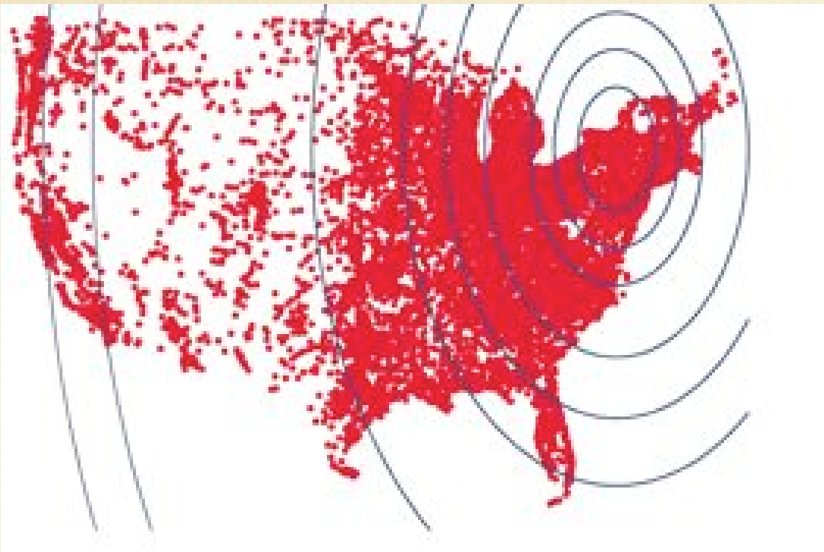
Data



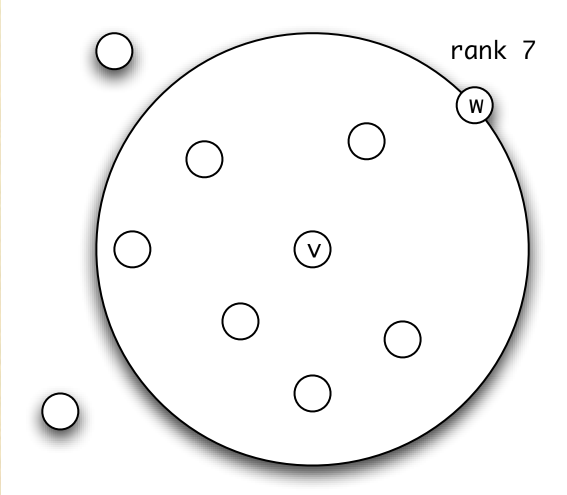
Population density of LiveJournal network (Liben-Nowell et al. '05)

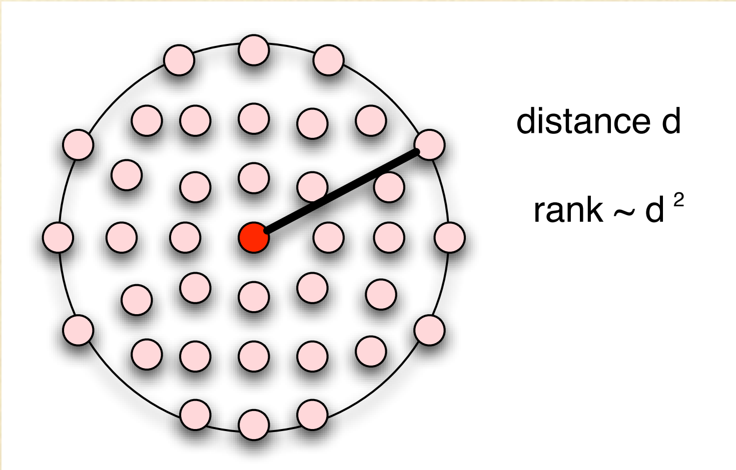
Rank addresses variations in population density

Data



Population density of LiveJournal network (Liben-Nowell et al. '05)

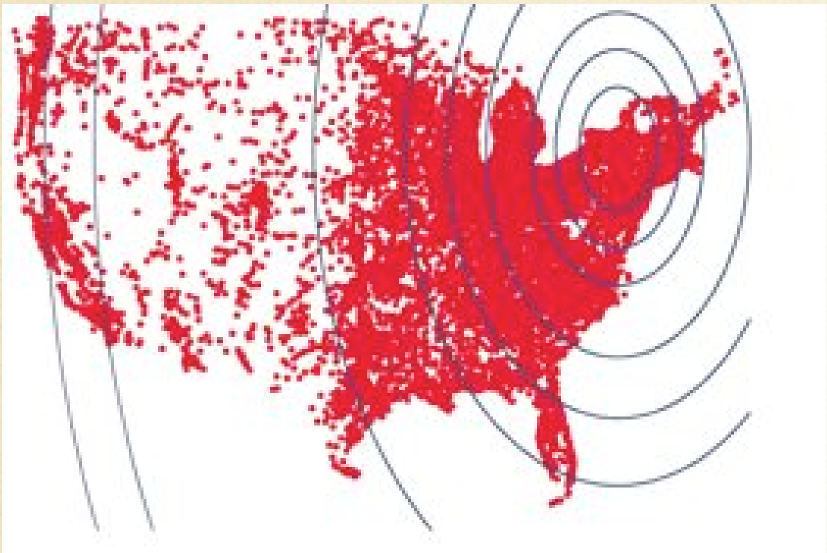




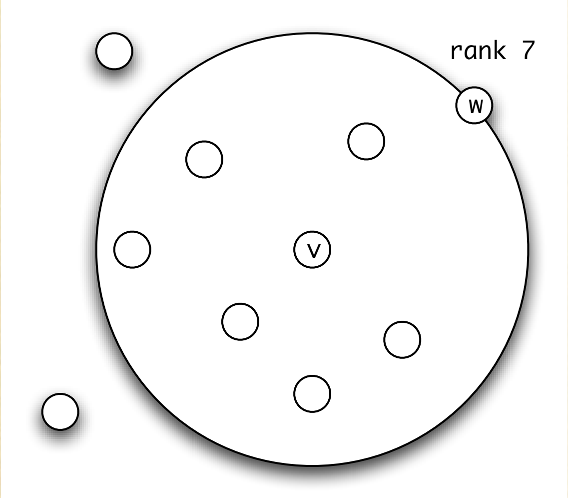
Rank addresses variations in population density

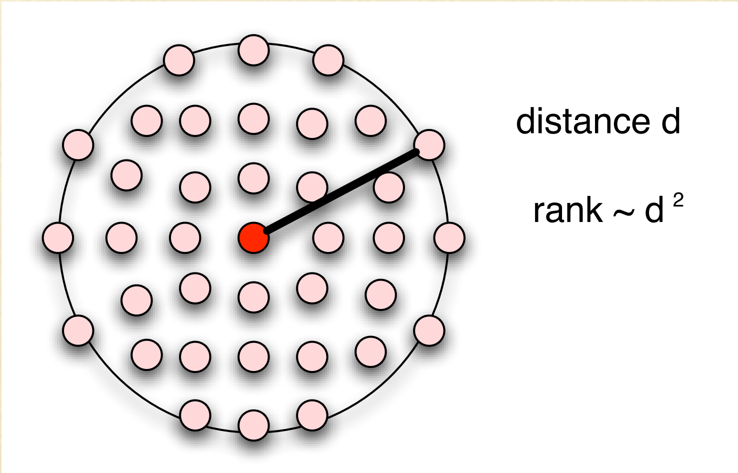
General Case: prob. of link to node w/ rank r is $\propto 1/r$

Data



Population density of LiveJournal network (Liben-Nowell et al. '05)



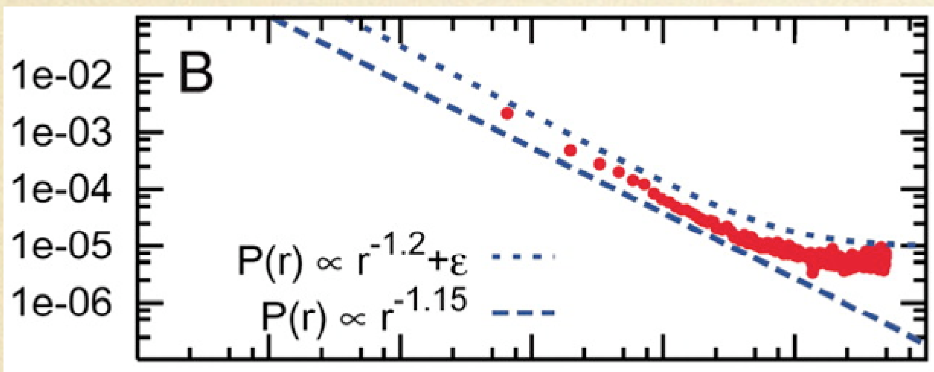


Rank addresses variations in population density

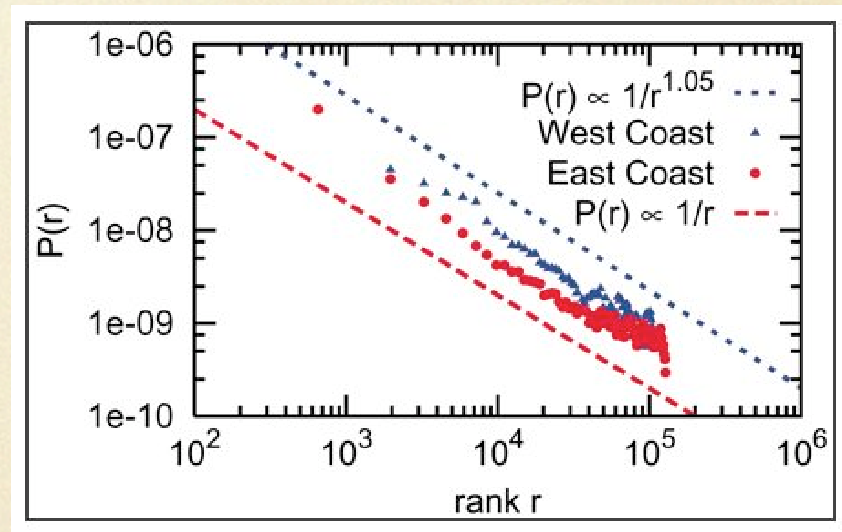
General Case: prob. of link to node w/ rank r is $\propto 1/r$

Acknowledgement: Many of the figures in this talk are from the book *Networks, Crowds and Markets: Reasoning about a Highly Connected World* by David Easley and Jon Kleinberg

Data



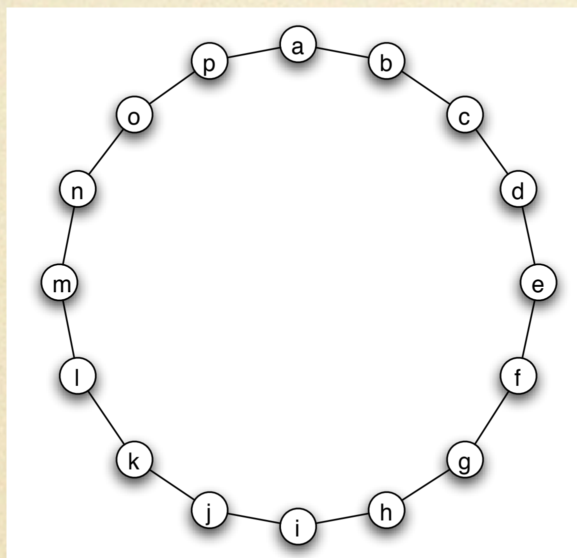
(a) *Rank-based friendship on LiveJournal*

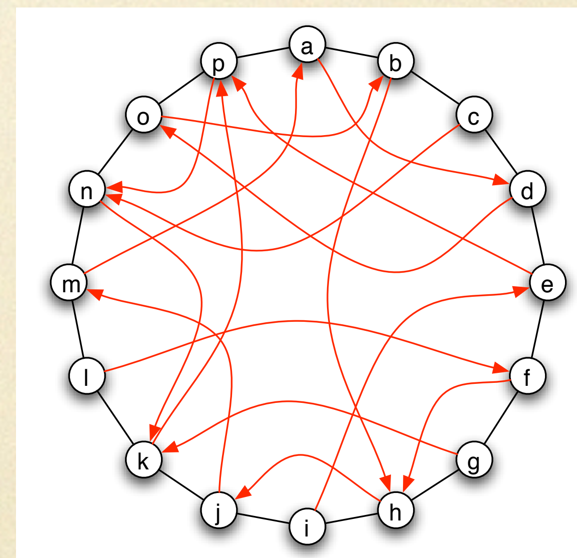


(b) *Rank-based friendship: East and West coasts*

Observed probability fits very close to 1/r

Ring

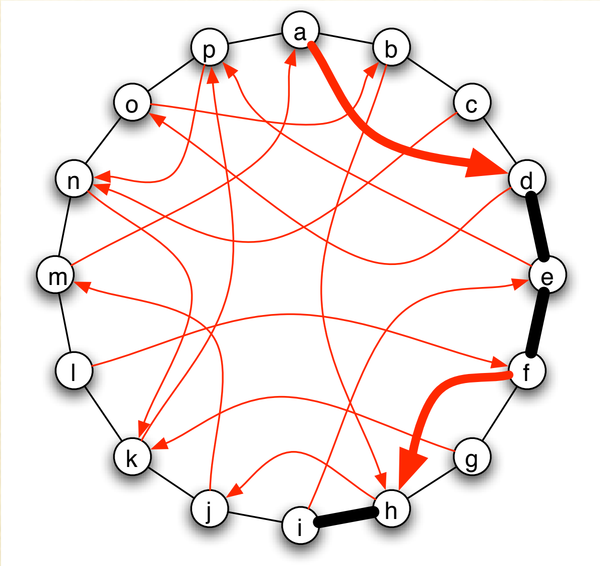




Easier to do analysis on a ring (but same techniques work for a grid)

Random link to x will now happen with probability $\propto 1/(\text{distance to } x)$

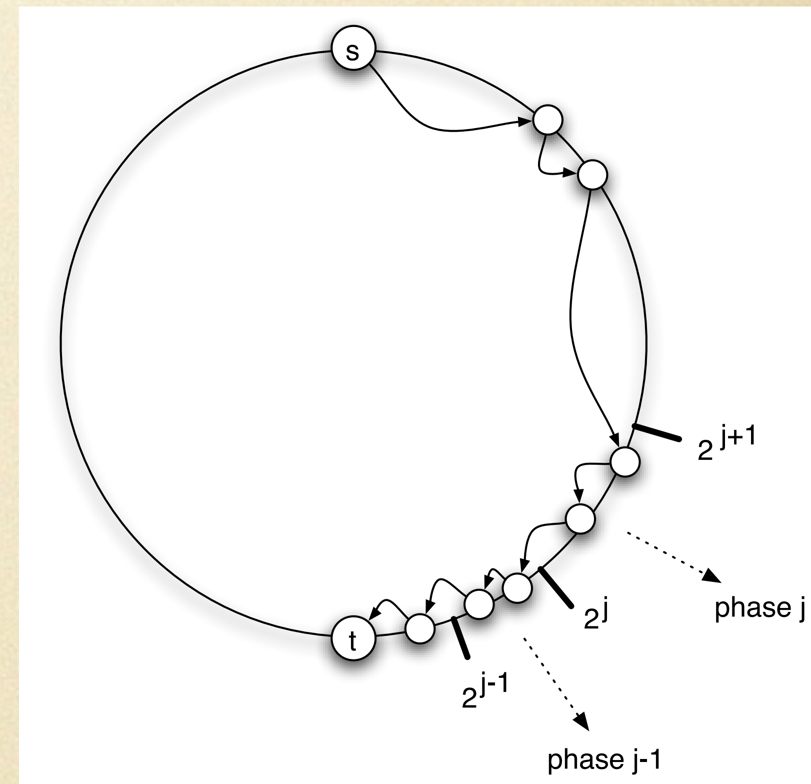
Ring vs Grid



Algorithm: Current message holder forwards message to person it knows who is closest to target

Analysis

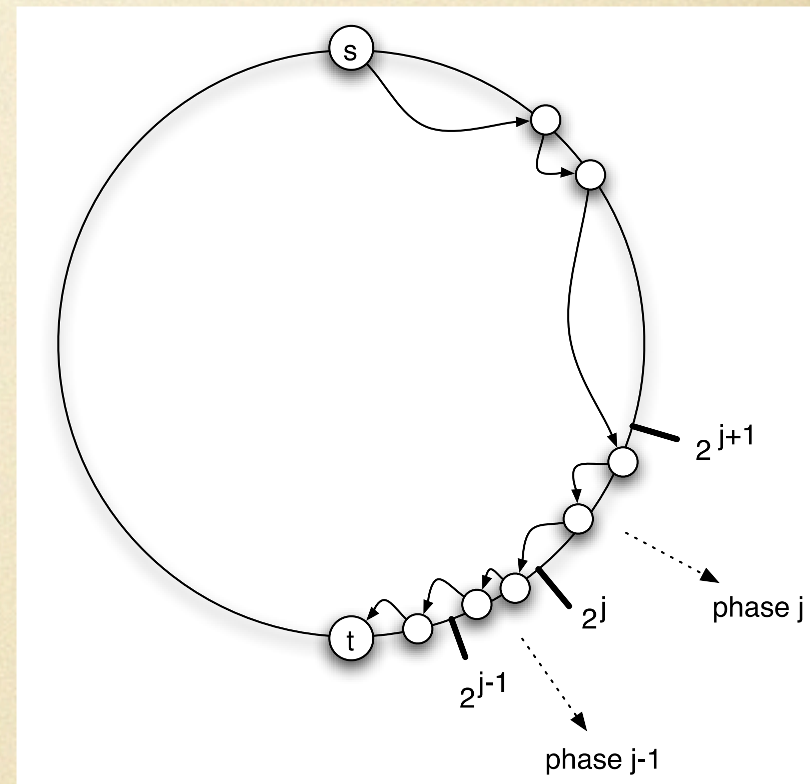
We'll say we're in **phase j** of the algorithm when distance from target is between 2^j and 2^{j-1}



Analysis

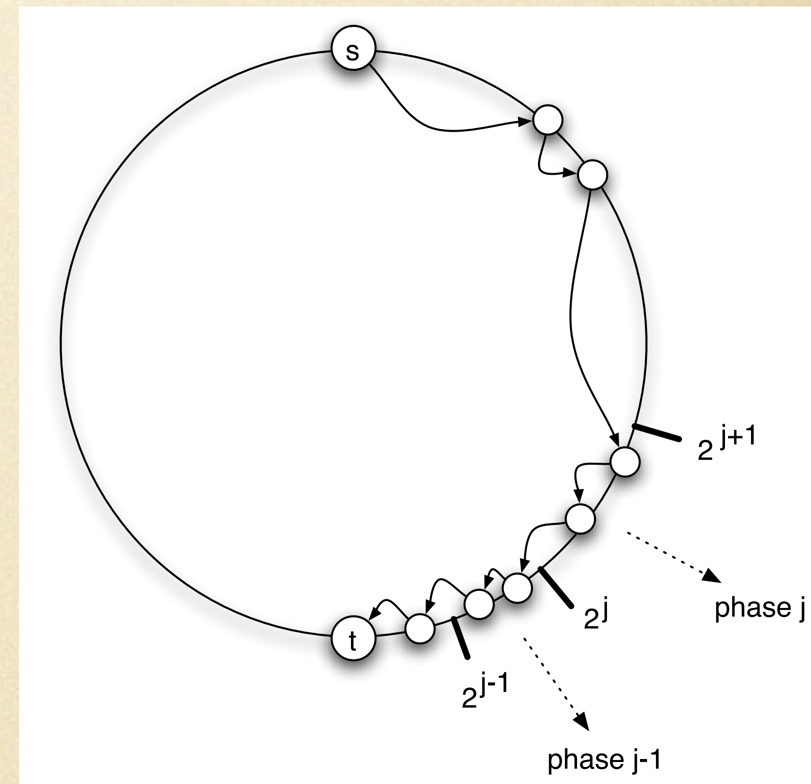
We'll say we're in **phase j** of the algorithm when distance from target is between 2^j and 2^{j-1}

Number of phases is log n



Analysis

We'll say we're in **phase j** of the algorithm when distance from target is between 2^j and 2^{j-1}

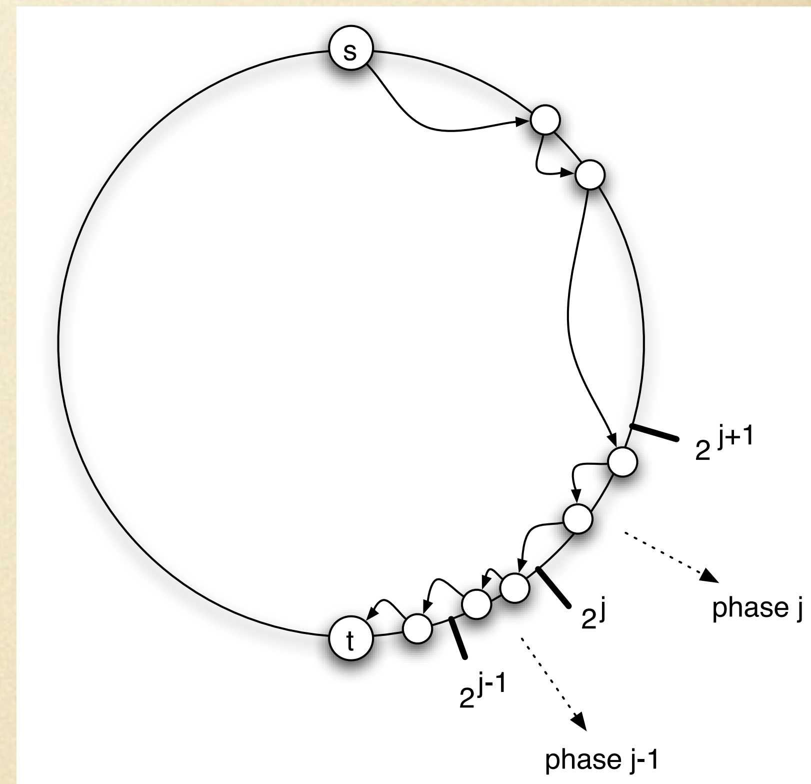


Number of phases is log n

Let X = # hops total; X_i= # hops in phase i

Analysis

We'll say we're in **phase j** of the algorithm when distance from target is between 2^j and 2^{j-1}



Number of phases is log n

Let X = # hops total; X_i= # hops in phase i

Then $X = X_1 + X_2 + ... + X_{\log n}$

Analysis

Then $X = X_1 + X_2 + \ldots + X_{\log n}$

Analysis

Then $X = X_1 + X_2 + \ldots + X_{\log n}$

$E(X) = E(X_1 + X_2 + \ldots + X_{\log n})$

Analysis

Then $X = X_1 + X_2 + \ldots + X_{\log n}$

$E(X) = E(X_1 + X_2 + \ldots + X_{\log n})$

$E(X) = E(X_1) + E(X_2) + \ldots + E(X_{\log n})$

By Linearity of Expectation!

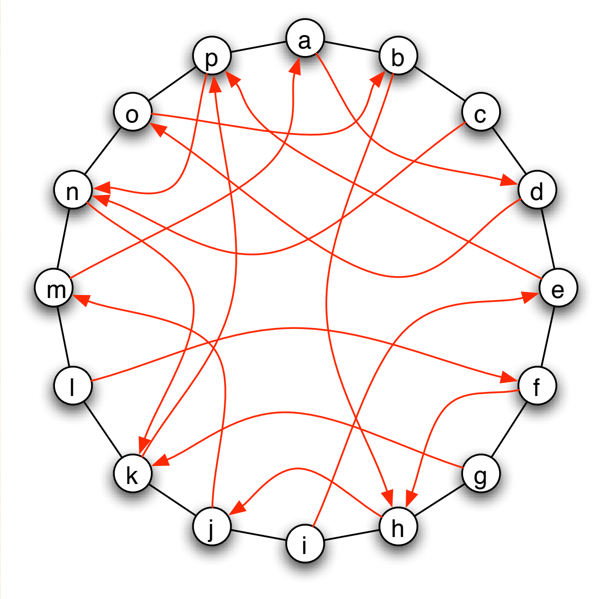
Analysis

$$E(X) = E(X_1) + E(X_2) + \ldots + E(X_{\log n})$$

Now we "just" need to calculate $E(X_i)$, the expected number of hops in phase i

To do this, we calculate the probability that a single random link allows us to end phase i

Probabilities

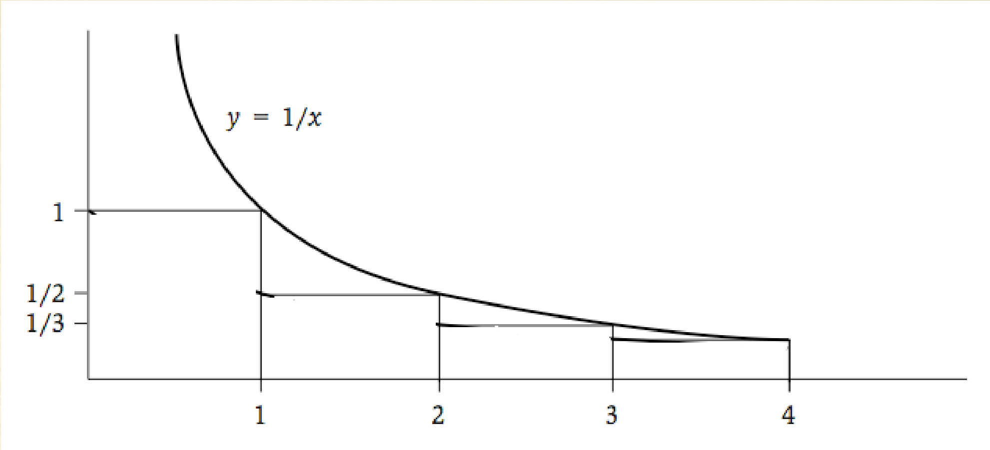


Recall: Random link to from u to v occurs with probability ∝1/(distance from u to v)

Normalizing constant Z is the sum over all v of 1/(distance from u to v)

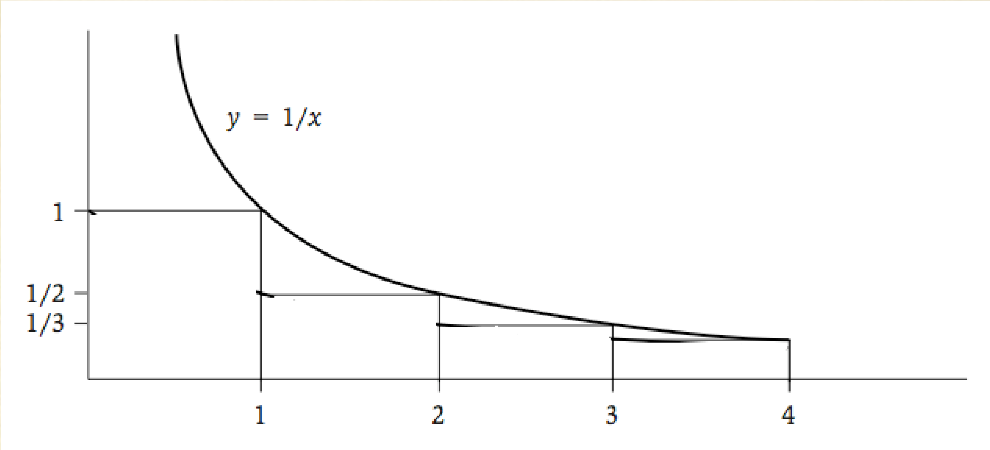
$$Z \leq 2(1 + 1/2 + 1/3 + 1/4 + \ldots 1/(n/2))$$

Z



$$Z \leq 2(1 + 1/2 + 1/3 + 1/4 + \ldots 1/(n/2))$$

Z

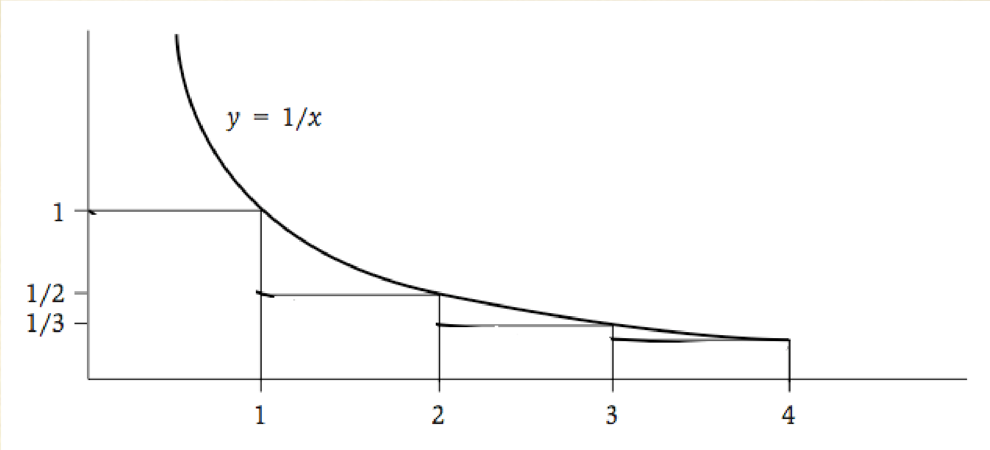


$$Z \leq 2(1 + 1/2 + 1/3 + 1/4 + \ldots 1/(n/2))$$

But:

$$(1 + 1/2 + 1/3 + 1/4 + \ldots 1/k) \leq 1 + \int_1^k \frac{1}{x} dx$$

Z



$$Z \leq 2(1 + 1/2 + 1/3 + 1/4 + \ldots 1/(n/2))$$

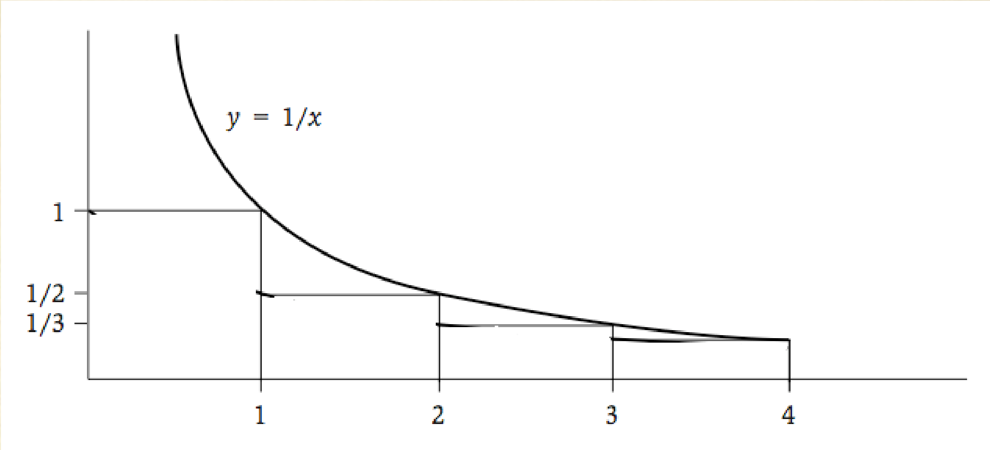
But:

$$(1 + 1/2 + 1/3 + 1/4 + \ldots 1/k) \leq 1 + \int_1^k \frac{1}{x} dx$$

And:

$$1 + \int_1^k \frac{1}{x} dx = 1 + \ln k$$

Z



$$Z \leq 2(1 + 1/2 + 1/3 + 1/4 + \ldots 1/(n/2))$$

But:

$$(1 + 1/2 + 1/3 + 1/4 + \ldots 1/k) \leq 1 + \int_1^k \frac{1}{x} dx$$

And:

$$1 + \int_1^k \frac{1}{x} dx = 1 + \ln k$$

So:

$$Z \leq 2(1 + \ln(n/2) \leq 2 \log_2 n$$

Probabilities

So:

$$Z \leq 2(1 + \ln(n/2) \leq 2\log_2 n$$

Probabilities

So:

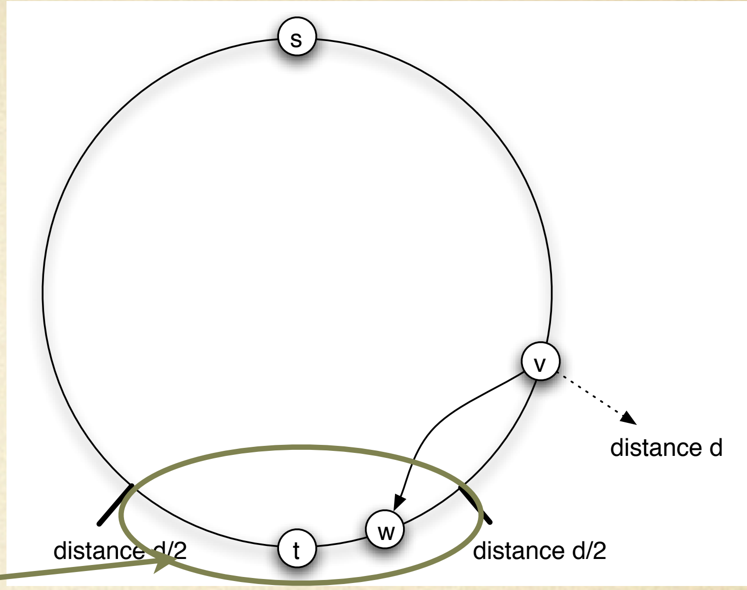
$$Z \leq 2(1 + \ln(n/2) \leq 2 \log_2 n$$

Let d(u,v) = distance from u to v. Then prob. u links to v is

$$\frac{1}{Z} d(u,v)^{-1} \geq \frac{1}{2 \log n} d(u,v)^{-1}$$

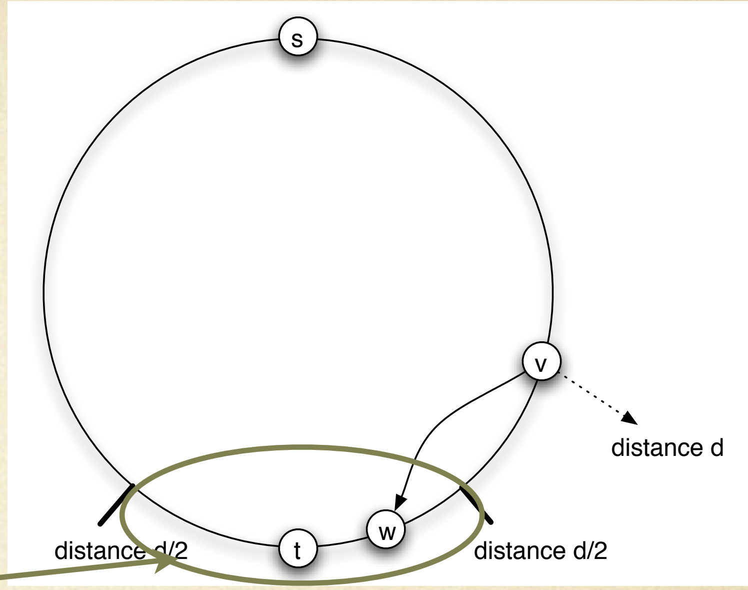
Only remaining task is to add up these probabilities over all vertices v that will let us exit the current phase

Probabilities



d+1 nodes within distance d/2 of t

Probabilities

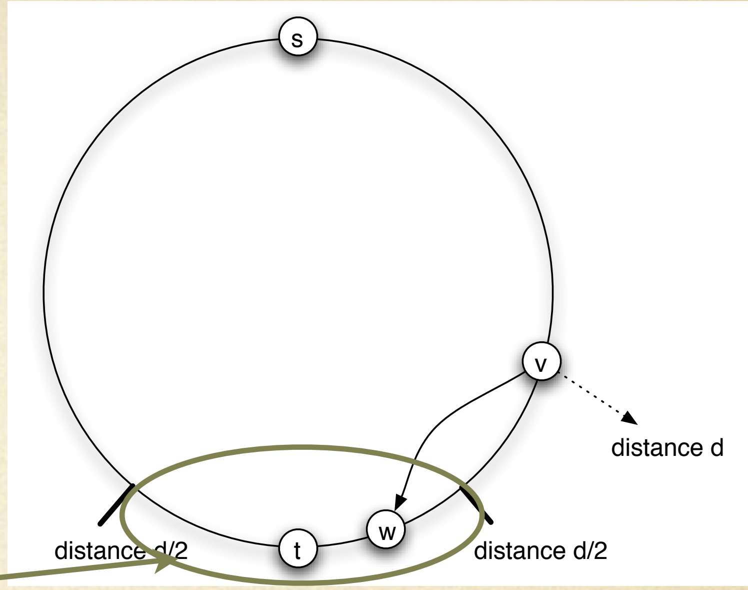


d+1 nodes within distance d/2 of t

Prob. of hitting particular node v in there at least:

$$\frac{1}{2\log n} d(u,v)^{-1} \geq \frac{1}{2\log n}\frac{1}{3d/2} = \frac{1}{3d\log n}$$

Probabilities



d+1 nodes within distance d/2 of t

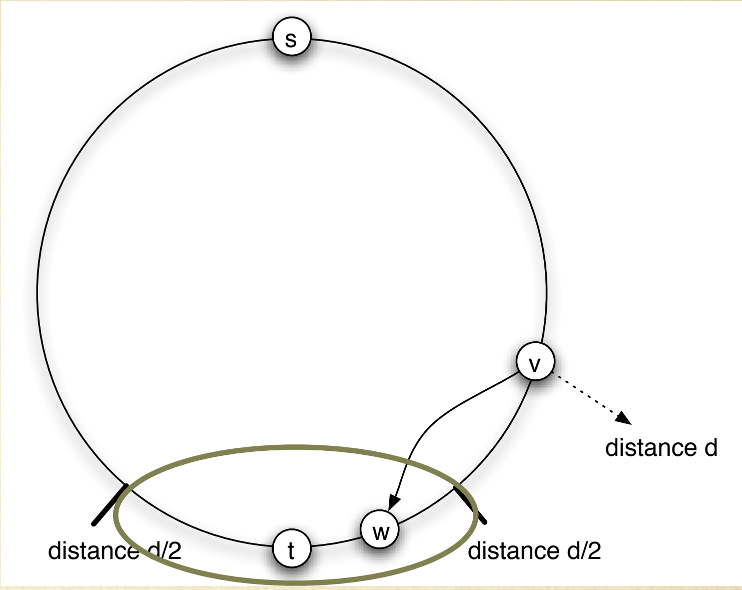
Prob. of hitting particular node v in there at least:

$$\frac{1}{2\log n}d(u,v)^{-1} \geq \frac{1}{2\log n}\frac{1}{3d/2} = \frac{1}{3d\log n}$$

d+1 total nodes; prob. of hitting one is at least:

$$(d+1)\frac{1}{3d\log n} = \frac{1}{3\log n}$$

Phases

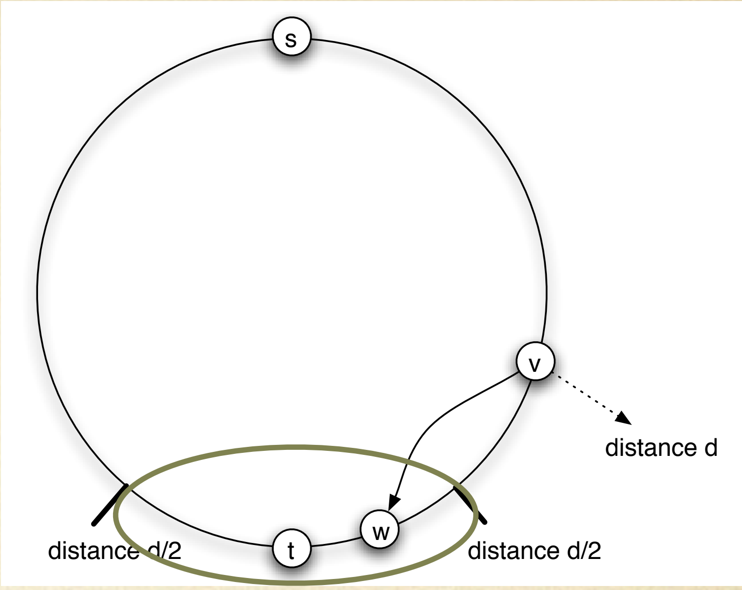


So we're walking around in phase j

Every time we see a random edge, it has prob. at least **1/(3 log n)** of taking us to next phase

Q: How long do we expect to walk before finding one of these special edges?

Phases

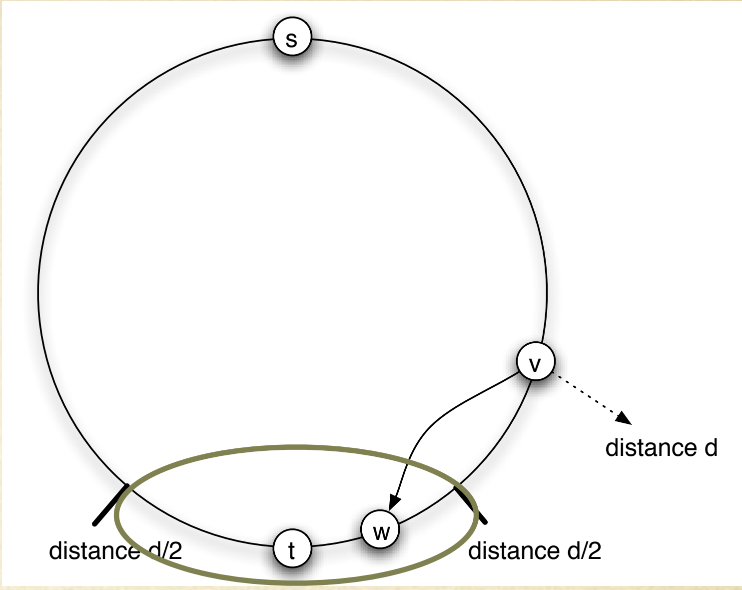


Q: How long do we expect to walk before finding one of these special edges?

Q: If a coin has probability **p** of coming up heads, how many times do you expect to flip it before you get heads?

A: 1/p

Phases



Q: How long do we expect to walk before finding one of these special edges?

Q: If a coin has probability **p** of coming up heads, how many times do you expect to flip it before you get heads?

A: 1/p

$$E(X) = p^*1 + (1-p)(1 + E(X))$$

Wrapup

Recall: $E(X) = E(X_1) + E(X_2) + \ldots + E(X_{\log n})$

Thus: $E(X) \leq 3 \log n + 3 \log n + \ldots + 3\log n$

$\leq 3 \log^2 n$

Wrapup

Recall: $E(X) = E(X_1) + E(X_2) + \ldots + E(X_{\log n})$

Thus: $E(X) \leq 3 \log n + 3 \log n + \ldots + 3 \log n$

$\leq 3 \log^2 n$

The End!

Wrapup

Recall: $E(X) = E(X_1) + E(X_2) + \dots + E(X_{\log n})$

Thus: $E(X) \leq 3 \log n + 3 \log n + \dots + 3 \log n$

$\leq 3 \log^2 n$

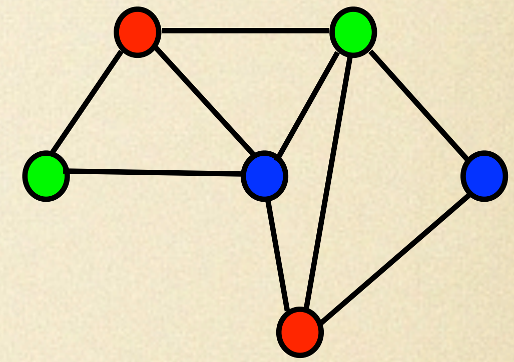
The End!

Or is It???

Open Questions

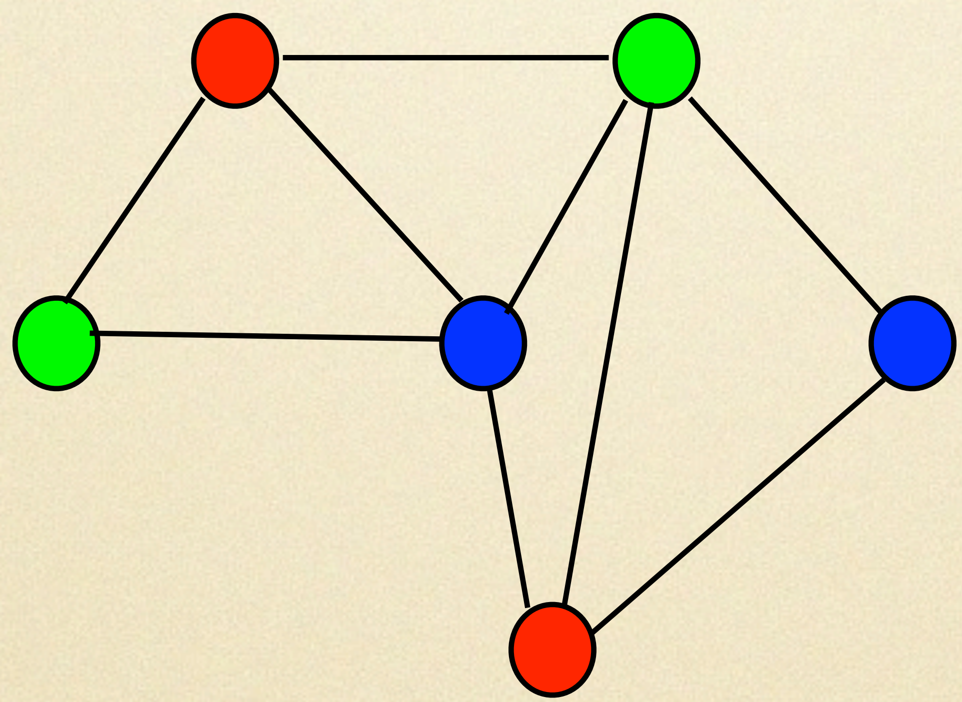
- Why do friendship links have the Kleinberg exponent?
- Why should routing speed determine the way in which we make friends?
- Why do we have friends?

Graph Coloring



- Must color each node in a graph (network)
- A coloring is **valid** if any pair of nodes that are linked have different colors
- Goal: Find a valid coloring using the smallest number of colors

Graph Coloring



Example graph and valid 3 coloring

Graph Coloring

- Unlike shortest paths, coloring is computational hard even when centralized
- Sudoku is a graph coloring problem (with some colors already fixed)
- How?

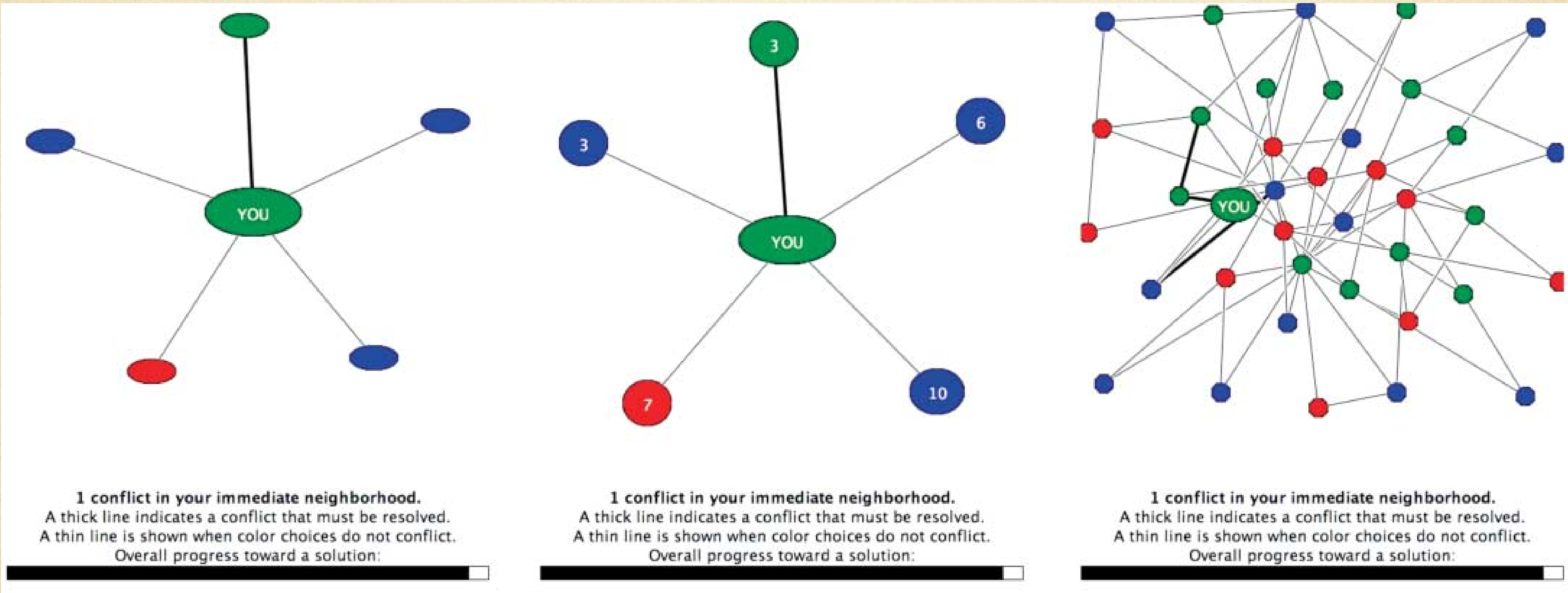
Distributed Coloring

- Division of resources in social networks
- Nodes are people, links represent friendships
- Colors are resources
- Goal: Assign resources to people so that friends don't fight over the same resource
- **Distributed**: Each node knows only local neighborhood

Example Resources

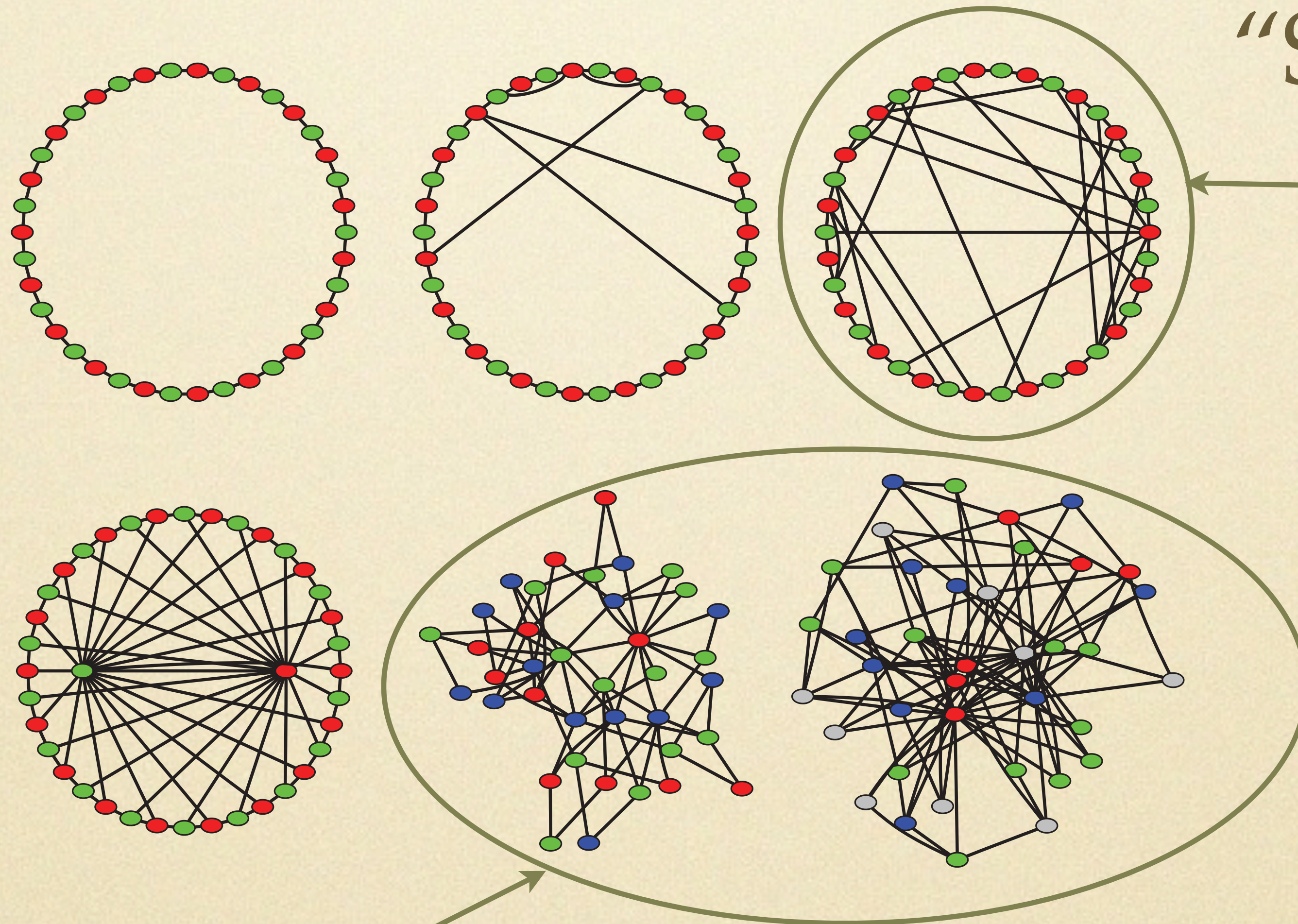
- **Time**: scheduling talks in conference rooms
- **Economic**: pursuing different expertise/markets by people/companies
- **Political**: pursuing different political offices
- **Technological**: selecting a channel unused by close parties in a wireless network

An Experiment



Kearns et al. '06 ran a distributed coloring experiment on people

Graphs Used



"Small World" (Watts-Strogatz)

Kleinberg?!?

Preferential attachment

Empirical Results

	Graph statistics								
	Colors required (No.)	Min. links (No.)	Max. links (No.)	Avg. links (No.)	SD	Avg. distance (No. of links)	Avg. experiment duration (s) and fraction solved		Distributed heuristic (No. of color changes)
Simple cycle	2	2	2	2	0	9.76	144.17	5/6	378
5-chord cycle	2	2	4	2.26	0.60	5.63	121.14	7/7	687
20-chord cycle	2	2	7	3.05	1.01	3.34	65.67	6/6	8265
Leader cycle	2	3	19	3.84	3.62	2.31	40.86	7/7	8797
Pref. att., $\nu = 2$	3	2	13	3.84	2.44	2.63	219.67	2/6	1744
Pref. att., $\nu = 3$	4	3	22	5.68	4.22	2.08	154.83	4/6	4703

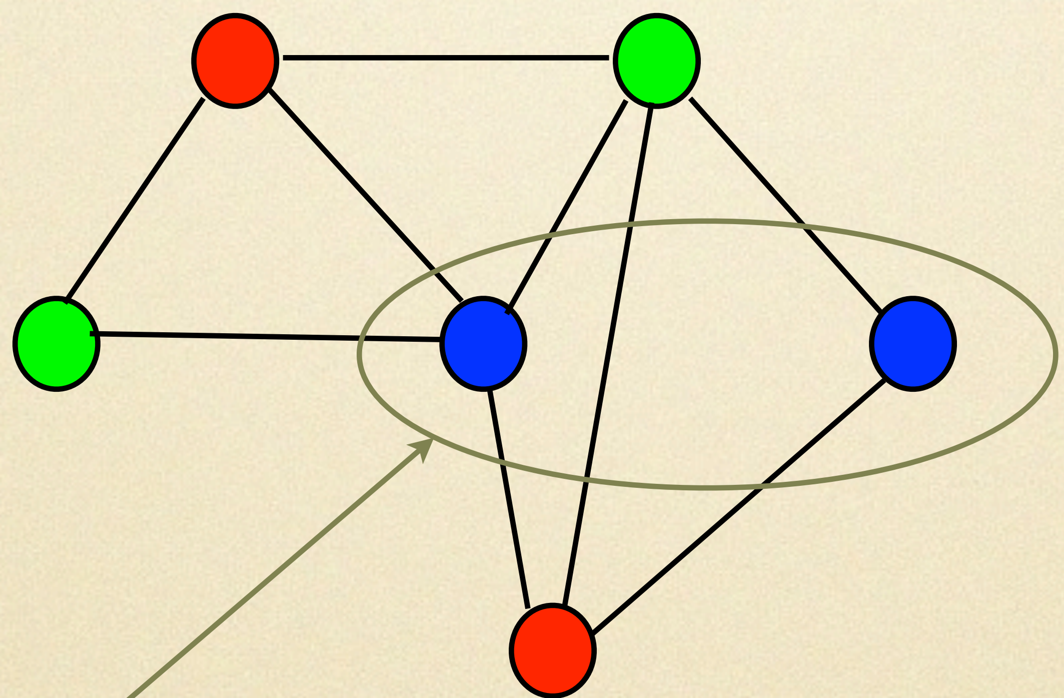
Small world easy

Preferential attachment hard

Maximal Independent Set

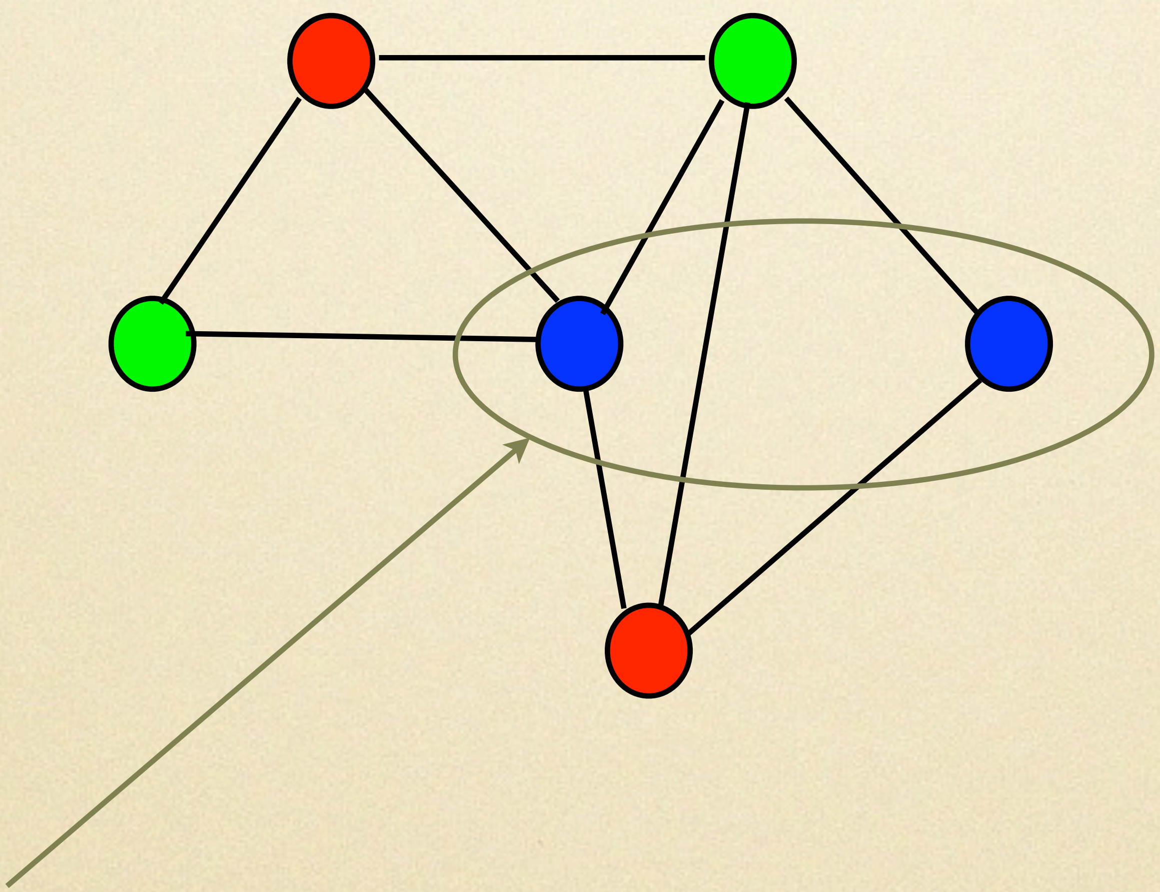
- To solve distributed graph coloring, we first address a simpler problem:
- **Independent Set:** A set of nodes in a network, such that there is no edge between any pair in the set
- An independent set is **maximal** if no nodes can be added

Maximal Independent Set



This is a maximal independent set

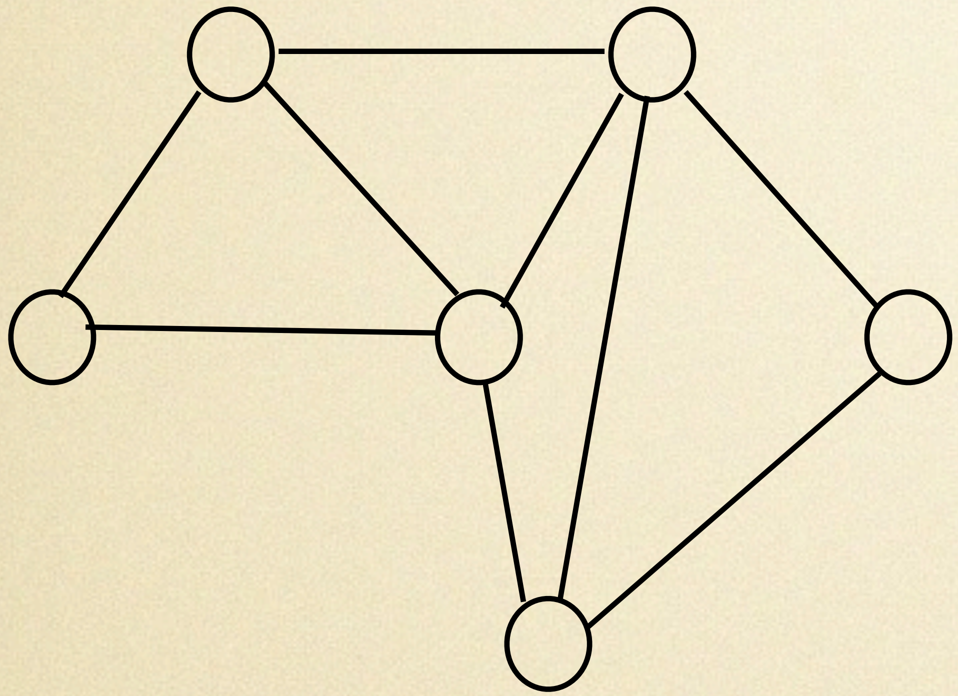
Maximal Independent Set



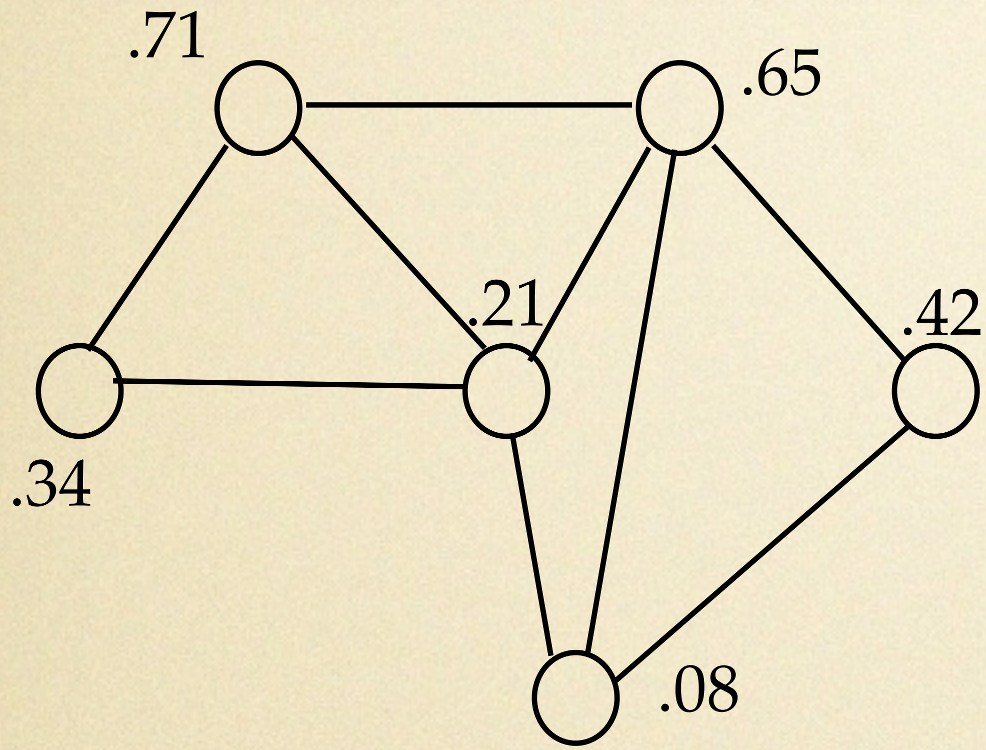
This is a maximal independent set

Note: all nodes in an independent set can be colored with the same color

An MIS Algorithm

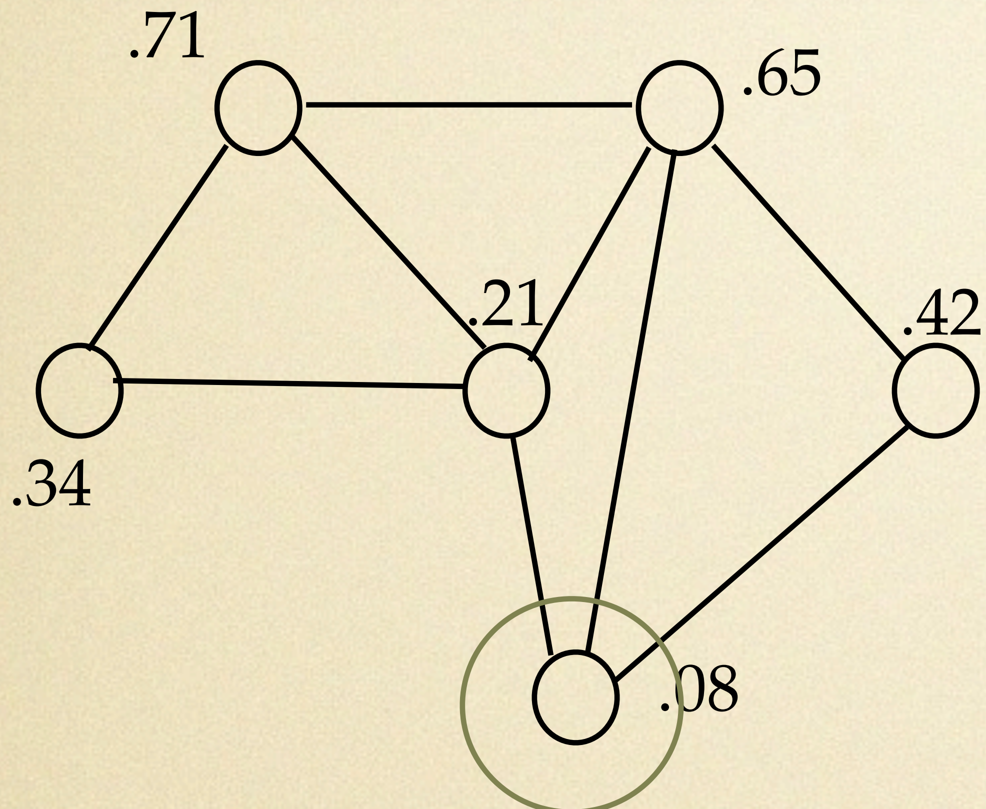


An MIS Algorithm



1) Each node v chooses a random value, r(v), in [0,1] and sends it to its neighbors

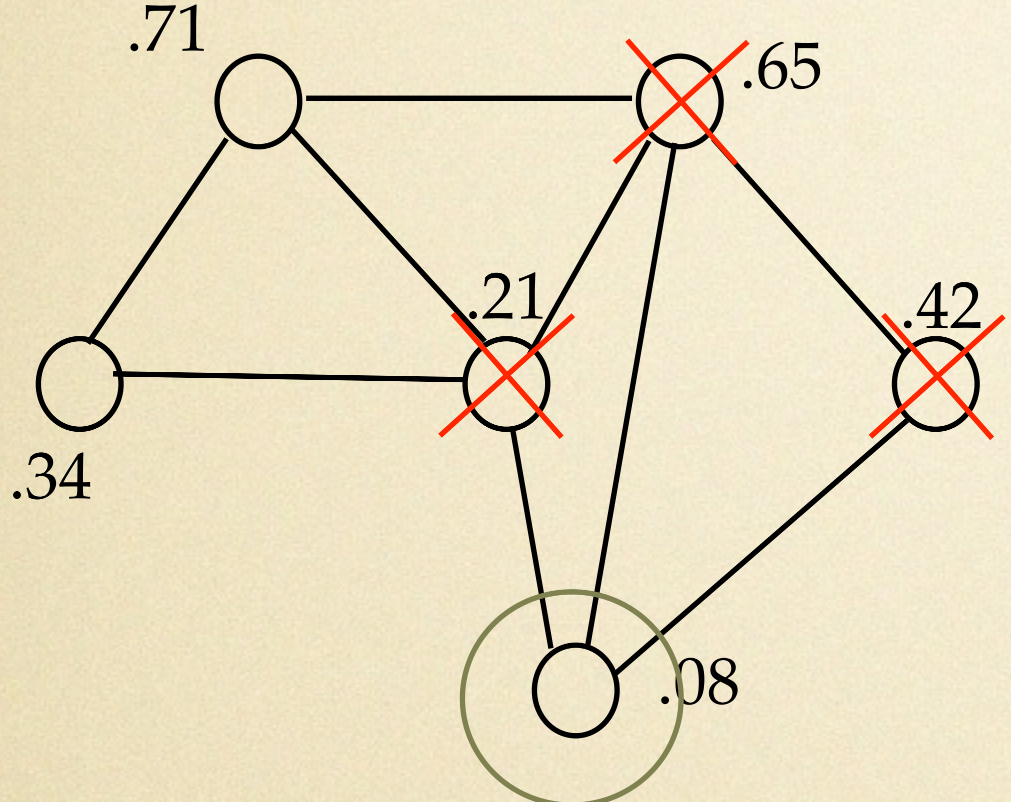
An MIS Algorithm



1) Each node v chooses a random value, r(v), in [0,1] and sends it to its neighbors

2) If r(v) < r(w) for all neighbors w of v, then v enters the MIS and informs its neighbors

An MIS Algorithm



1) Each node v chooses a random value, r(v), in [0,1] and sends it to its neighbors

2) If r(v) < r(w) for all neighbors w of v, then v enters the MIS and informs its neighbors

3) If v or a neighbor entered the MIS, it terminates (removing all edges); otherwise go back to step 1

Some Facts

- The algorithm always finds a MIS
- The algorithm terminates since in each loop, at least one node is added
- Q: How fast is the algorithm?

Analysis

- We'll show that, in expectation, half of the edges are removed in each loop of the algorithms
- This implies that number of loops is only log m where m is number of edges, n number of nodes
- Since $m \leq n^2$, we know that $\log m \leq 2 \log n$
- We'll let d(x) be the "degree of x" i.e. number of edges incident to x

A Clever Trick

Let $v \Rightarrow w$ be the event that $r(v) \leq r(w)$ **and** $r(v) \leq r(x)$ for all neighbors x of w

A Clever Trick

Let $v \Rightarrow w$ be the event that $r(v) \leq r(w)$ **and** $r(v) \leq r(x)$ for all neighbors x of w

Let $X_{v \Rightarrow w} = d(w)$ if event $v \Rightarrow w$ occurs and 0 otherwise

A Clever Trick

Let $v \Rightarrow w$ be the event that $r(v) \leq r(w)$ **and** $r(v) \leq r(x)$ for all neighbors x of w

Let $X_{v \Rightarrow w} = d(w)$ if event $v \Rightarrow w$ occurs and 0 otherwise

Let $X = \sum_{((v,w) \text{ in } E)} X_{v \Rightarrow w}$, where E is the set of edges

A Clever Trick

Let $v \Rightarrow w$ be the event that $r(v) \leq r(w)$ **and** $r(v) \leq r(x)$ for all neighbors x of w

Let $X_{v \Rightarrow w} = d(w)$ if event $v \Rightarrow w$ occurs and 0 otherwise

Let $X = \sum_{((v,w) \text{ in } E)} X_{v \Rightarrow w}$, where E is the set of edges

Note that $X \leq (1/2)$*total number of edges removed!

A Clever Trick

Let $v \Rightarrow w$ be the event that $r(v) \leq r(w)$ **and** $r(v) \leq r(x)$ for all neighbors x of w

Let $X_{v \Rightarrow w} = d(w)$ if event $v \Rightarrow w$ occurs and 0 otherwise

Let $X = \sum_{((v,w) \text{ in } E)} X_{v \Rightarrow w}$, where E is the set of edges

Note that $X \leq (1/2)$*total number of edges removed!

Since for any edge (s,t), at most one event $X_{* \Rightarrow s}$ and at most one event $X_{* \Rightarrow t}$ can happen.

A Clever Trick

Now all that remains is to compute E(X)

$$E(X) = E(\sum_{((v,w) \text{ in } E)} X_{v \Rightarrow w})$$

A Clever Trick

Now all that remains is to compute E(X)

$$E(X) = E(\textstyle\sum_{((v,w) \text{ in } E)} X_{v \Rightarrow w})$$

$$E(X) = \textstyle\sum_{((v,w) \text{ in } E)} E(X_{v \Rightarrow w}) + E(X_{w \Rightarrow v})$$

A Clever Trick

Now all that remains is to compute E(X)

$$E(X) = E(\sum_{((v,w) \text{ in } E)} X_{v \Rightarrow w})$$

$$E(X) = \sum_{((v,w) \text{ in } E)} E(X_{v \Rightarrow w}) + E(X_{w \Rightarrow v})$$

$$= \sum_{((v,w) \text{ in } E)} \Pr(\text{event } v \Rightarrow w)\, d(w) + \Pr(\text{event } w \Rightarrow v) d(v)$$

A Clever Trick

Now all that remains is to compute E(X)

$$E(X) = E(\sum_{((v,w) \text{ in } E)} X_{v\Rightarrow w})$$

$$E(X) = \sum_{((v,w) \text{ in } E)} E(X_{v\Rightarrow w}) + E(X_{w\Rightarrow v})$$

$$= \sum_{((v,w) \text{ in } E)} \Pr(\text{event } v\Rightarrow w)\, d(w) + \Pr(\text{event } w\Rightarrow v) d(v)$$

$$\geq \sum_{((v,w) \text{ in } E)} d(w)/(d(v)+d(w)) + d(v)/(d(w) + d(v))$$

A Clever Trick

Now all that remains is to compute E(X)

$$E(X) = E(\textstyle\sum_{((v,w) \text{ in } E)} X_{v\Rightarrow w})$$

$$E(X) = \textstyle\sum_{((v,w) \text{ in } E)} E(X_{v\Rightarrow w}) + E(X_{w\Rightarrow v})$$

$$= \textstyle\sum_{((v,w) \text{ in } E)} \Pr(\text{event } v\Rightarrow w)\, d(w) + \Pr(\text{event } w\Rightarrow v) d(v)$$

$$\geq \textstyle\sum_{((v,w) \text{ in } E)} d(w)/(d(v)+d(w)) + d(v)/(d(w) + d(v))$$

$$= \textstyle\sum_{((v,w) \text{ in } E)} 1$$

A Clever Trick

Now all that remains is to compute E(X)

$$E(X) = E(\sum_{((v,w) \text{ in } E)} X_{v \Rightarrow w})$$

$$E(X) = \sum_{((v,w) \text{ in } E)} E(X_{v \Rightarrow w}) + E(X_{w \Rightarrow v})$$

$$= \sum_{((v,w) \text{ in } E)} \Pr(\text{event } v \Rightarrow w)\, d(w) + \Pr(\text{event } w \Rightarrow v) d(v)$$

$$\geq \sum_{((v,w) \text{ in } E)} d(w) / (d(v)+d(w)) + d(v) / (d(w) + d(v))$$

$$= \sum_{((v,w) \text{ in } E)} 1$$

$$= m$$

Recap

- We've shown that E(X) = m
- We also shown that the number of edges removed in each loop is at least X/2
- Implies that we expect **half** the edges to be removed in each loop
- Thus, we expect only log m iterations of the loop!

Recap

- We've shown that E(X) = m
- We also shown that the number of edges removed in each loop is at least X/2
- Implies that we expect **half** the edges to be removed in each loop
- Thus, we expect only log m iterations of the loop!

The End!

Recap

- We've shown that E(X) = m
- We also shown that the number of edges removed in each loop is at least X/2
- Implies that we expect **half** the edges to be removed in each loop
- Thus, we expect only log m iterations of the loop!

The End!

Or is It???

Open Problems

- A major open problem in distributed computing is whether or not we can do better than logarithmic time for MIS

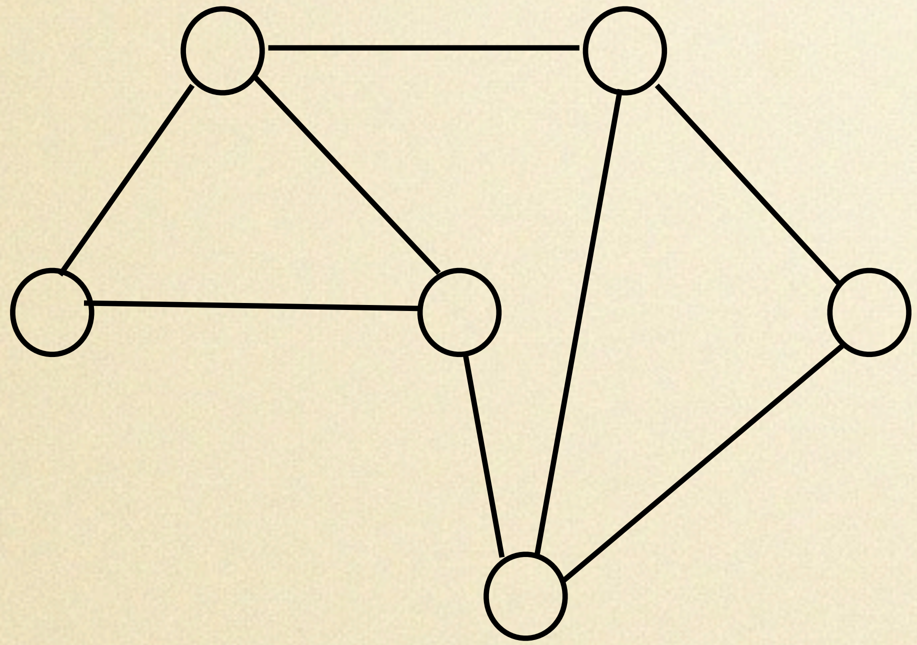
- Or at least come up with a deterministic algorithm that takes logarithmic time.

Open Problems

- A major open problem in distributed computing is whether or not we can do better than logarithmic time for MIS
- Or at least come up with a deterministic algorithm that takes logarithmic time.

Also: hey, what about graph coloring?

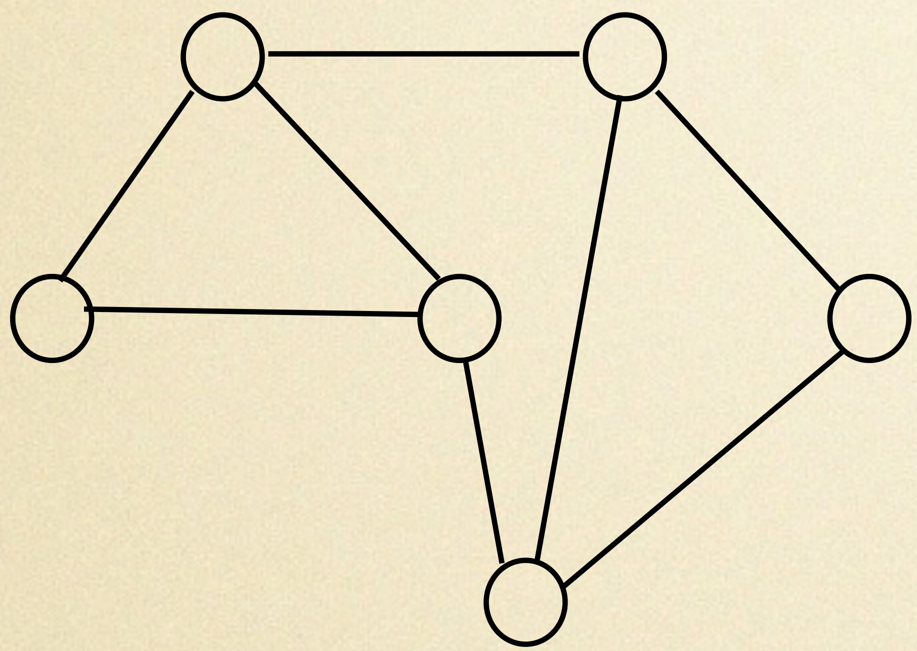
Create New Graph

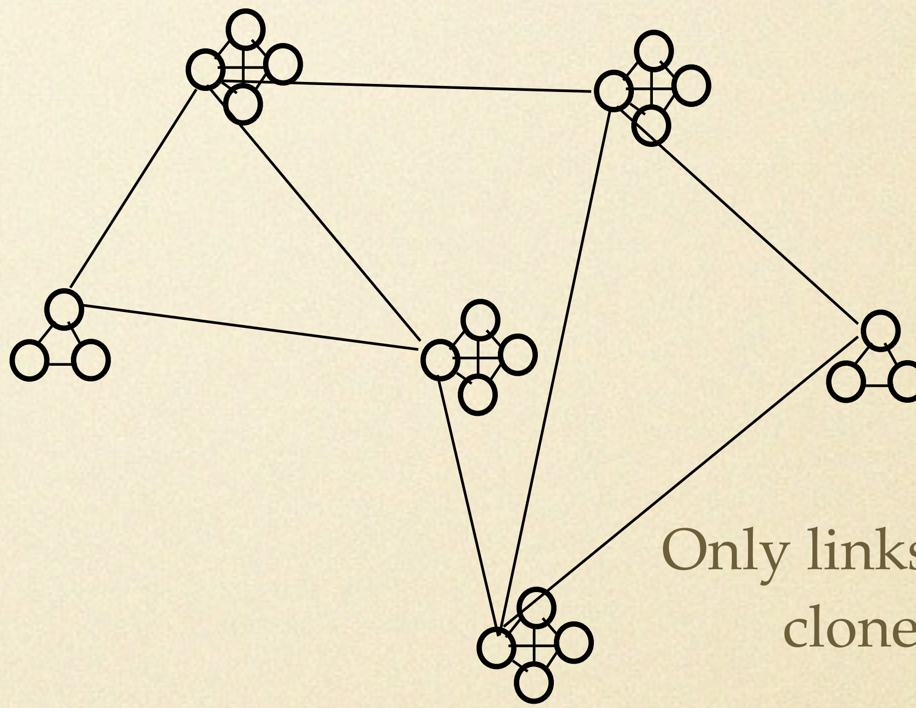


1) Each node v makes d(v)+1 clones. All clones of v are linked together

2) If u and v neighbors, then for all i, the i-th clone of u is linked to the i-th clone of v

Create New Graph



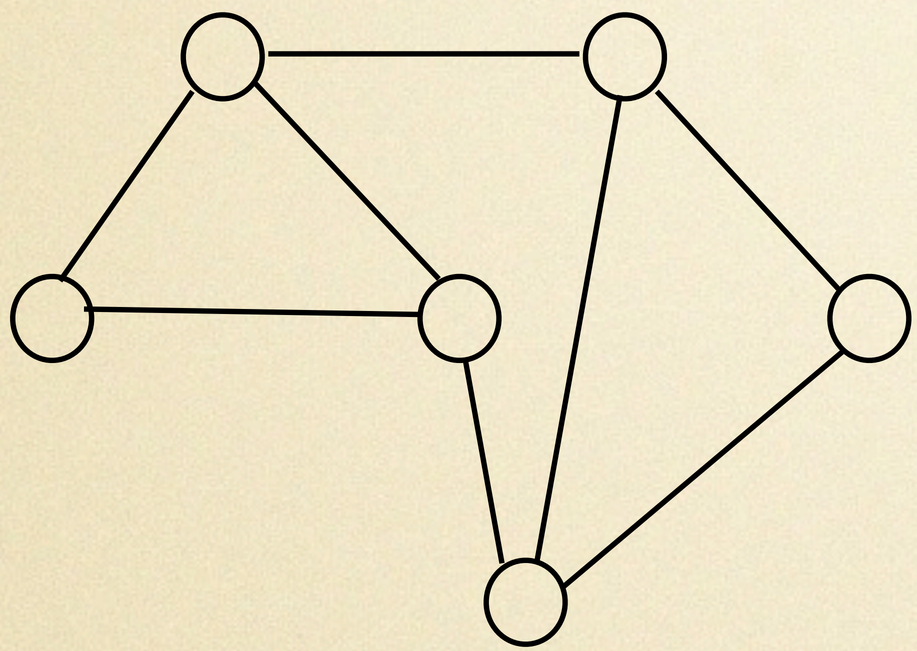


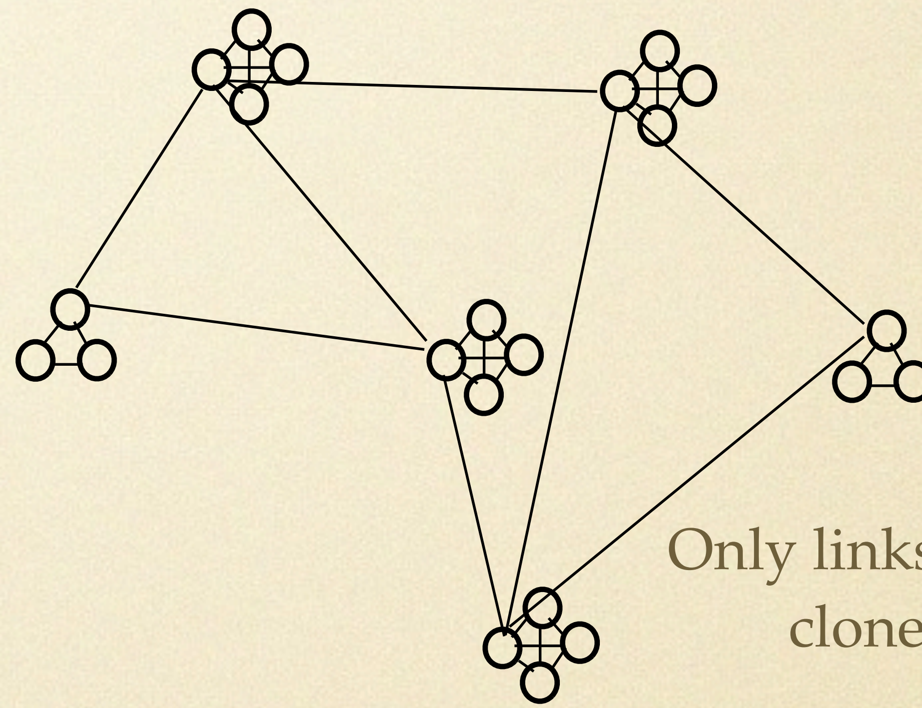
Only links between 1st clones shown

1) Each node v makes d(v)+1 clones. All clones of v are linked together

2) If u and v neighbors, then for all i, the i-th clone of u is linked to the i-th clone of v

Create New Graph





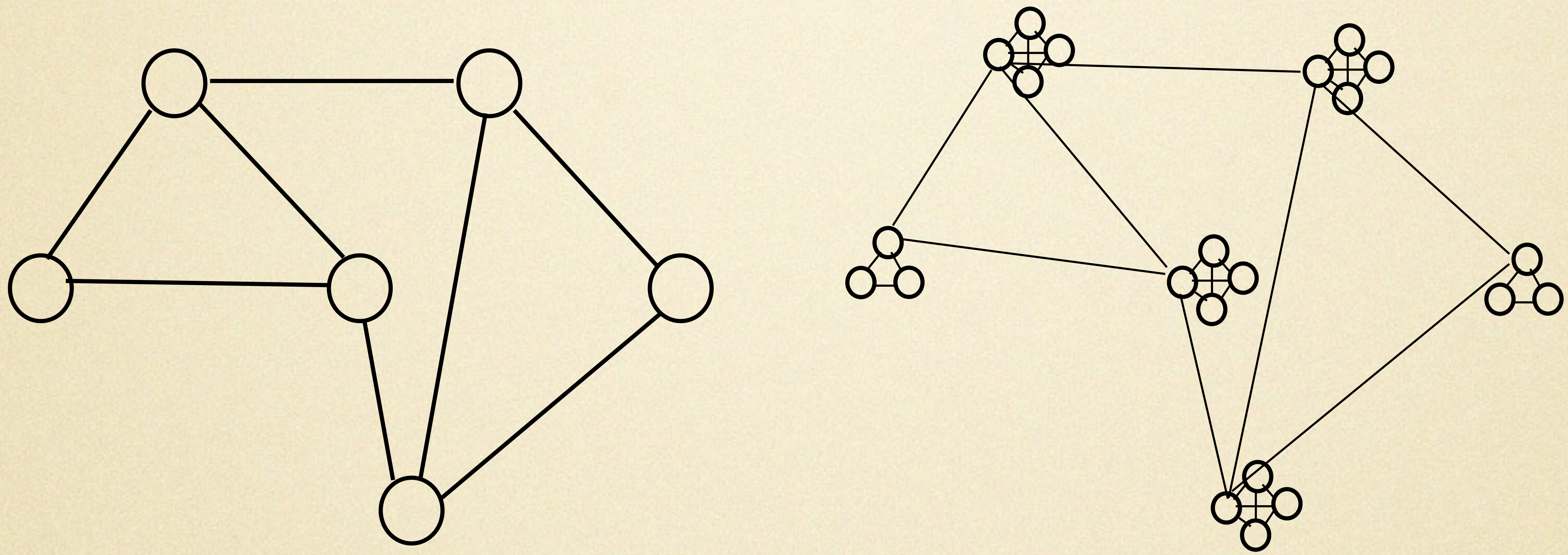
Only links between 1st clones shown

1) Each node v makes d(v)+1 clones. All clones of v are linked together

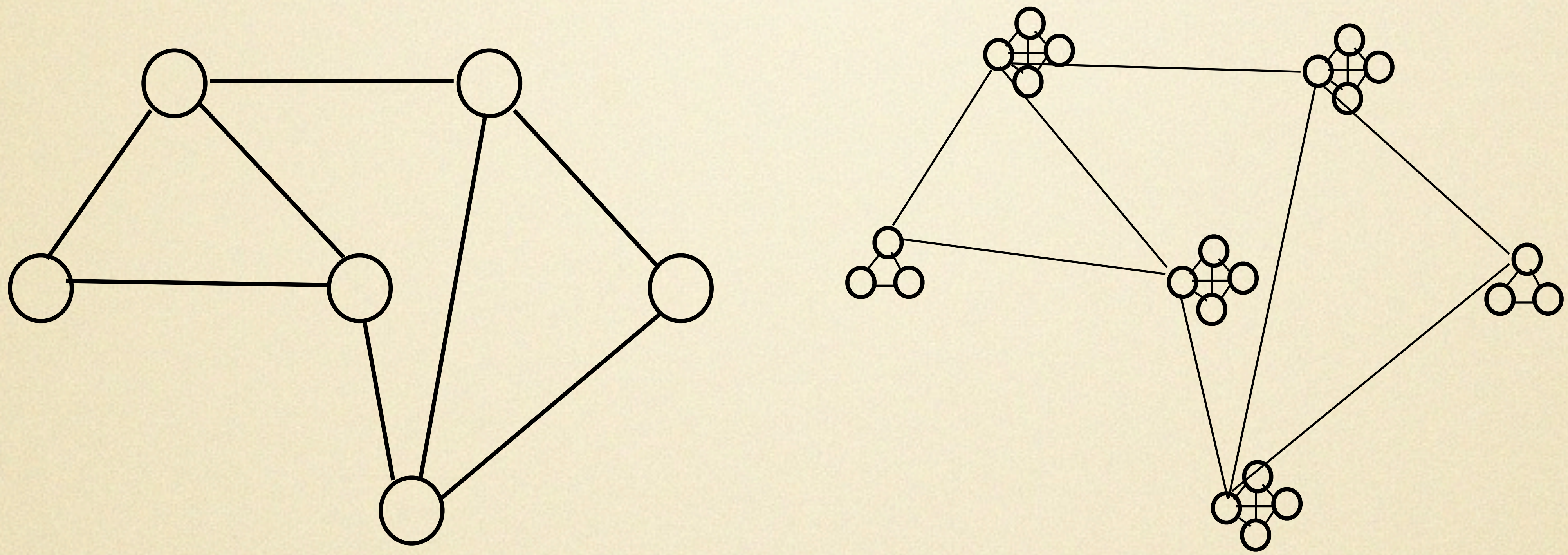
2) If u and v neighbors, then for all i, the i-th clone of u is linked to the i-th clone of v

3) We now run the MIS algorithm on the new graph. If the i-th clone of v is in the MIS, v is colored i!

A Coloring Algorithm

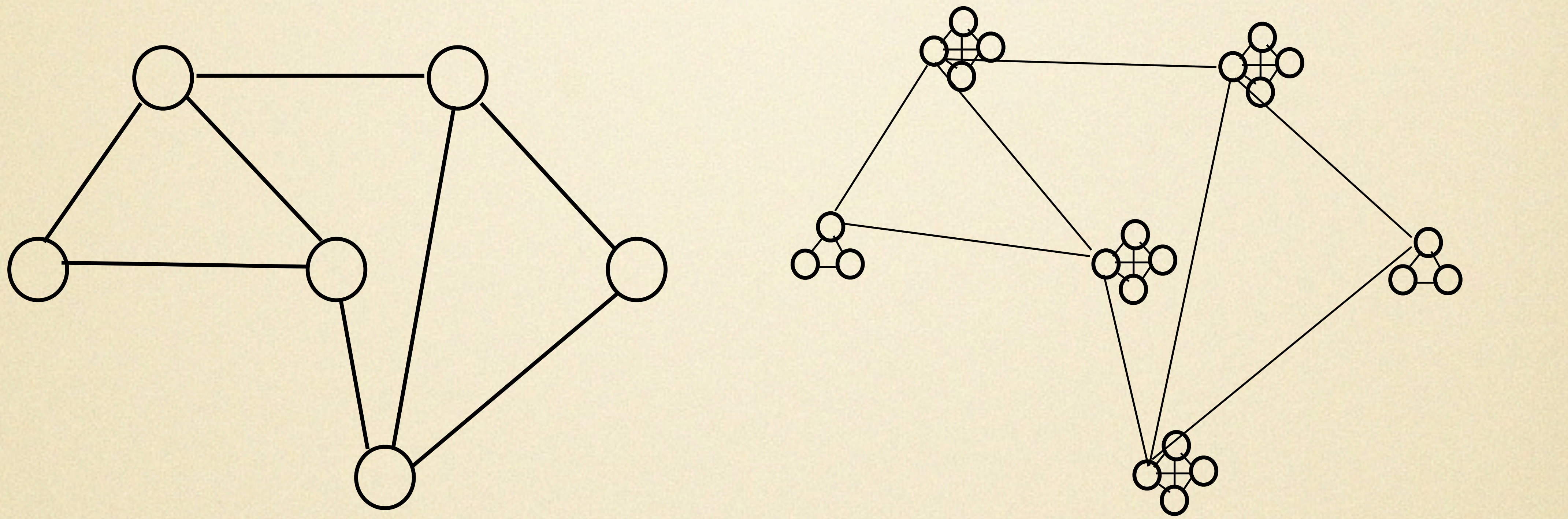


A Coloring Algorithm



Fact 1: For any node v, at most one clone is in the MIS

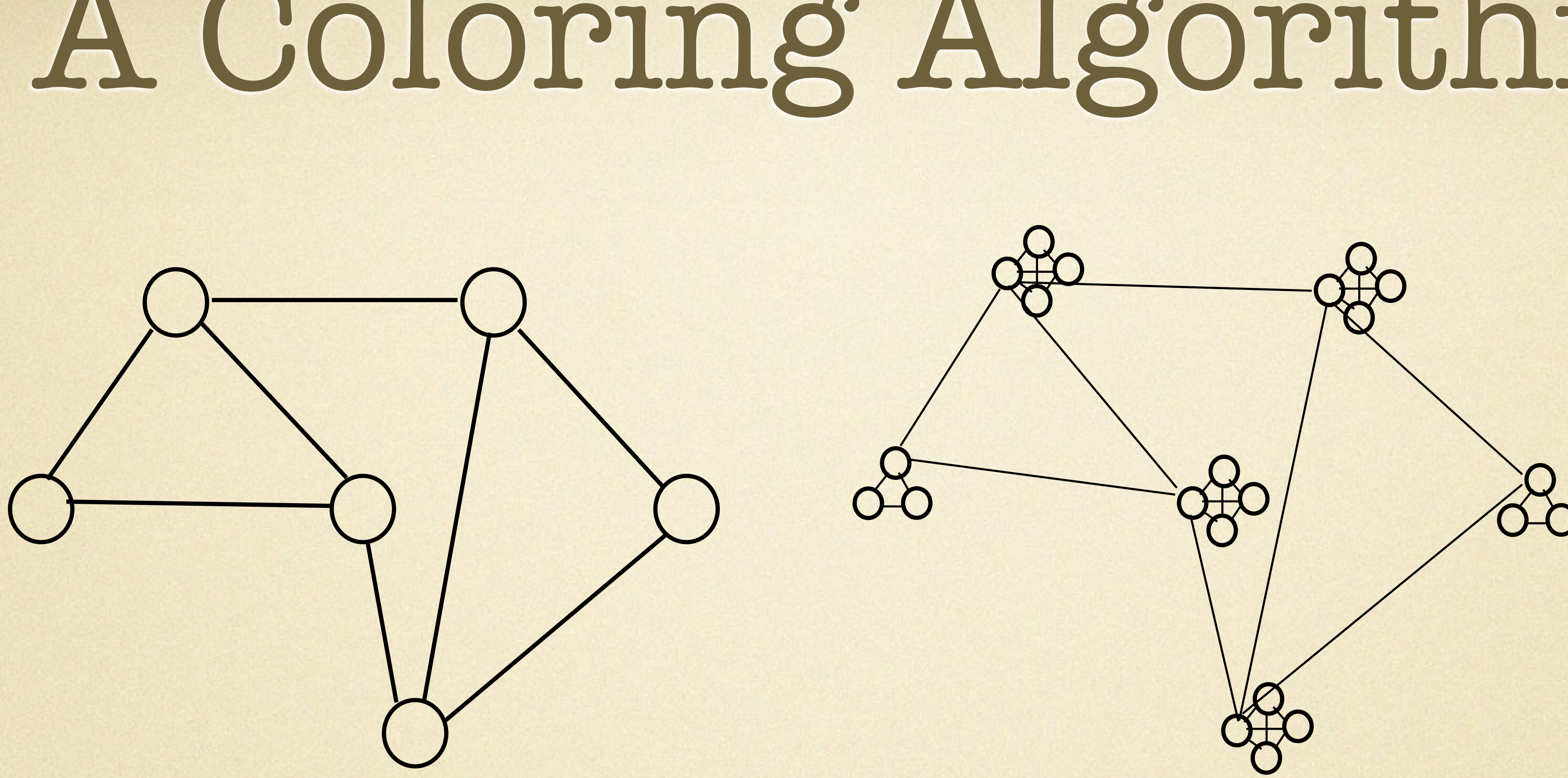
A Coloring Algorithm



Fact 1: For any node v, at most one clone is in the MIS

Fact 2: For any node v, at **least** one clone is in the MIS

A Coloring Algorithm

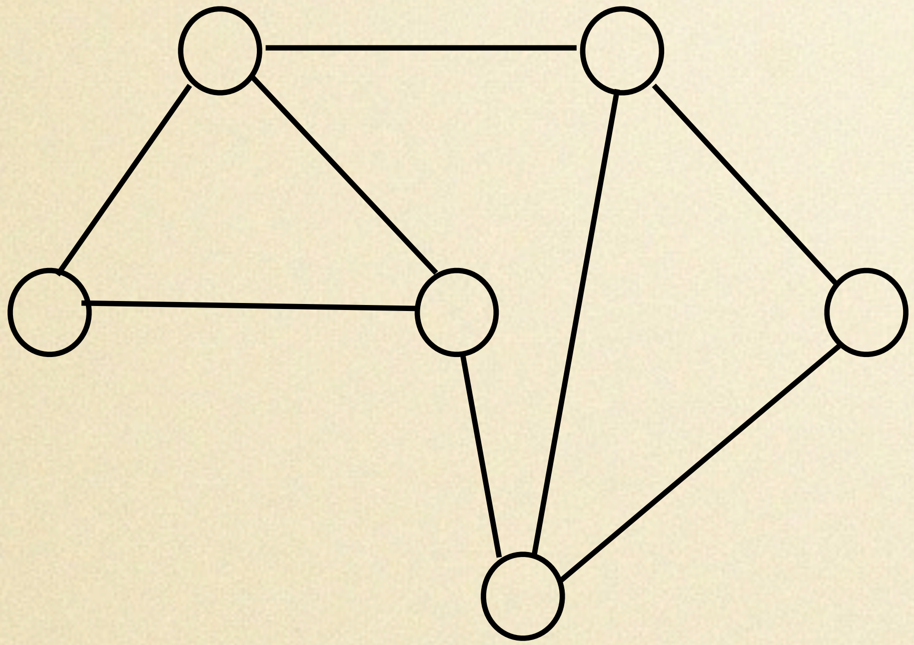


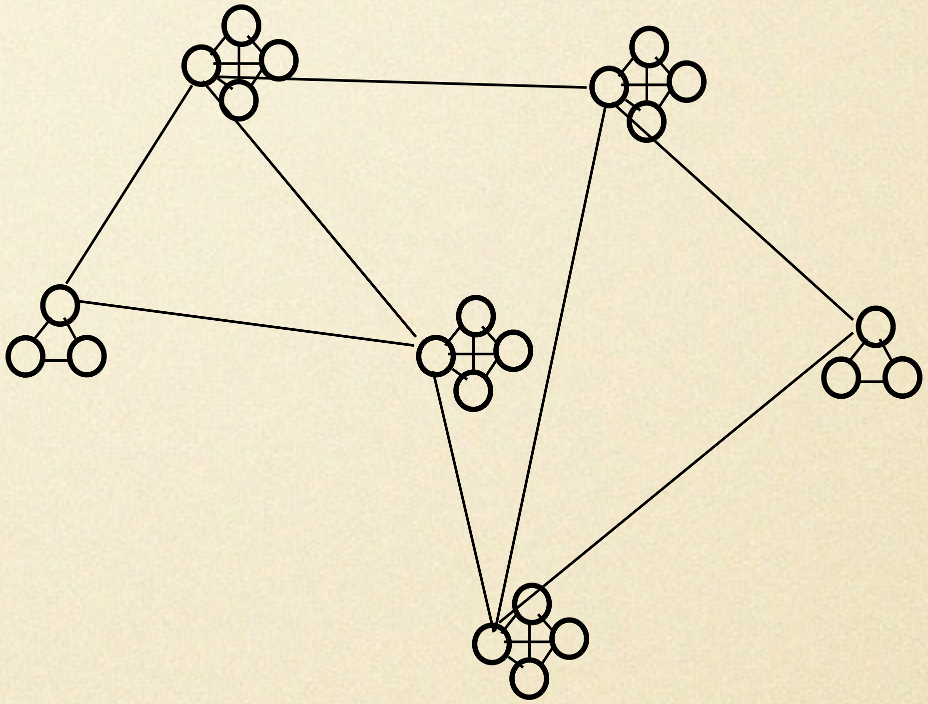
Fact 1: For any node v, at most one clone is in the MIS

Fact 2: For any node v, at **least** one clone is in the MIS

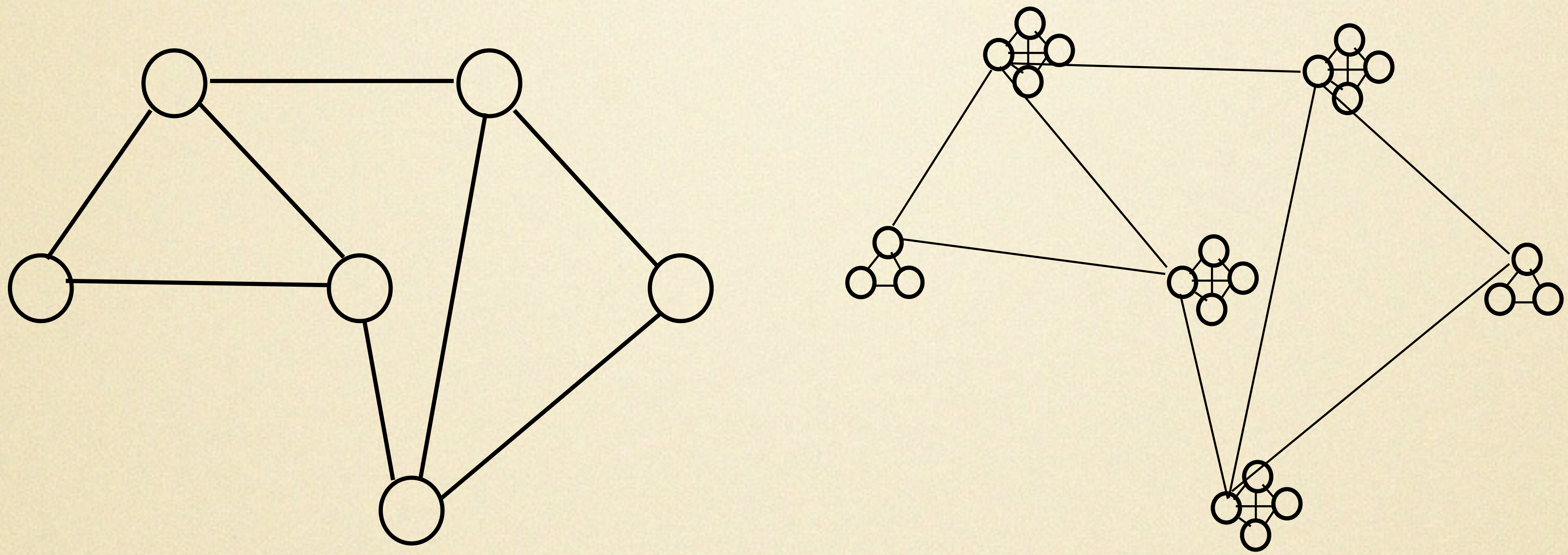
Fact 3: The running time is logarithmic since the new graph has at most m^2 edges

Wrapup



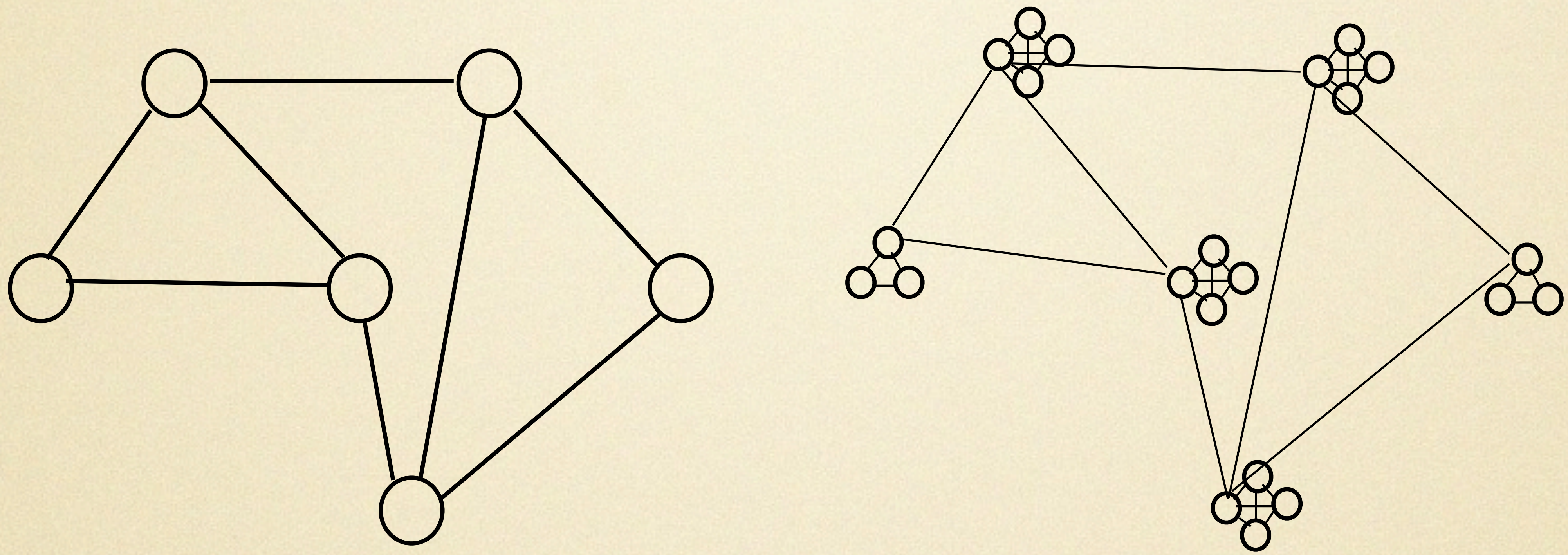


Wrapup



The algorithm colors any graph with Δ+1 colors, where Δ is the maximum degree of a node

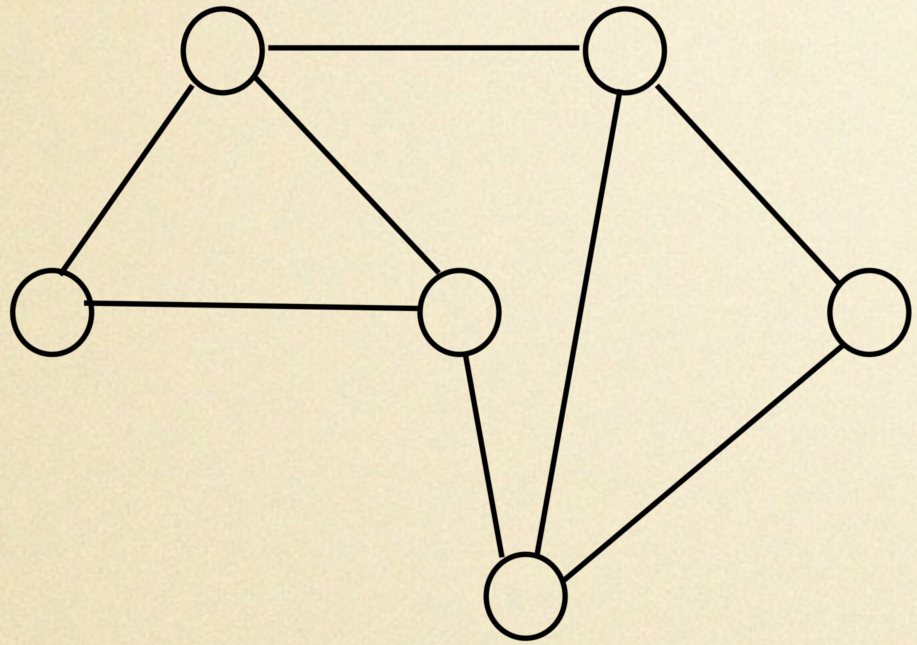
Wrapup

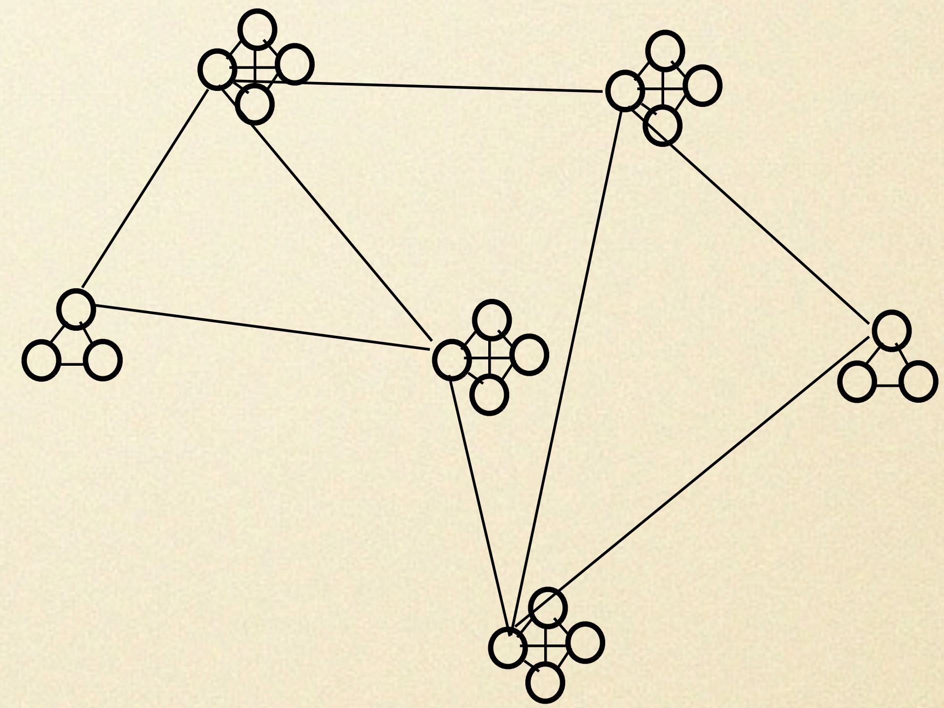


The algorithm colors any graph with Δ+1 colors, where Δ is the maximum degree of a node

The algorithm takes time logarithmic in n

Wrapup





The algorithm colors any graph with Δ+1 colors, where Δ is the maximum degree of a node

The algorithm takes time logarithmic in n

Note: Δ is not necessarily the minimum number of colors needed!

Questions

Questions

Note: There are faster coloring algorithms (log log n is even possible)

Questions

Note: There are faster coloring algorithms (log log n is even possible)

Question: How does the structure of the graph (small world, preferential attachment) effect the difficulty of graph coloring in practice?

Questions

Note: There are faster coloring algorithms (log log n is even possible)

Question: How does the structure of the graph (small world, preferential attachment) effect the difficulty of graph coloring in practice?

Answer: We don't know!

Conclusion

- Many problems can be solved efficiently over large networks
- Randomness is a powerful tool, but need to get the distributions right!
- Interaction between **Form** (topology) and **Function** (computation) is critical
- Still much work needed to understand this interaction