# Sinkless orientation *made simple*

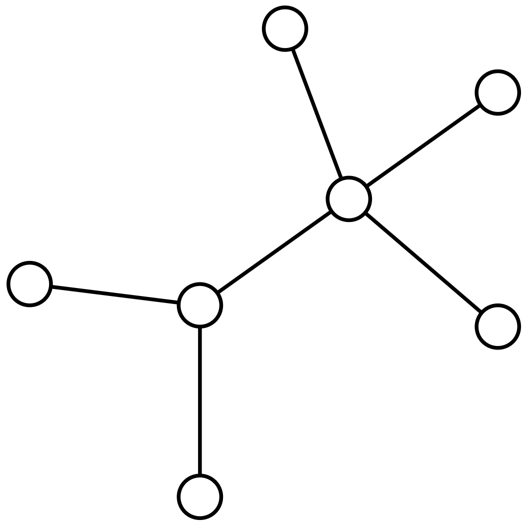
Aalto University · Finland
Gran Sasso Science Institute · Italy
IST Austria · Austria
LISN, CNRS · France
TU Berlin · Germany
University of Freiburg · Germany

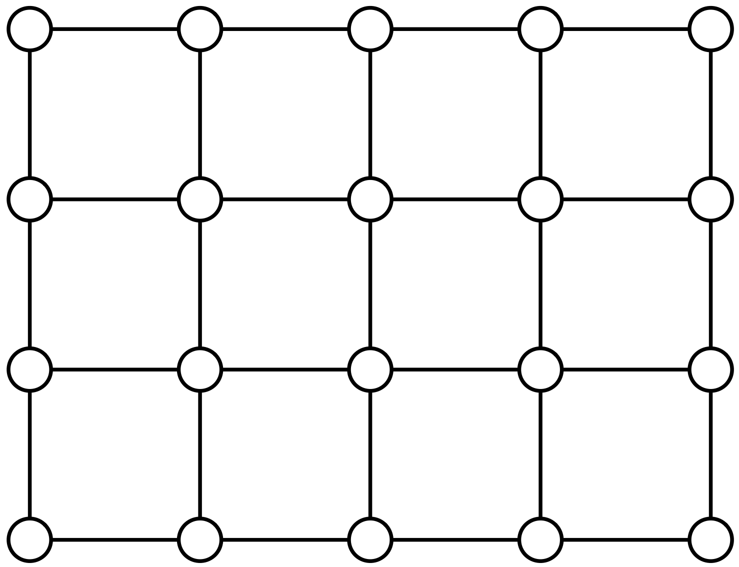
Alkida Balliu
Janne H. Korhonen
Fabian Kuhn
Henrik Lievonen
Dennis Olivetti
Shreyas Pai
Ami Paz
Joel Rybicki
Stefan Schmid
Jan Studený
Jukka Suomela
**Jara Uitto**

# Sinkless orientation

Given a graph

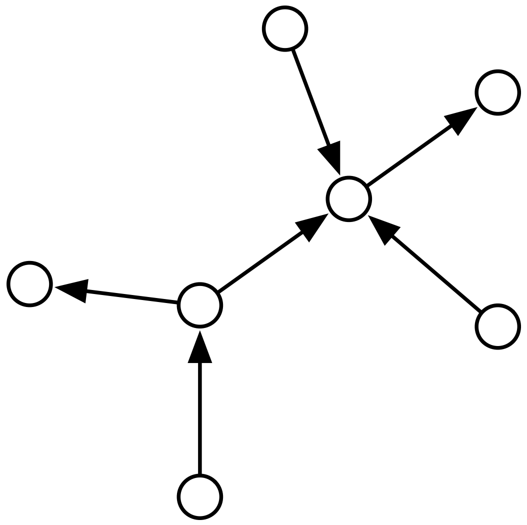
…

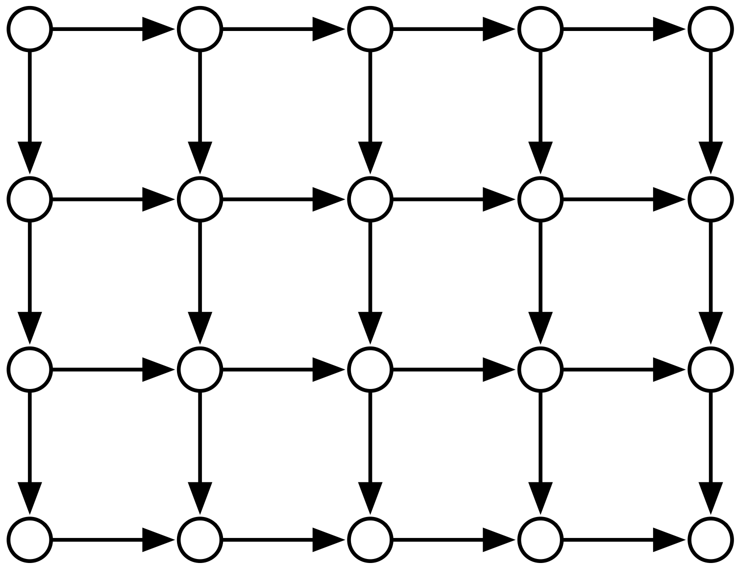




# Sinkless orientation

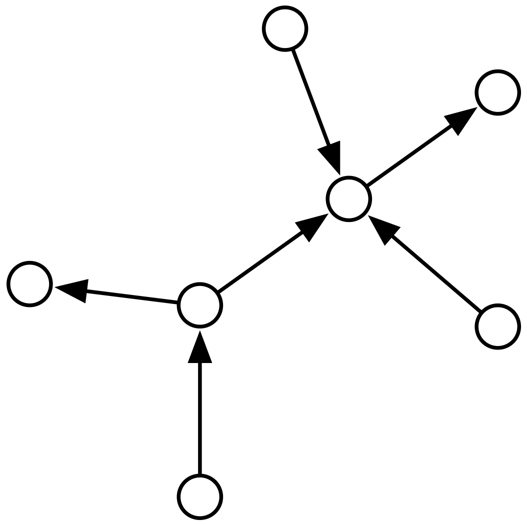
Given a graph
orient the edges
…

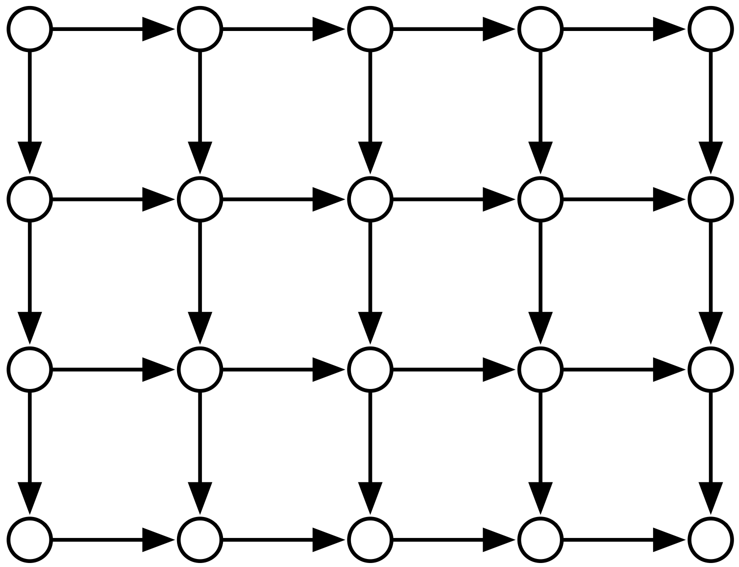




# Sinkless orientation

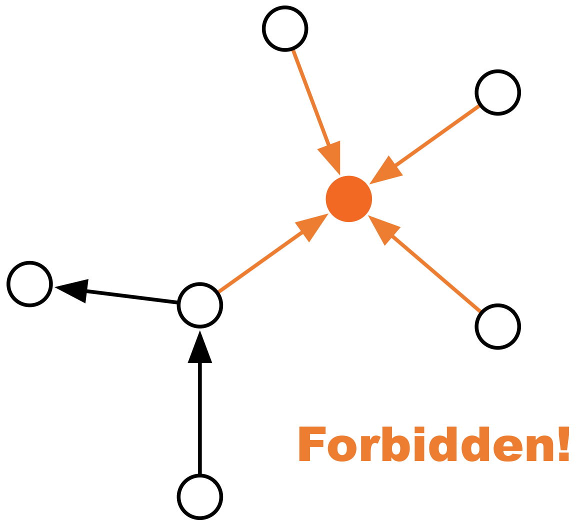
Given a graph
orient the edges
so that nodes with
degree ≥ 3 have
**at least one outgoing edge**

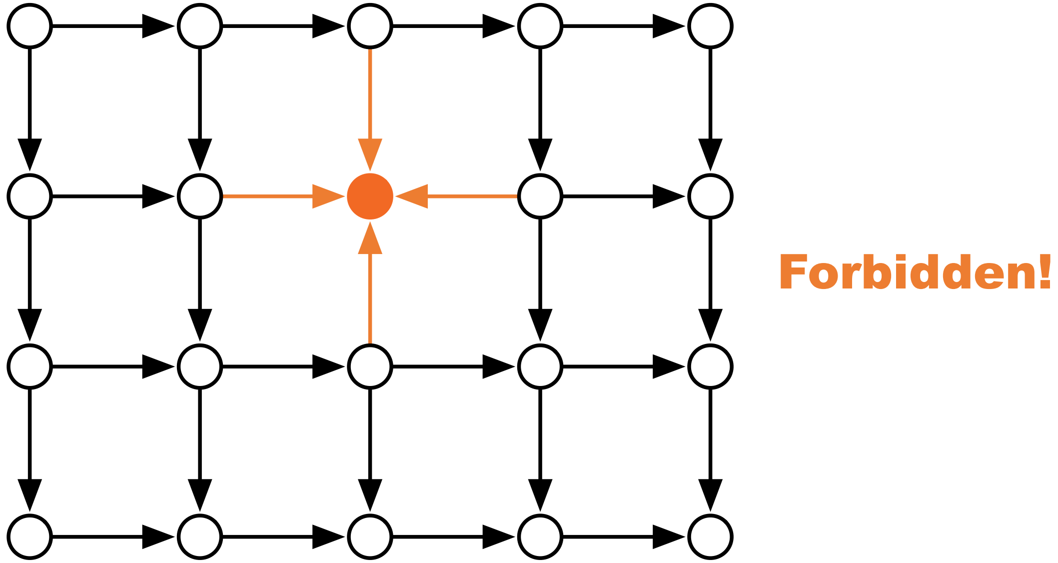




# Sinkless orientation

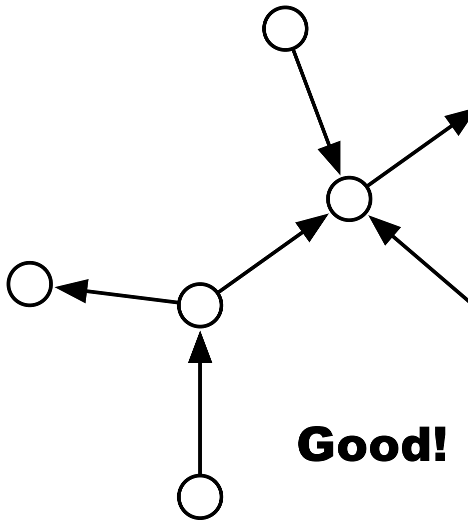
Given a graph
orient the edges
so that nodes with
degree ≥ 3 have
**at least one outgoing edge**

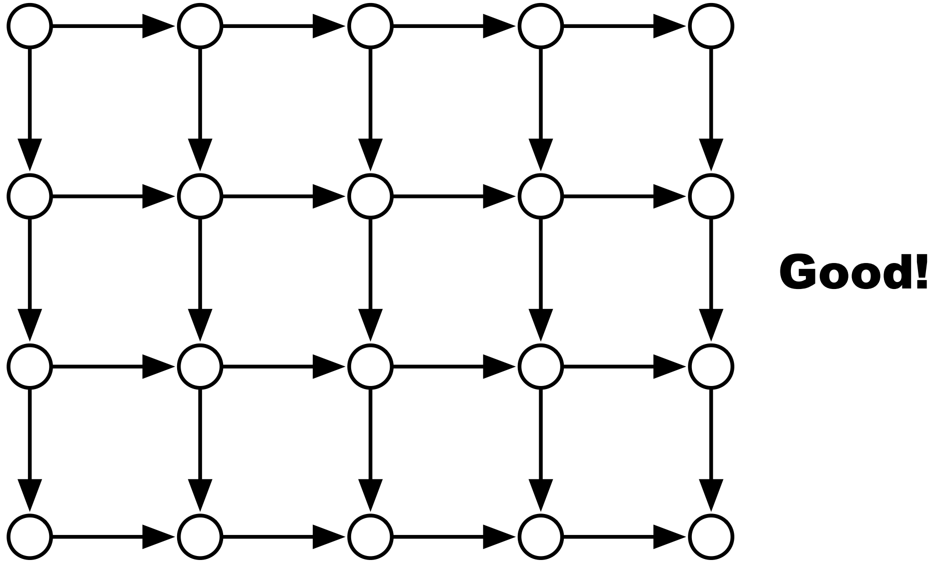


# Sinkless orientation

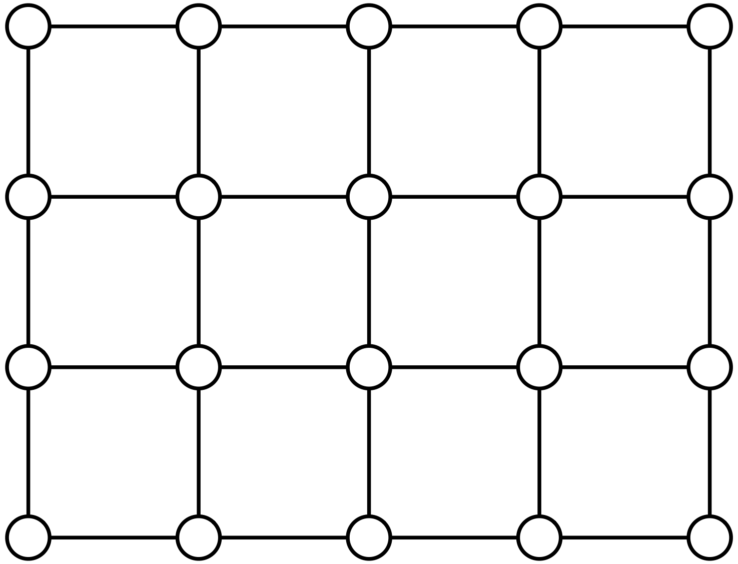
Given a graph
orient the edges
so that nodes with
degree ≥ 3 have
**at least one**
**outgoing edge**



# Centralized setting

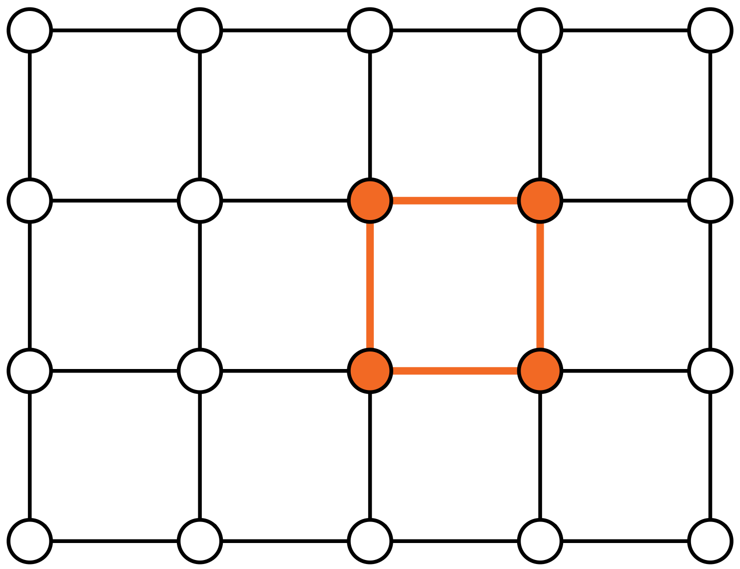
How do we find
a sinkless orientation
in general?



# Centralized setting

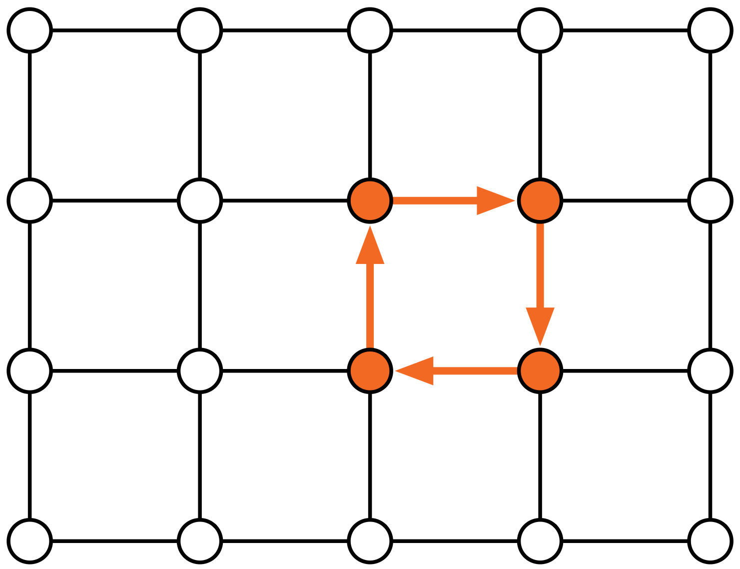
- Choose any **cycle**

  …



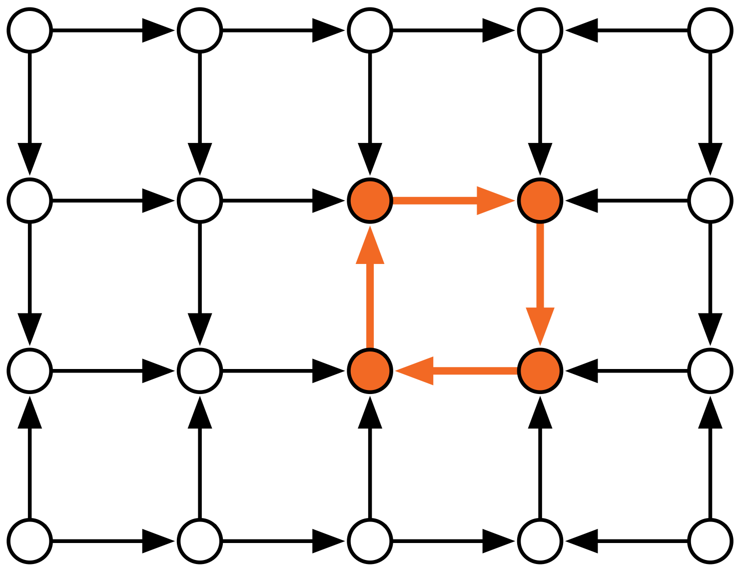
# Centralized setting

- Choose any **cycle**,
  orient it consistently
  …



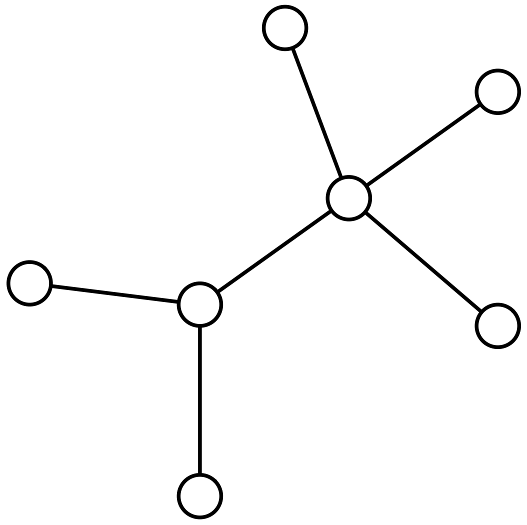
# Centralized setting

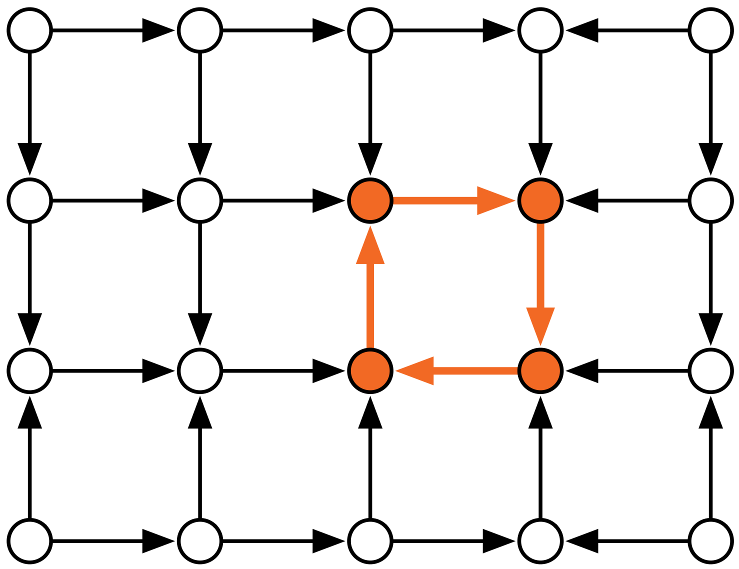
- Choose any **cycle**, orient it consistently, orient everyone towards it



# Centralized setting

- Choose any **cycle**, orient it consistently, orient everyone towards it
- Otherwise …

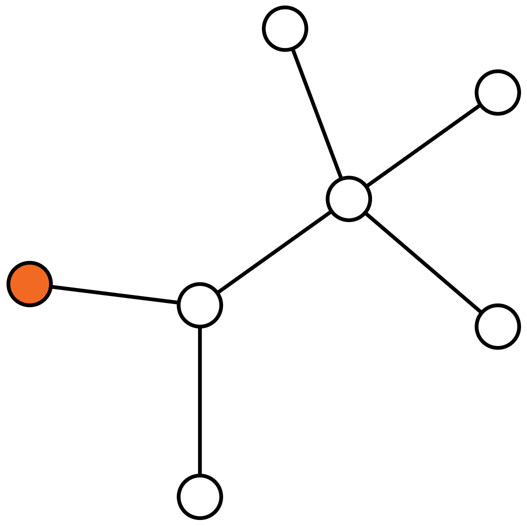


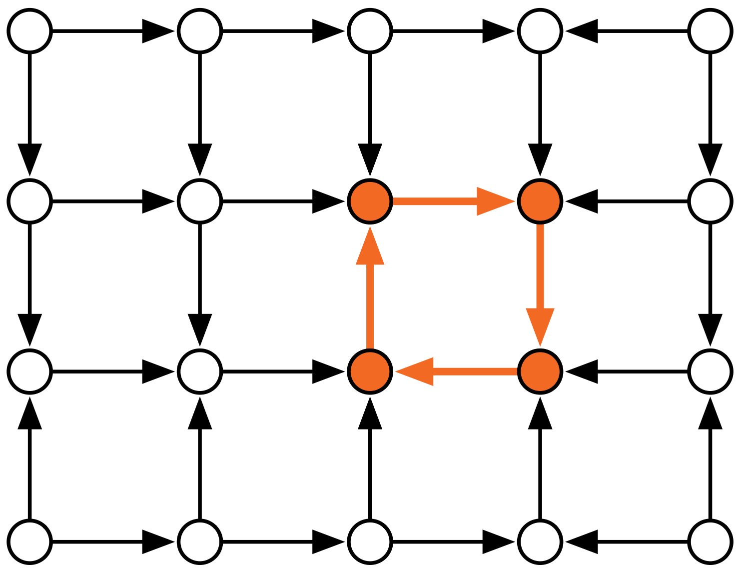


# Centralized setting

- Choose any **cycle**, orient it consistently, orient everyone towards it
- Otherwise choose any **leaf node**

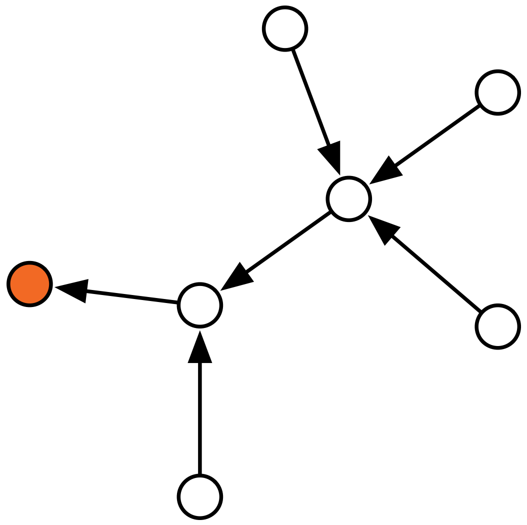
…

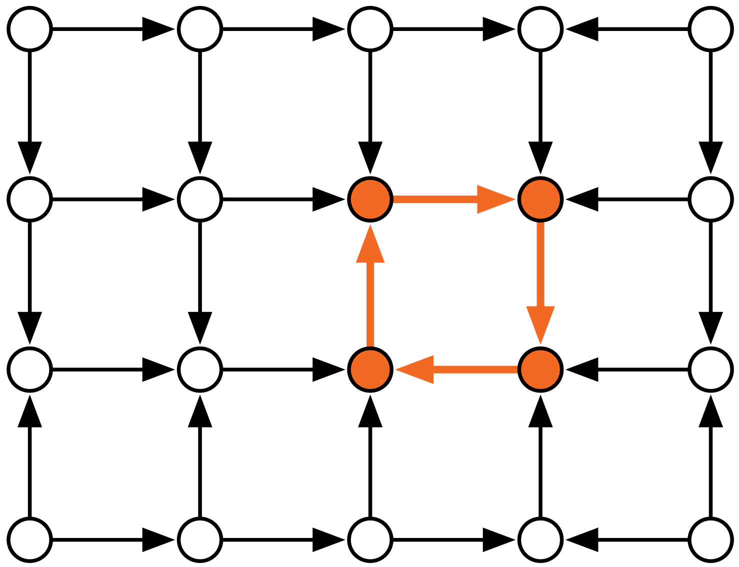




# Centralized setting

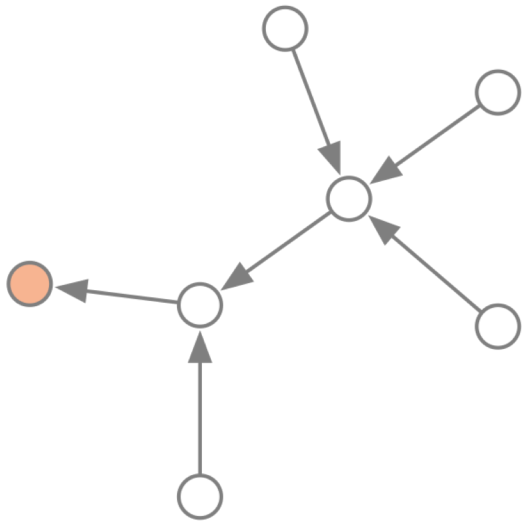
- Choose any **cycle**, orient it consistently, orient everyone towards it
- Otherwise choose any **leaf node**, orient everyone towards it





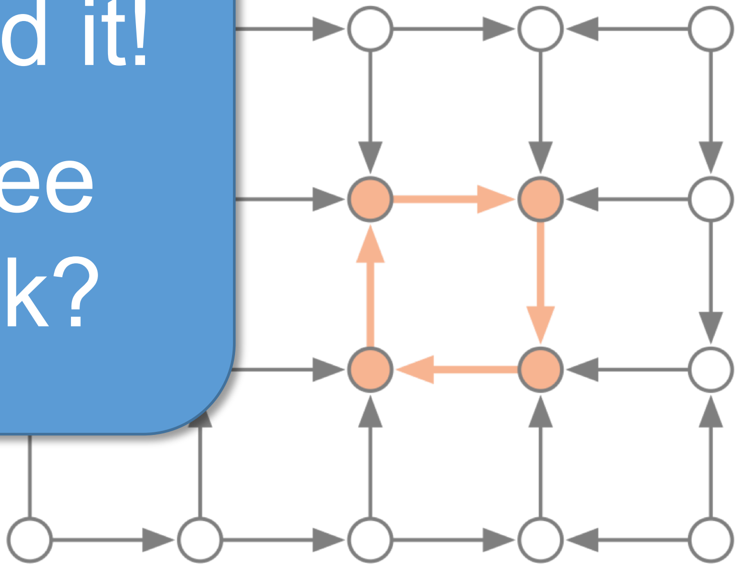
# Centralized setting

- Choose any **cycle**, orient it consistently, orient everyone towards it
- Otherwise choose any **leaf node**, orient everyone towards it



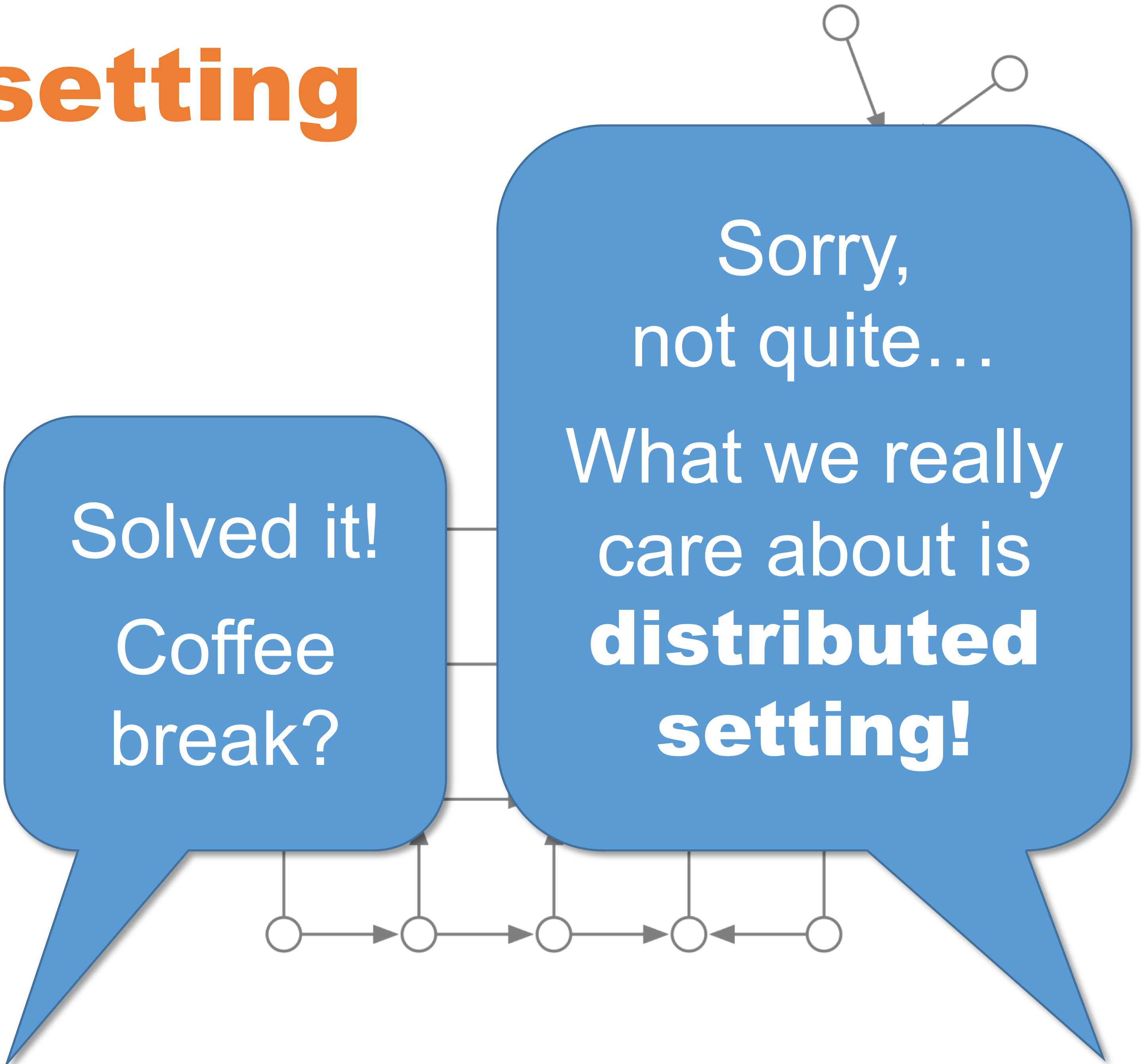
Solved it!

Coffee break?



# Centralized setting

- Choose any **cycle**, orient it consistently, orient everyone towards it
- Otherwise choose any **leaf node**, orient everyone towards it



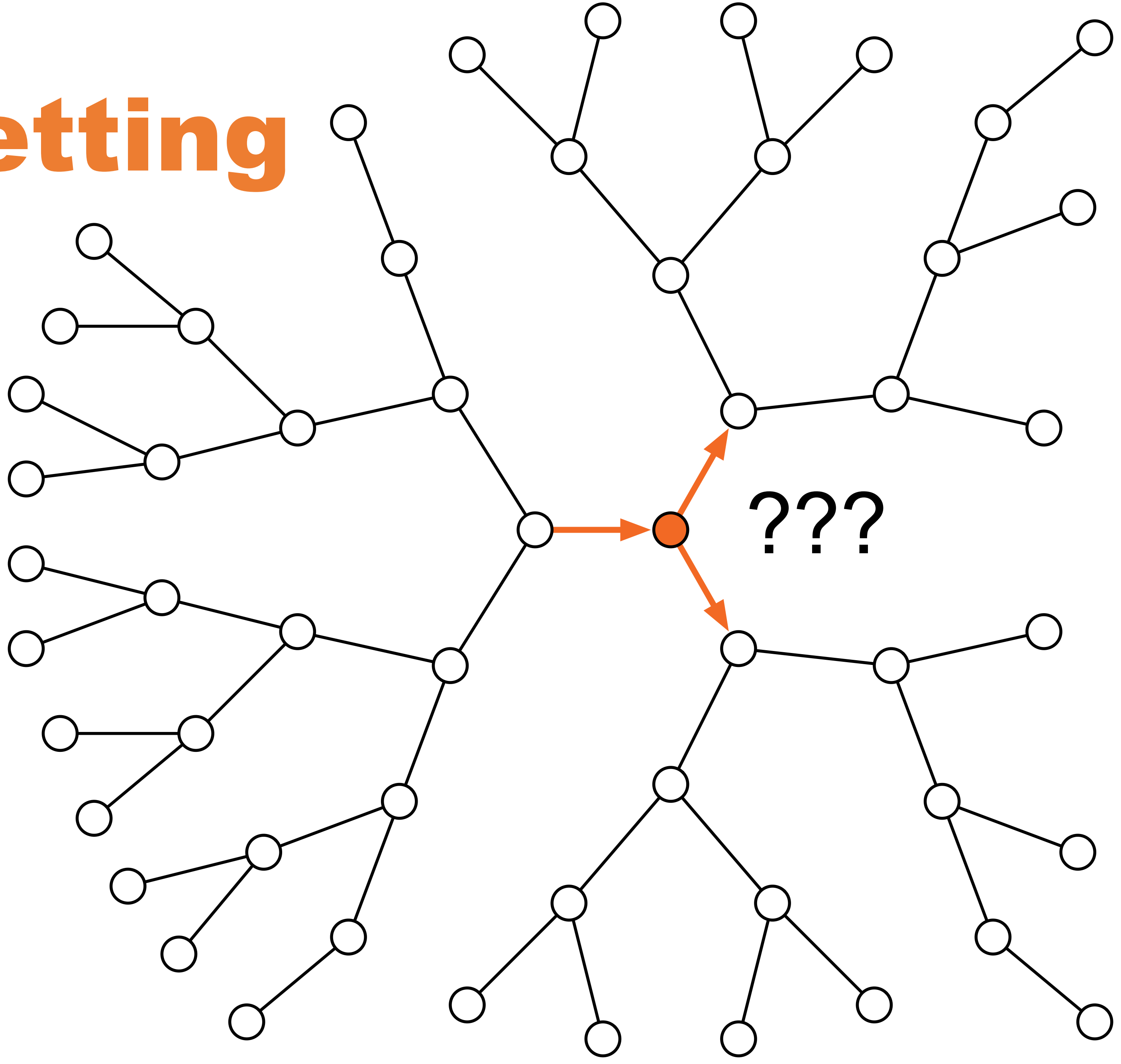
# Distributed setting

You are a node in the middle of a very large graph

# Distributed setting

You are a node in the middle of a very large graph

How to orient your **incident edges**?

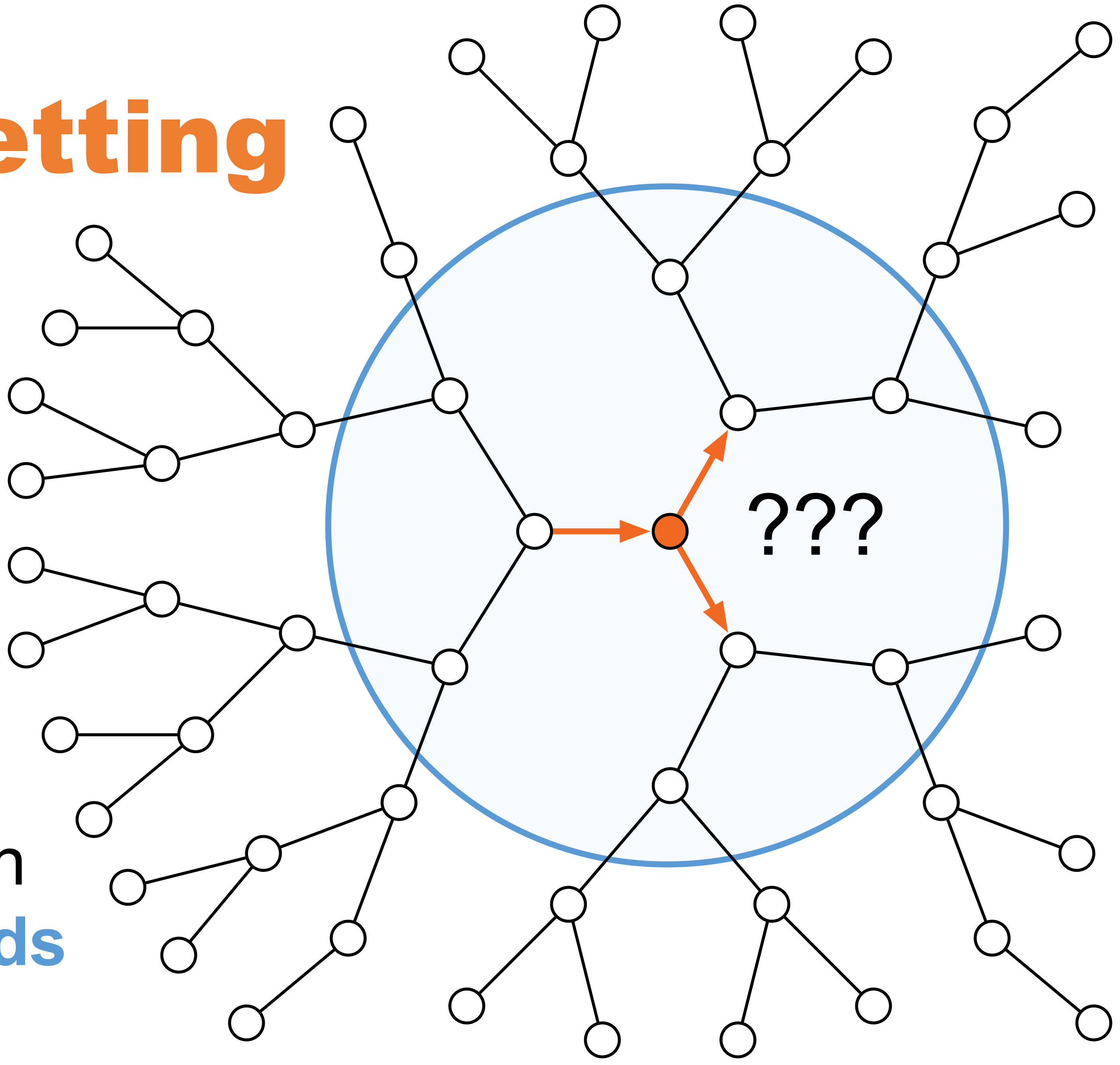


# Distributed setting

You are a node in the middle of a very large graph

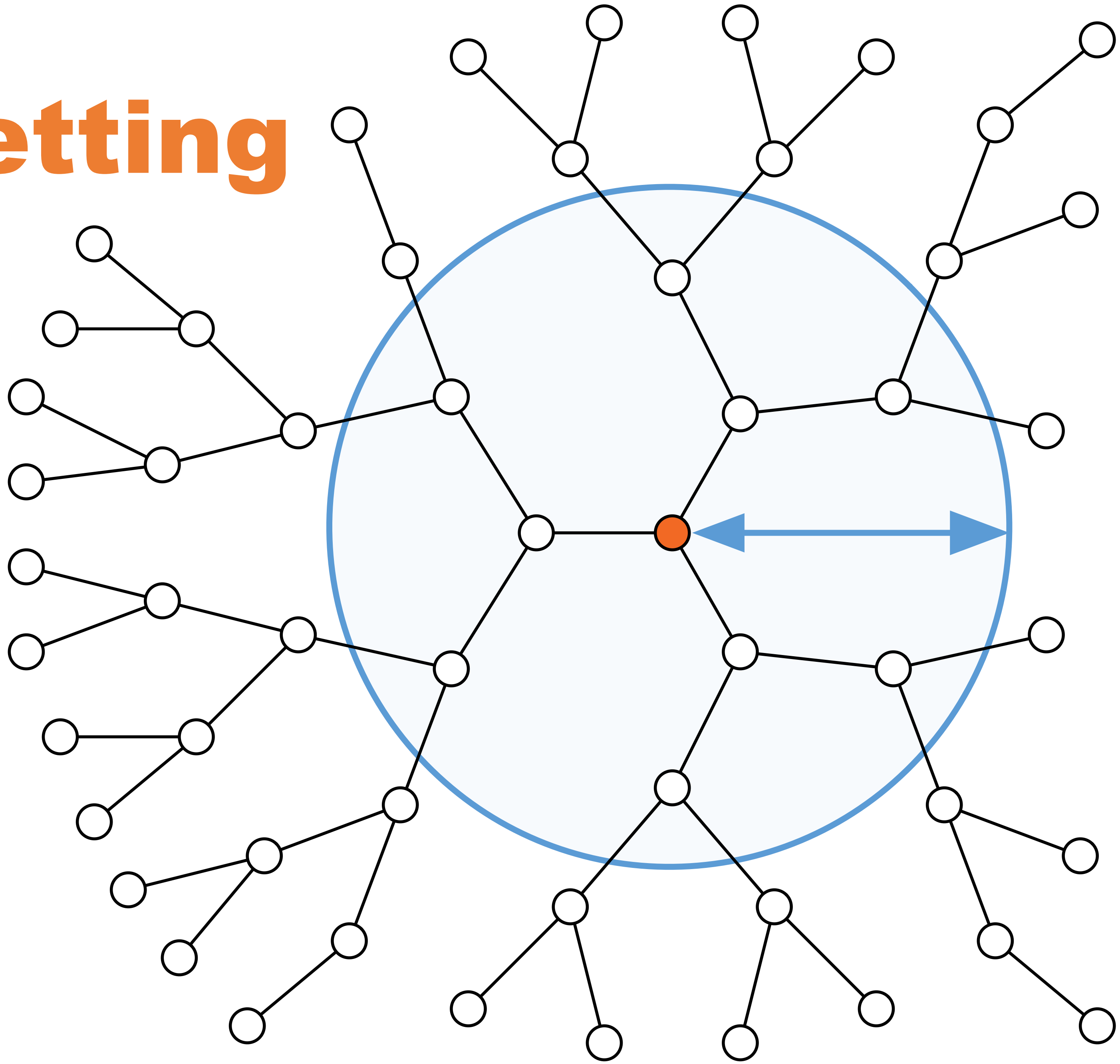
How to orient your **incident edges**?

No global coordination, everyone acting based on their **local neighborhoods**



# Distributed setting

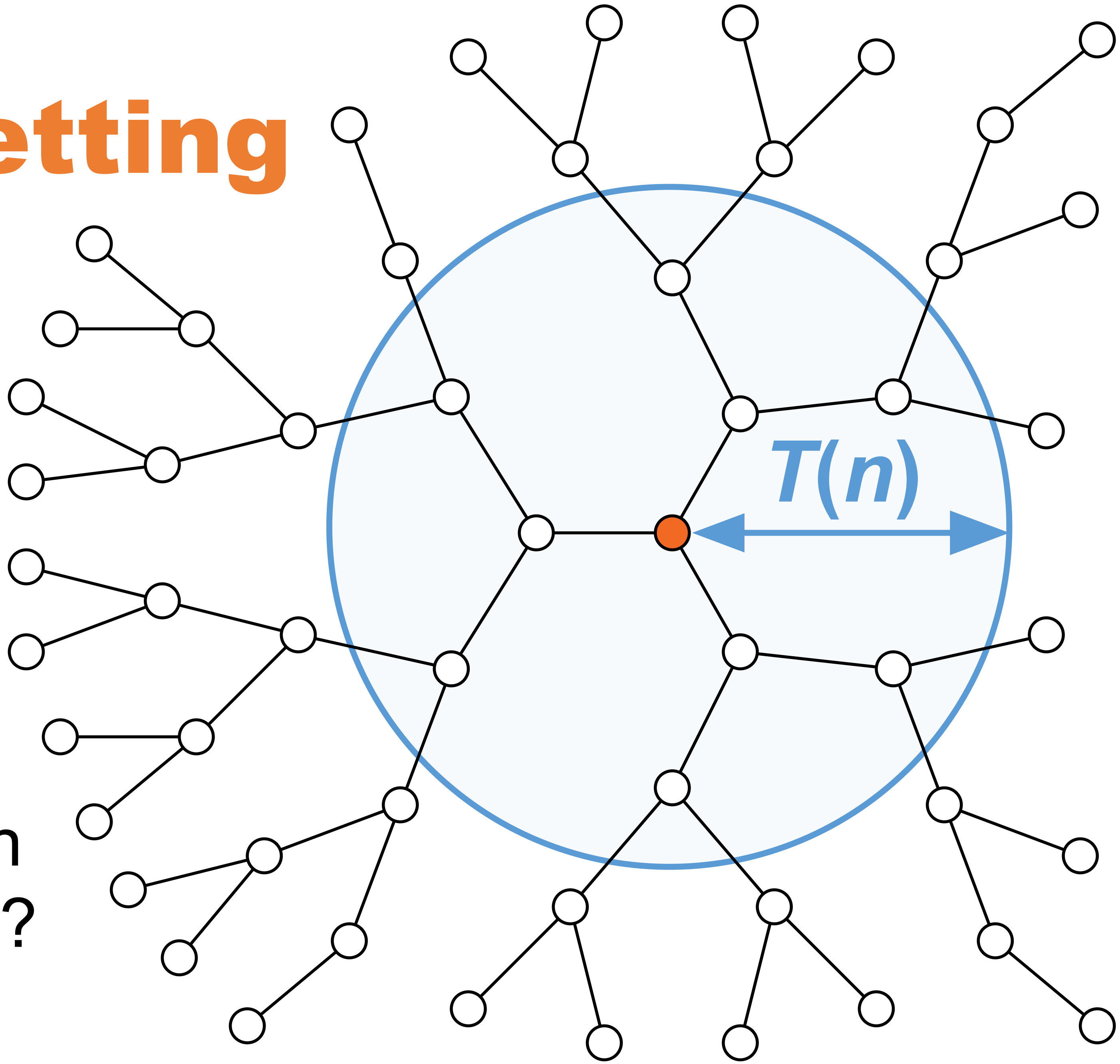
Key question: **locality**
= how far do you need
to see?



# Distributed setting

Key question: **locality**
= how far do you need to see?

What is the smallest $T(n)$ such that you can solve sinkless orientation if everyone acts based on their $T(n)$-neighborhoods?



# Why do we care?

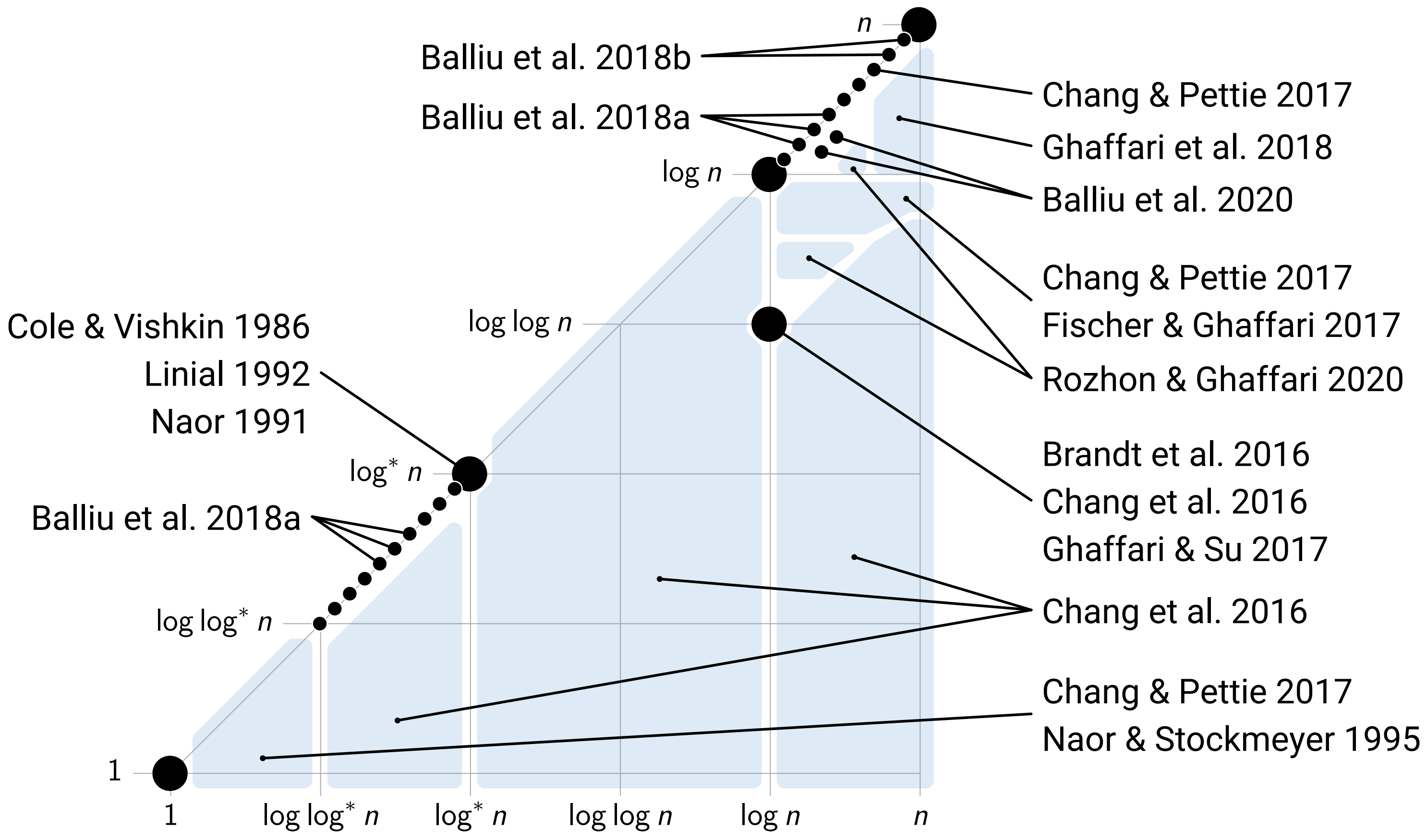
- Sinkless orientation plays a key role in **understanding distributed computational complexity**
  - cf. 3SAT in classical complexity theory

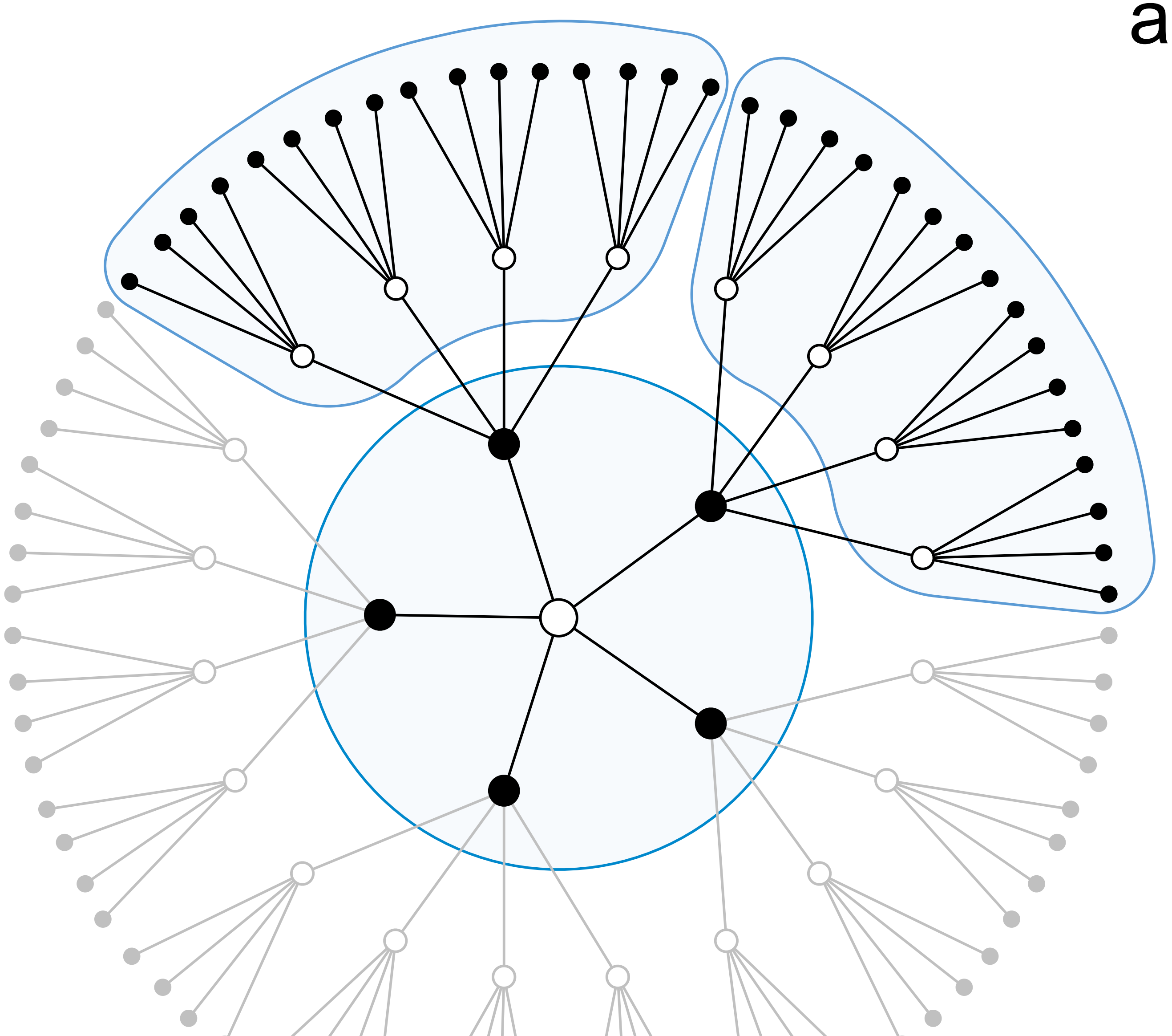
# Why do we care?

- Sinkless orientation plays a key role in **understanding distributed computational complexity**
  - cf. 3SAT in classical complexity theory
- Many problems are **at least as hard** as sinkless orientation—example: *Δ-coloring, Lovász local lemma*
- Many problems are **equivalent** to sinkless orientation—example: *degree splitting*

Study of **sinkless orientation** led to the development of modern distributed complexity theory

We now know the **landscape of locality**





Study of **sinkless orientation** also led to the discovery of the **round elimination technique**

Round elimination has been used to resolve major open questions — e.g. FOCS 2019 best paper

# What is known?

- **Deterministic LOCAL model:**
  - nodes labeled with unique identifiers from 1 … poly($n$)
  - all nodes **simultaneously in parallel** pick their output based on all information in their $T(n)$-radius neighborhood
- **Sinkless orientation:**

$$\boxed{T(n) = \Theta(\log n)}$$

# What is known?

- **Deterministic LOCAL model:**
  - nodes labeled with unique identifiers fr
  - all nodes **simultaneously in parallel** p
    based on all information in their $T(n)$-ra
- **Sinkless orientation:**

Tricky part:
lower bound!

$$\boxed{T(n) = \Theta(\log n)}$$

# Without unique identifiers

- Use the round elimination technique
- Deduce **deterministic $\Omega(\log n)$** lower bound

Pretty simple, but it does not tell us anything about the LOCAL model…

How can we handle unique IDs?

# With unique identifiers

- Use the round elimination technique
- …
- …
- …
- …
- Deduce **deterministic $\Omega(\log n)$** lower bound

# With unique identifiers

- Use the round elimination technique
- Analyze **randomized algorithms**
- …
- …
- …
- Deduce **deterministic $\Omega(\log n)$** lower bound

Wait, what, why??

# With unique identifiers

- Use the round elimination technique
- Analyze **randomized algorithms**
- Careful analysis of failure probabilities (nontrivial)
- …
- …
- Deduce **deterministic $\Omega(\log n)$** lower bound

# With unique identifiers

- Use the round elimination technique
- Analyze **randomized algorithms**
- Careful analysis of failure probabilities (nontrivial)
- Deduce **randomized $\Omega(\log \log n)$** lower bound
- …
- Deduce **deterministic $\Omega(\log n)$** lower bound

# With unique identifiers

- Use the round elimination technique
- Analyze **randomized algorithms**
- Careful analysis of failure probabilities (nontrivial)
- Deduce **randomized $\Omega(\log \log n)$** lower bound
- Apply general gap theorems (heavyweight machinery)
- Deduce **deterministic $\Omega(\log n)$** lower bound

# Our contribution: made simple

- Use the round elimination technique
- ~~Analyze **randomized algorithms**~~
- ~~Careful analysis of failure probabilities (nontrivial)~~
- ~~Deduce **randomized Ω(log log *n*)** lower bound~~
- ~~Apply general gap theorems (heavyweight machinery)~~
- Deduce **deterministic Ω(log *n*)** lower bound

# Round elimination

- Function “re” that **maps problems to problems**
- *Theorem:* If the locality of $X$ is $T$, then the locality of re($X$) is $T - 1$
- Works in many models of distributed computing, as long as we have **“independence”**

Coming back to this in a minute!

# Round elimination: application

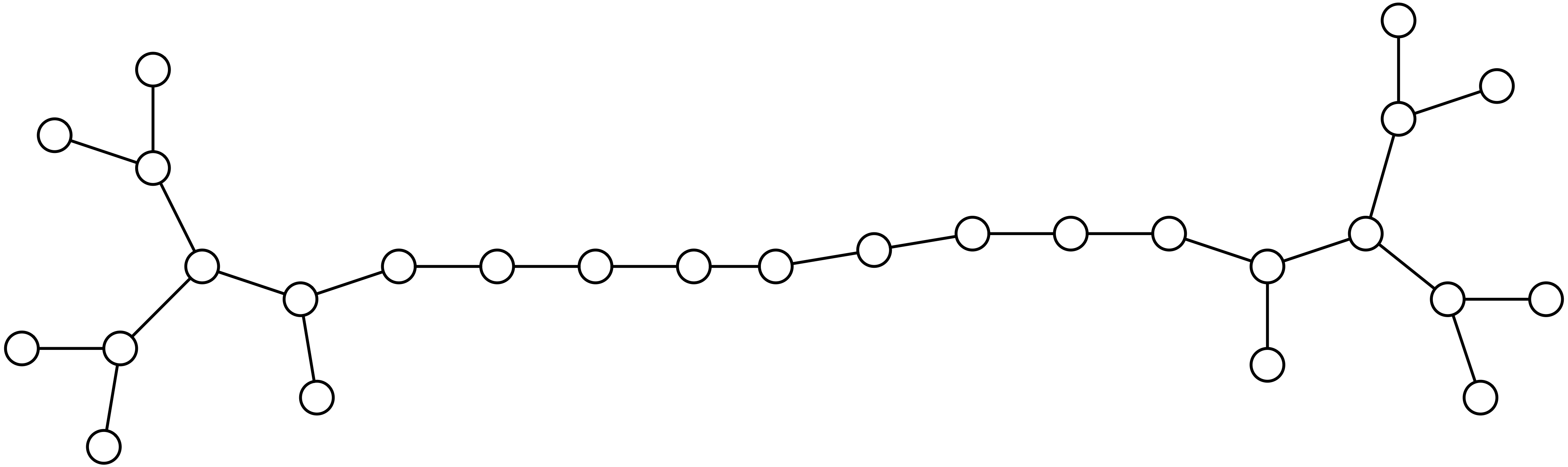
- Start with $X$ = **sinkless orientation**
- **Assume $X$ has locality $T(n) = o(\log n)$**
- Observe that $\mathbf{re}(X) = X$
- We could iteratively speed up sinkless orientation algorithms all the way to 0 locality!
- But it can’t be solved with 0 locality (easy to check)
- **Therefore the assumption must be wrong!**

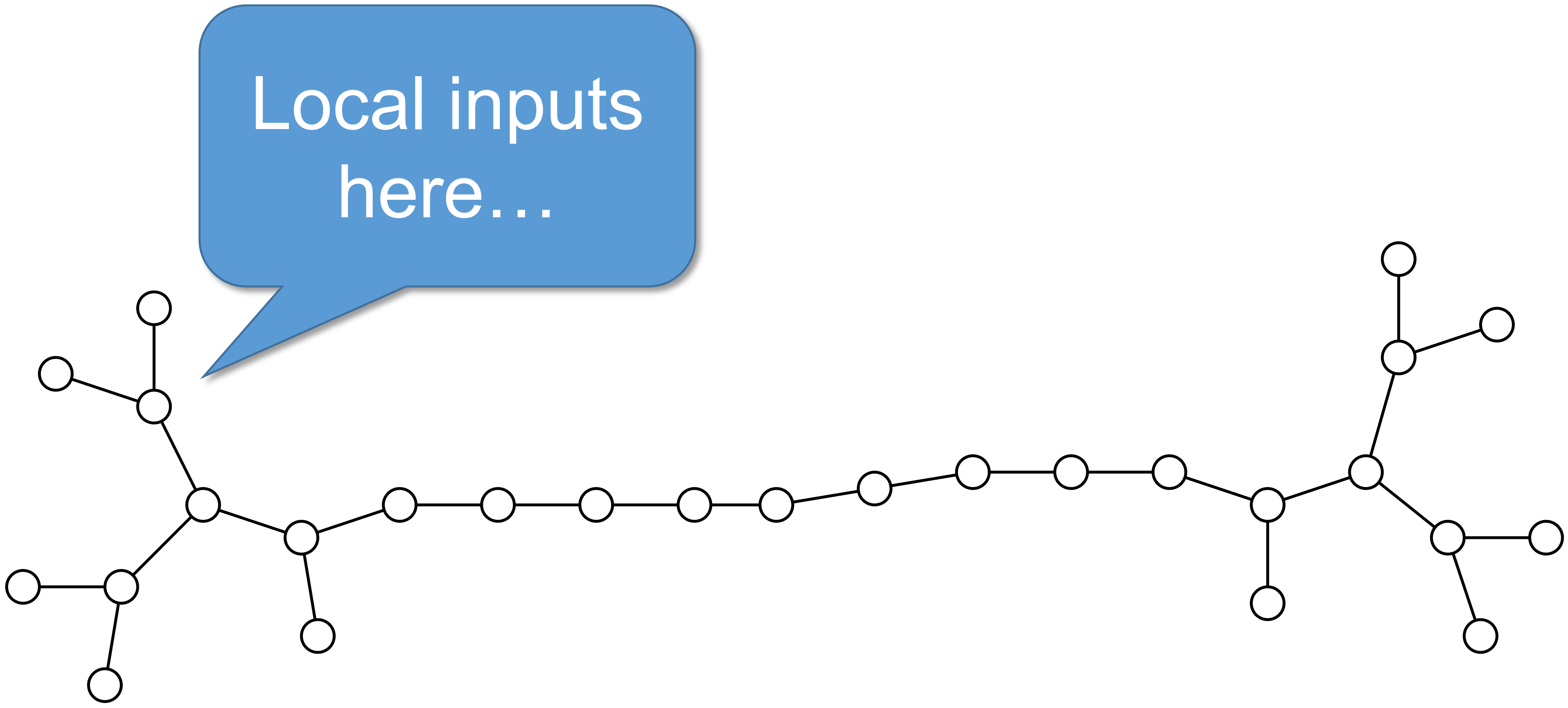
# Round elimination

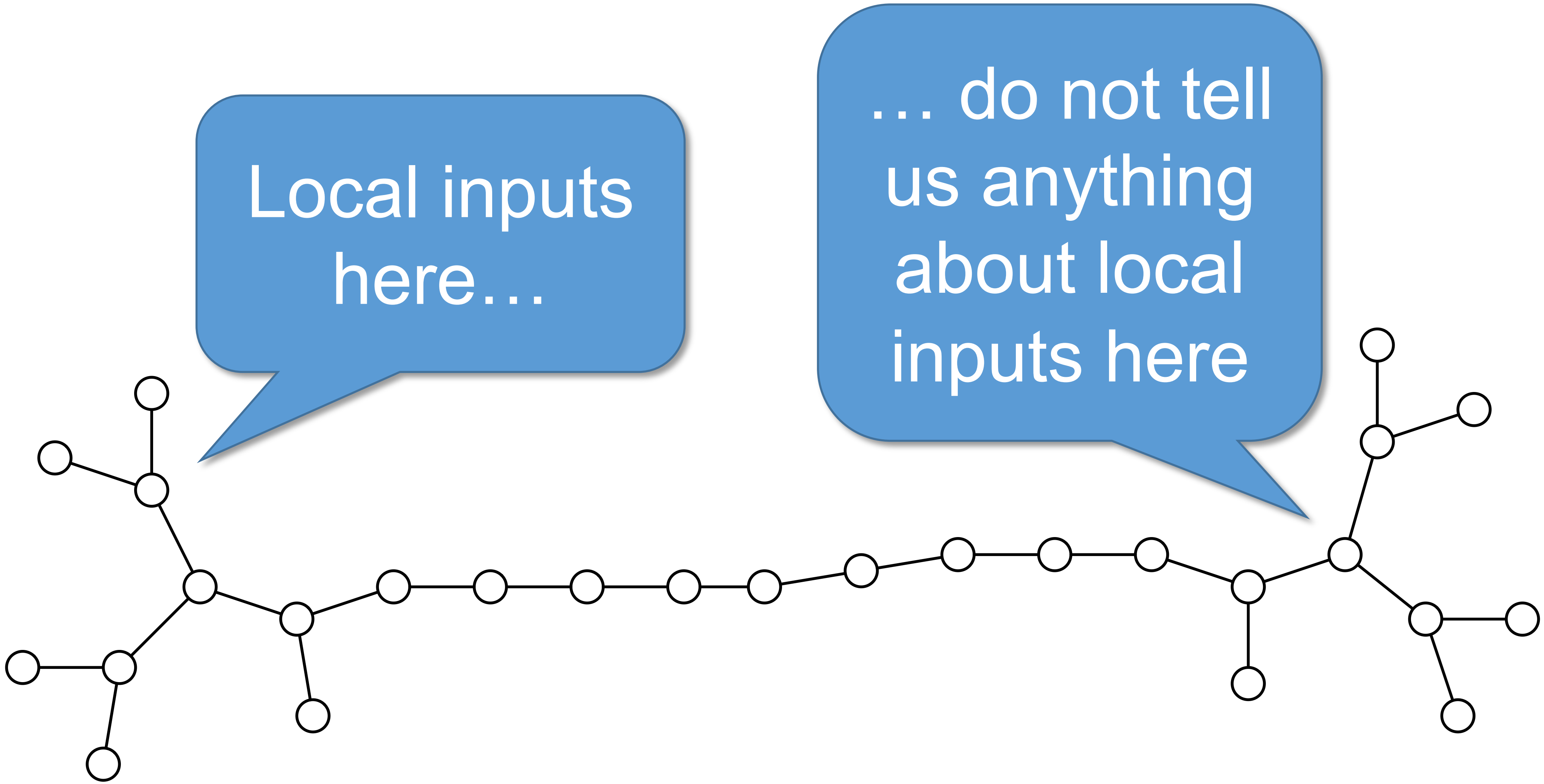
- Function "re" that **maps problems to problems**
- *Theorem:* If the locality of $X$ is $T$,
  then the locality of re($X$) is $T - 1$
- Works in many models of distributed computing,
  as long as we have **"independence"**

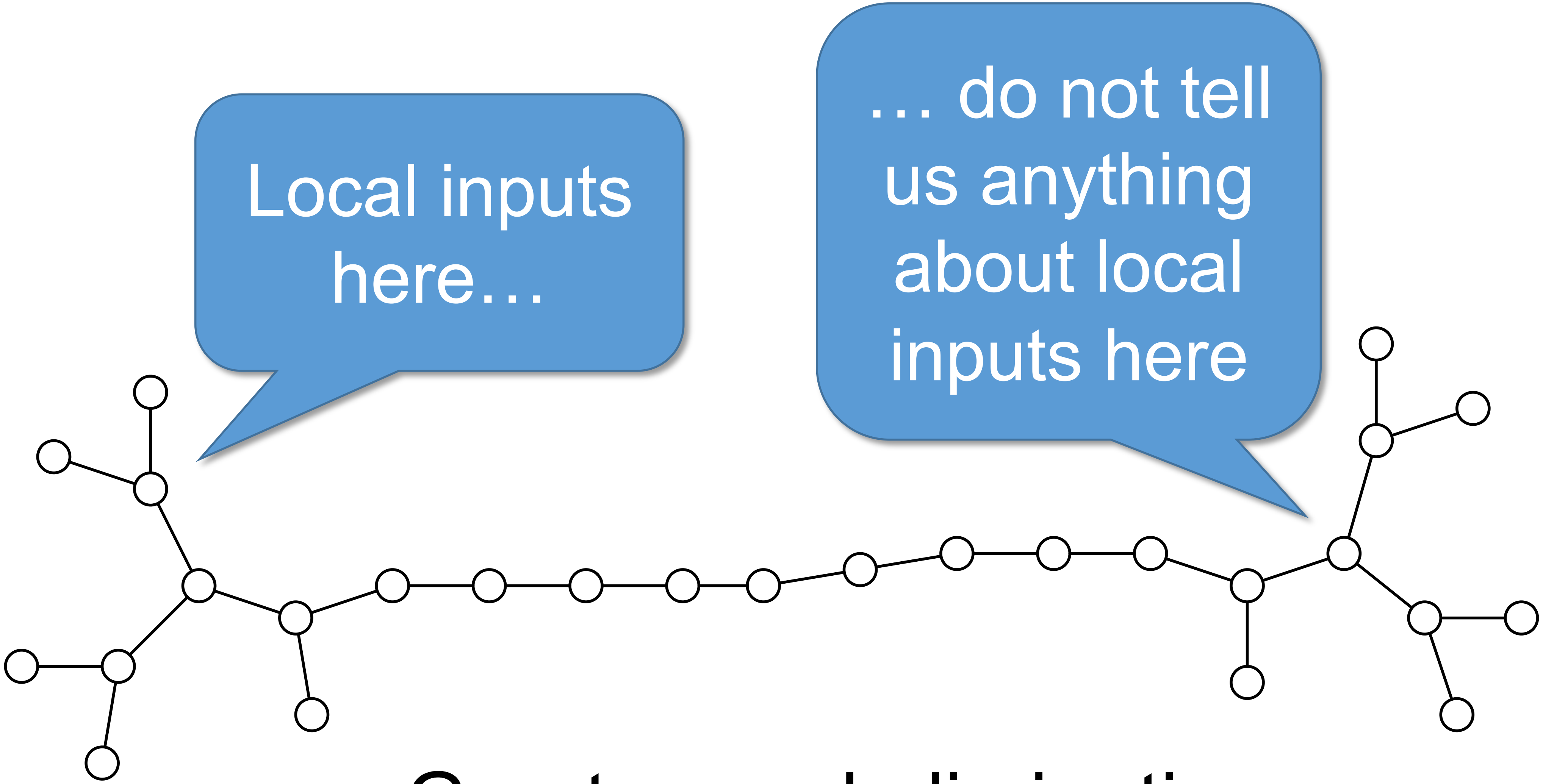
Now getting
back to this!

# Let’s consider a large network…

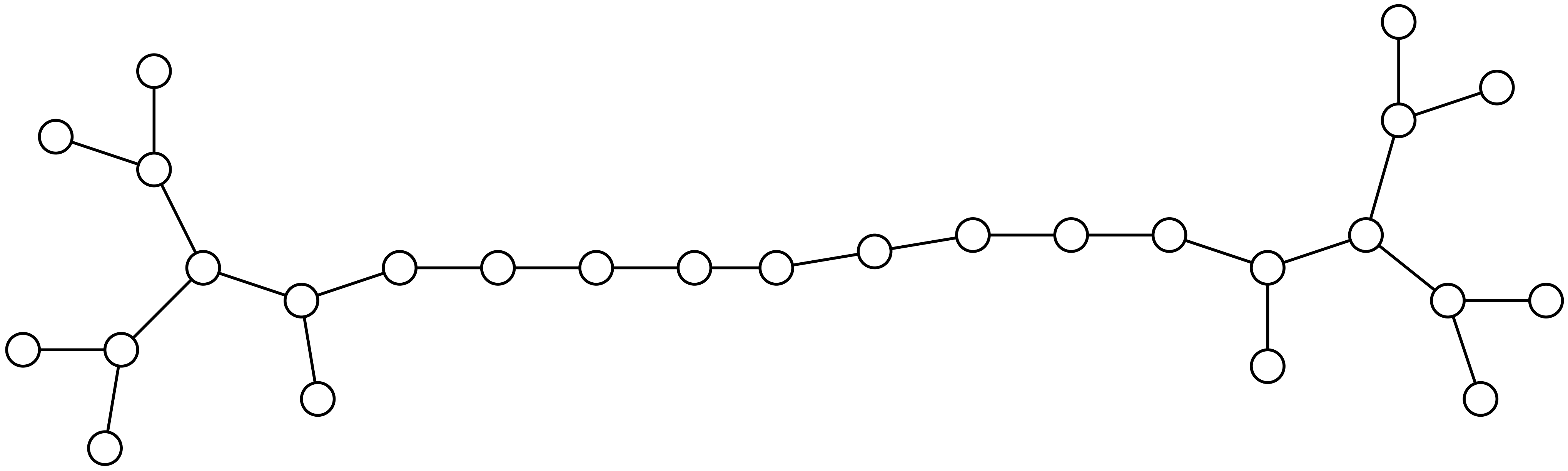


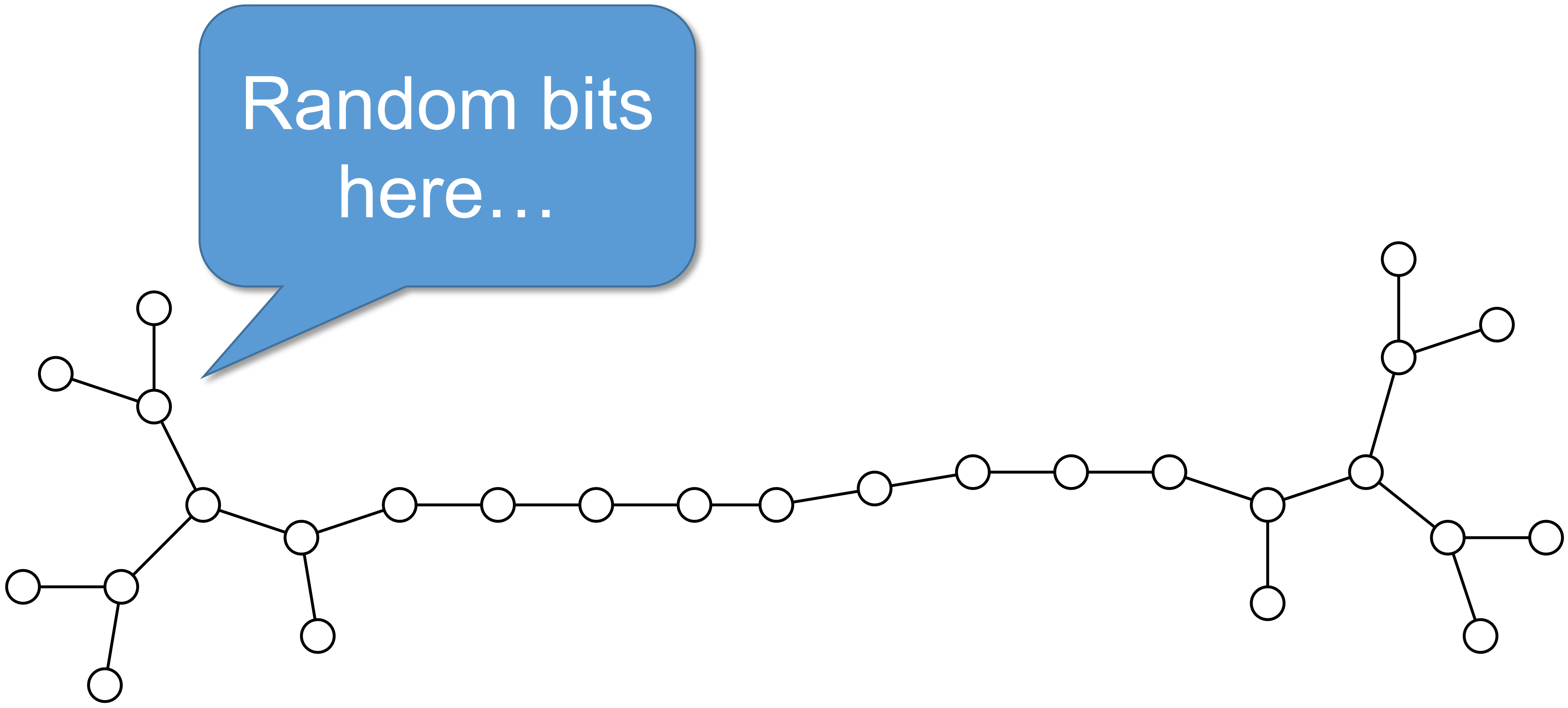

Local inputs here…

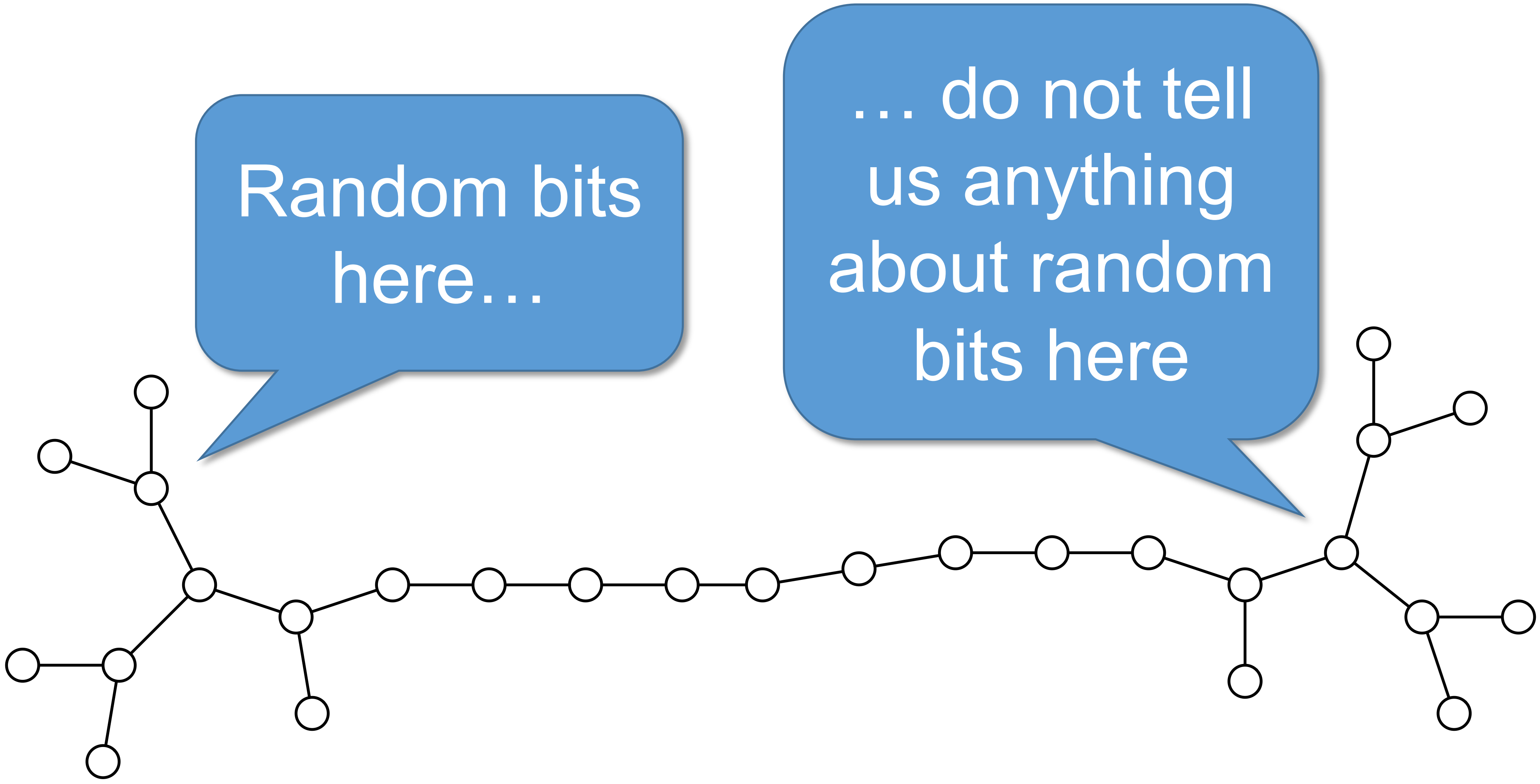

Local inputs here…
… do not tell us anything about local inputs here



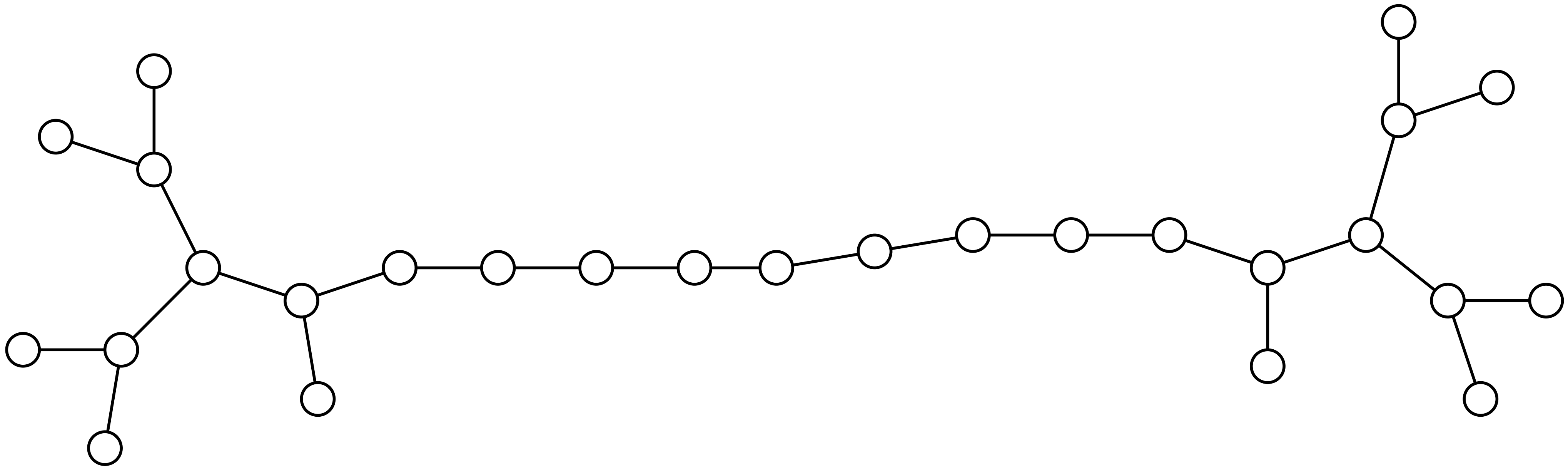

Great, round elimination can handle **local inputs!**

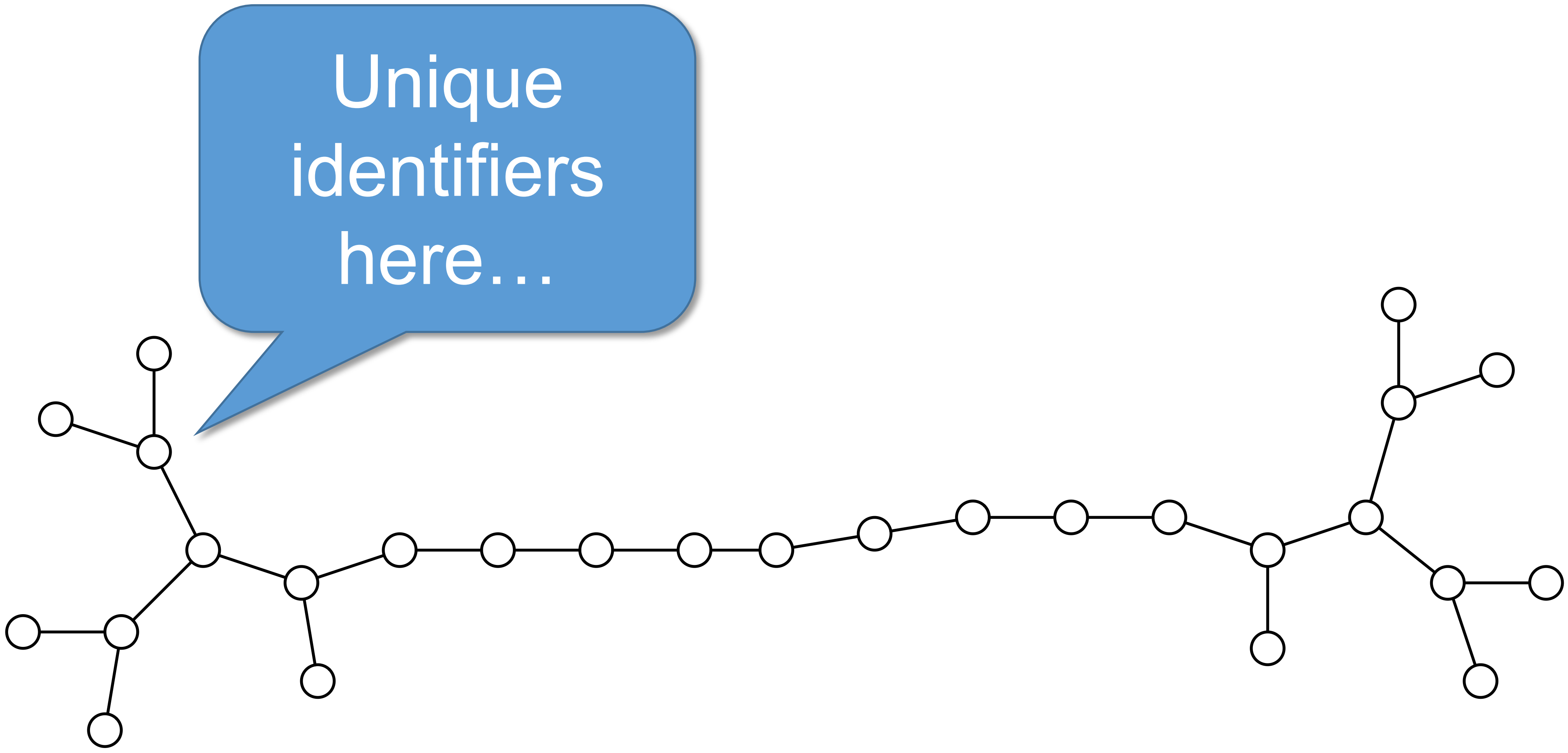


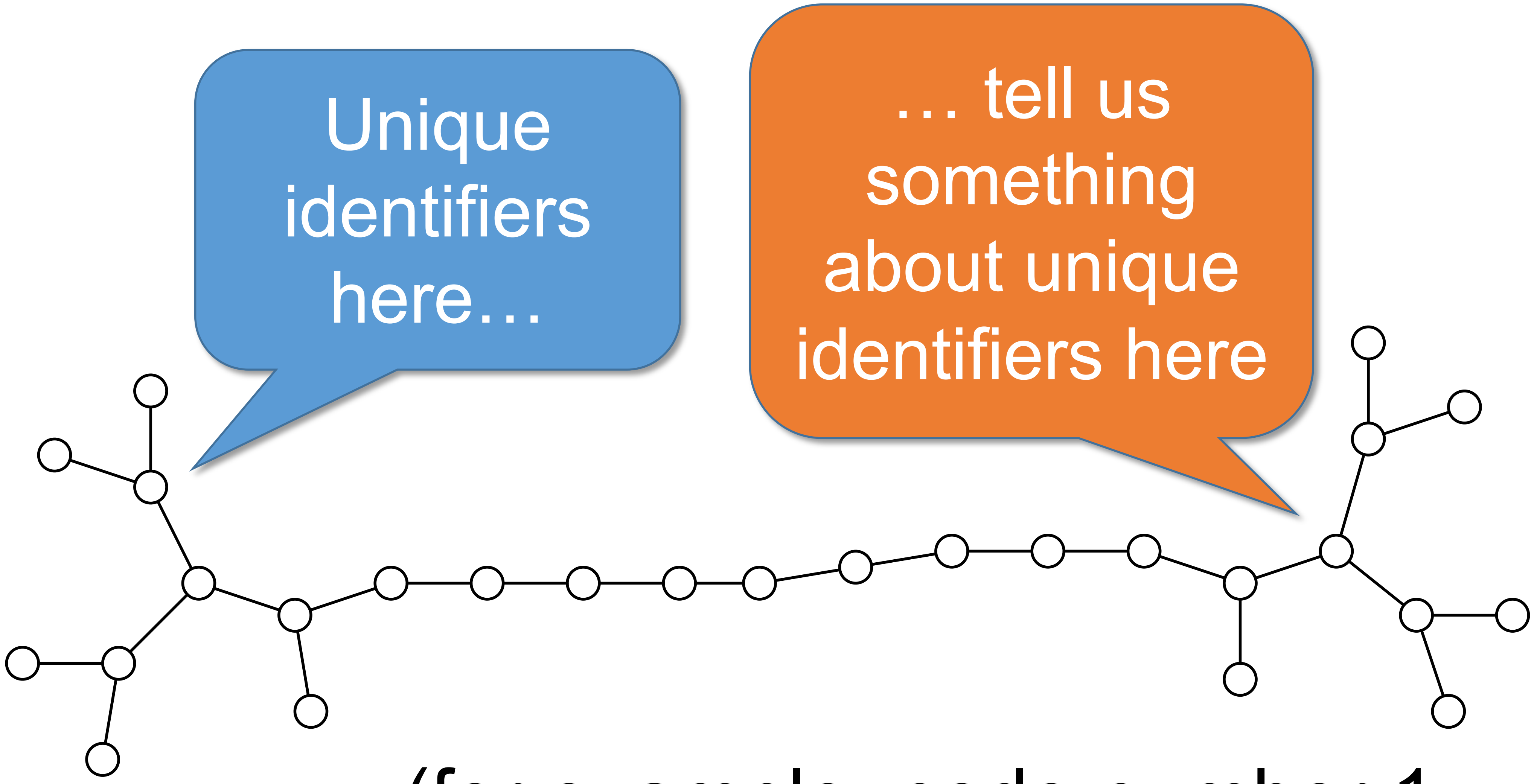

Random bits here…


Random bits here…
… do not tell us anything about random bits here

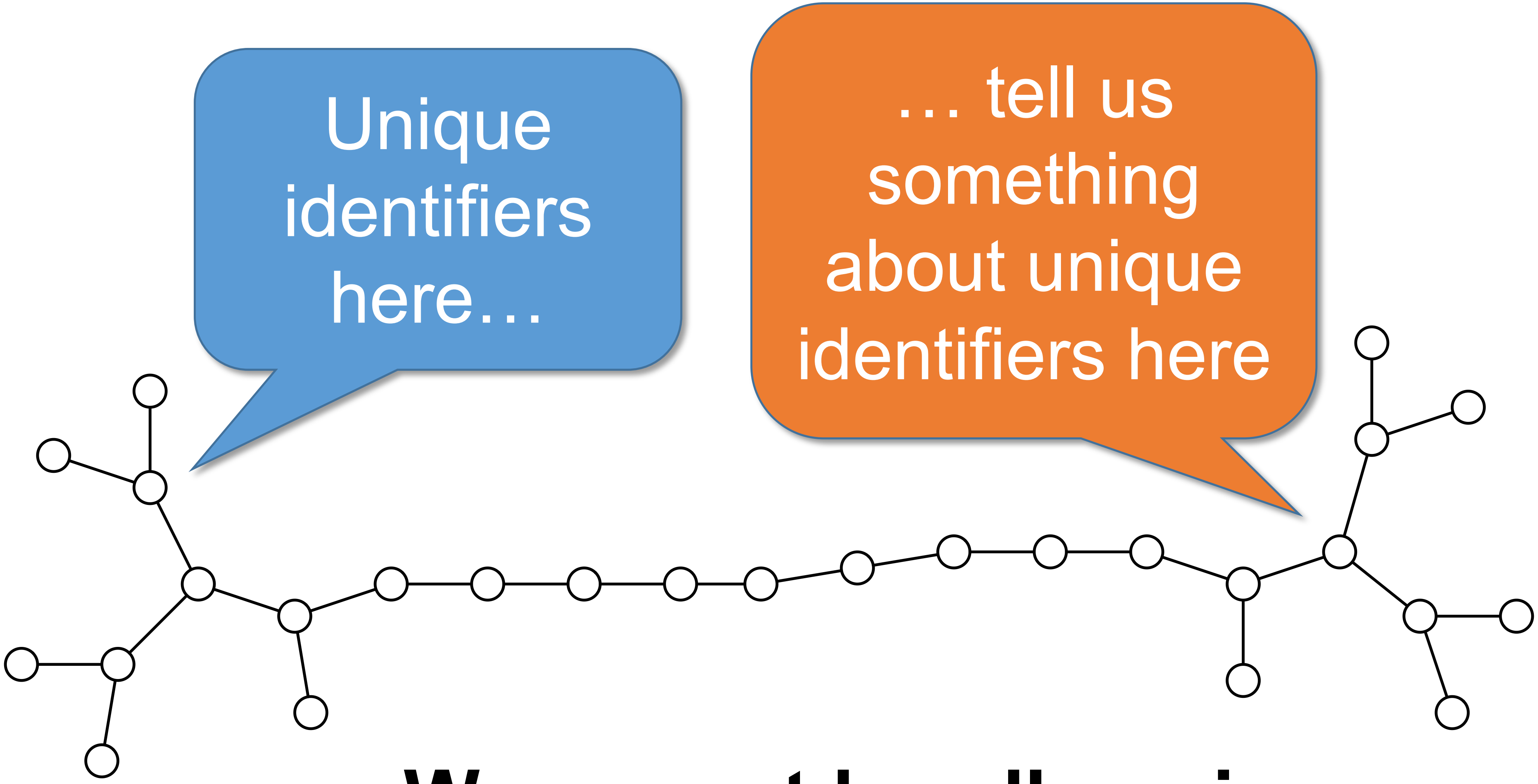

Great, round elimination can handle **randomized algorithms!**




Unique identifiers here…




(for example, node number 1 cannot appear in both places!)




**We cannot handle unique identifiers in round elimination!**

**We cannot handle unique identifiers in round elimination!**

# Our main contribution:
# a very simple workaround

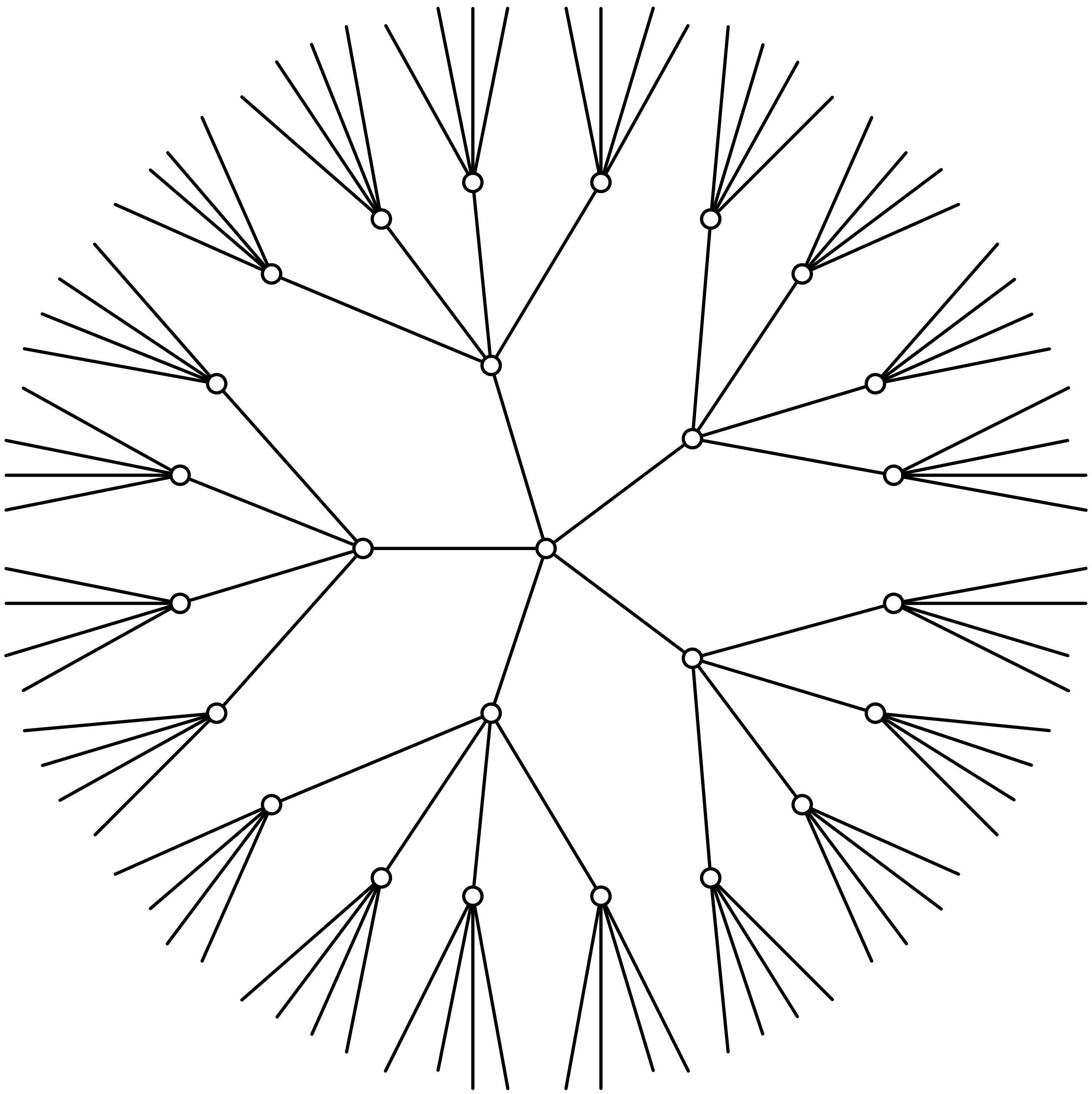
**We cann dle unique**
**identifier ound elimination!**

# Key insight: supported model

- **Not good:** fixed input, fixed unique identifiers
  - it is trivial to solve anything if we know everything
- **Not good:** fixed input, adversarial unique identifiers
  - no independence, cannot use round elimination
- **Good:**
  - fix a **support graph** $G$ in advance
  - fix some unique identifiers in $G$
  - reveal some adversarial **subgraph** $H$ of $G$

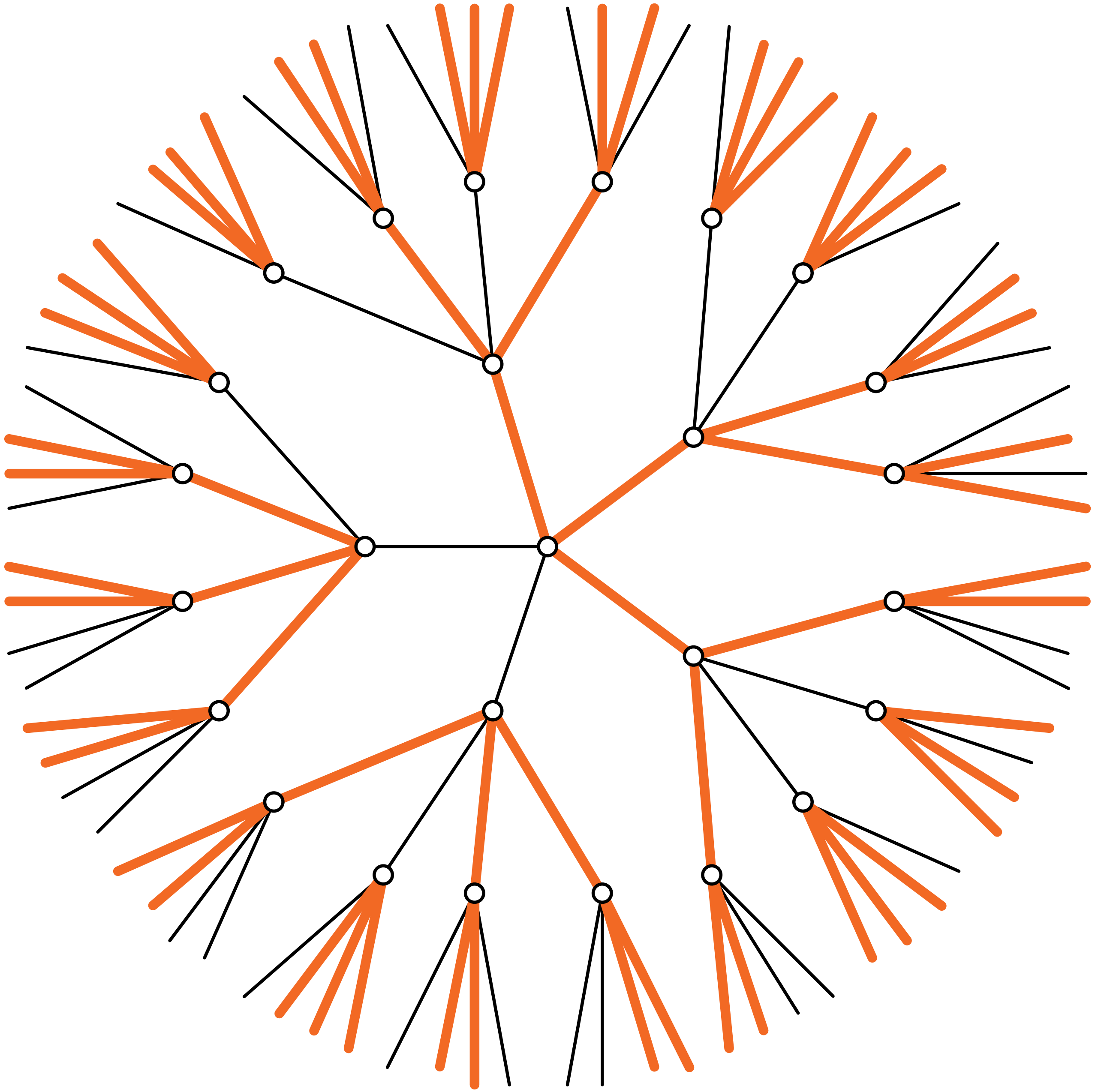
# Supported model

- Fix a 5-regular graph *G*
  - structure + identifiers globally known
  - you could precompute anything related to *G*



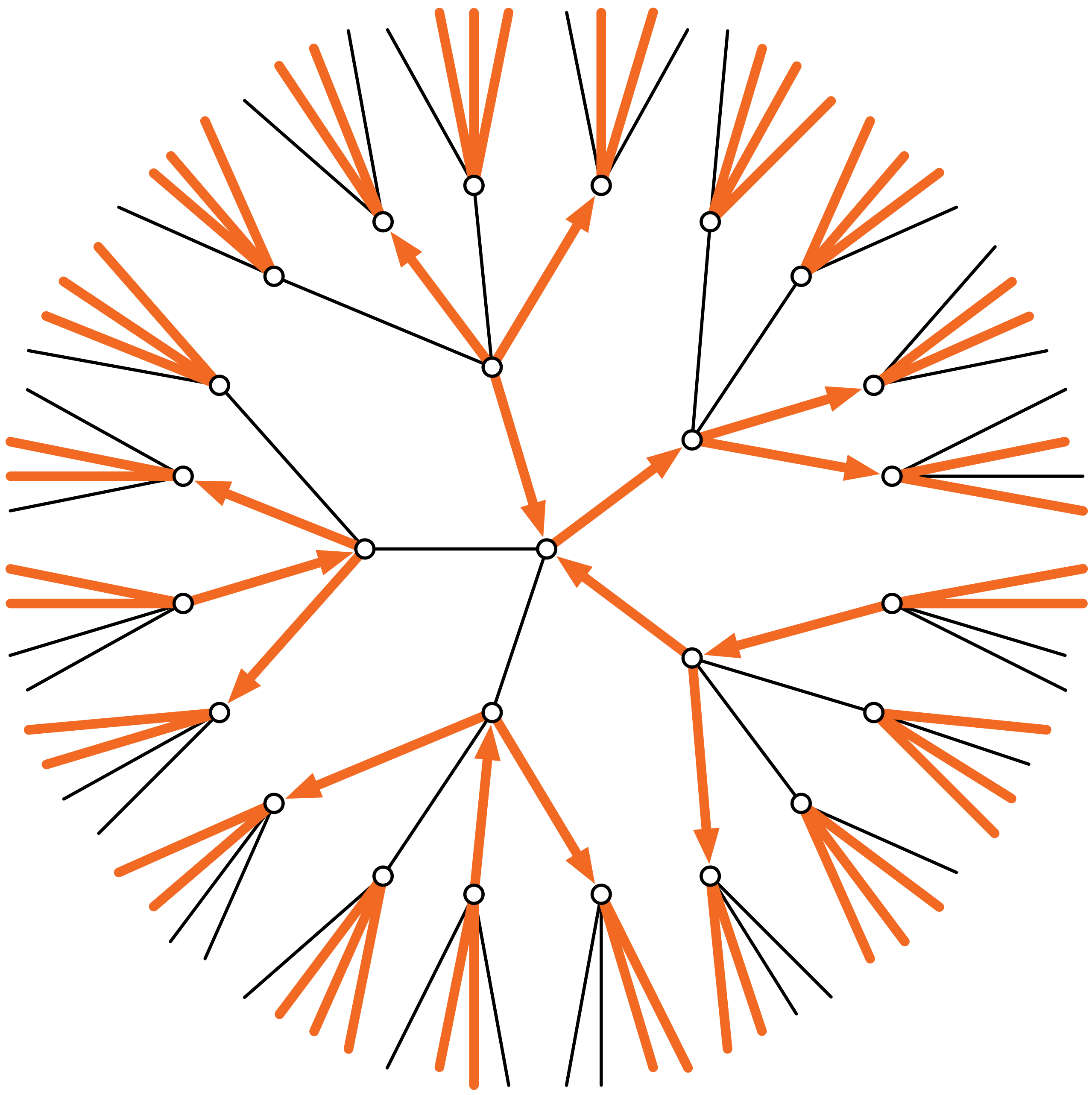
# Supported model

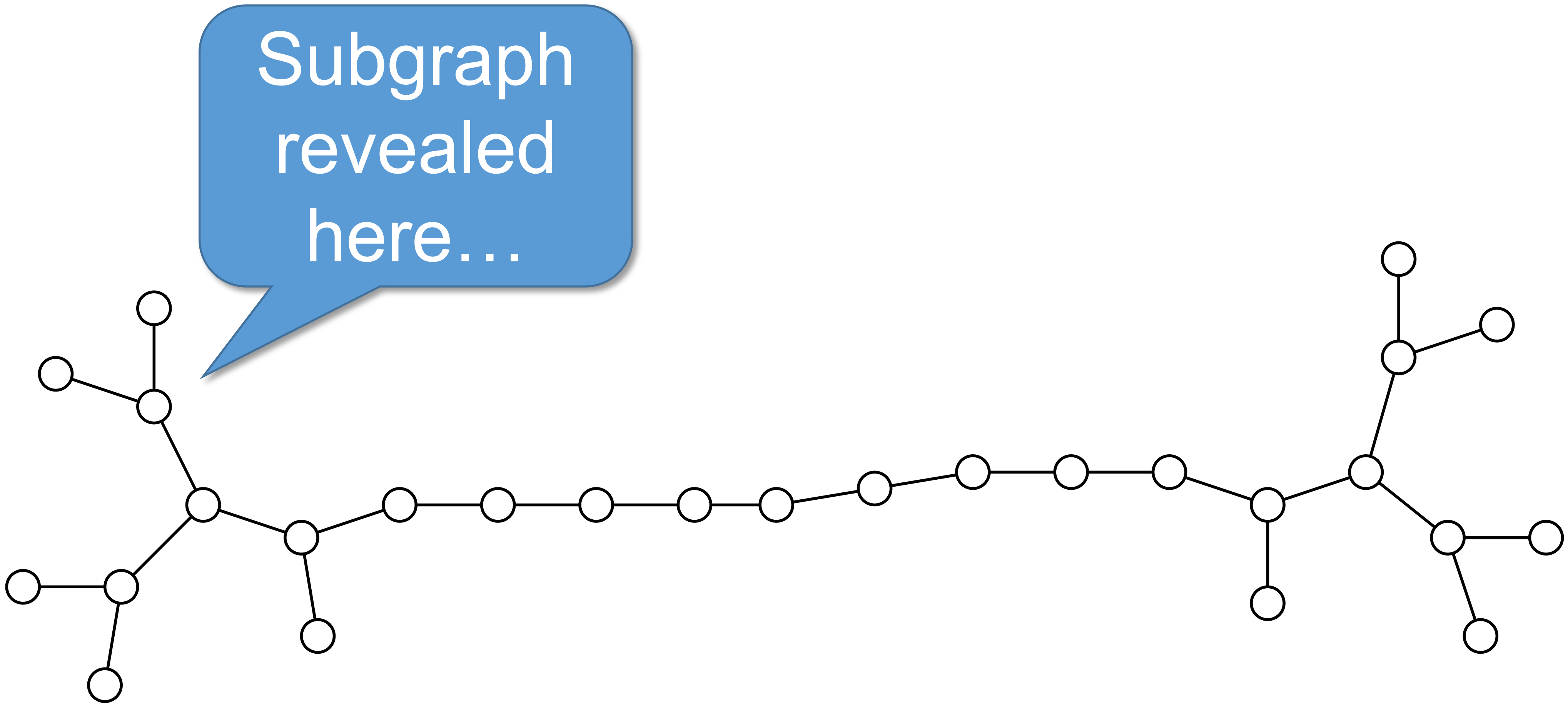
- Fix a 5-regular graph $G$
  - structure + identifiers globally known
- Reveal a **3-regular subgraph $H$**
  - only locally known

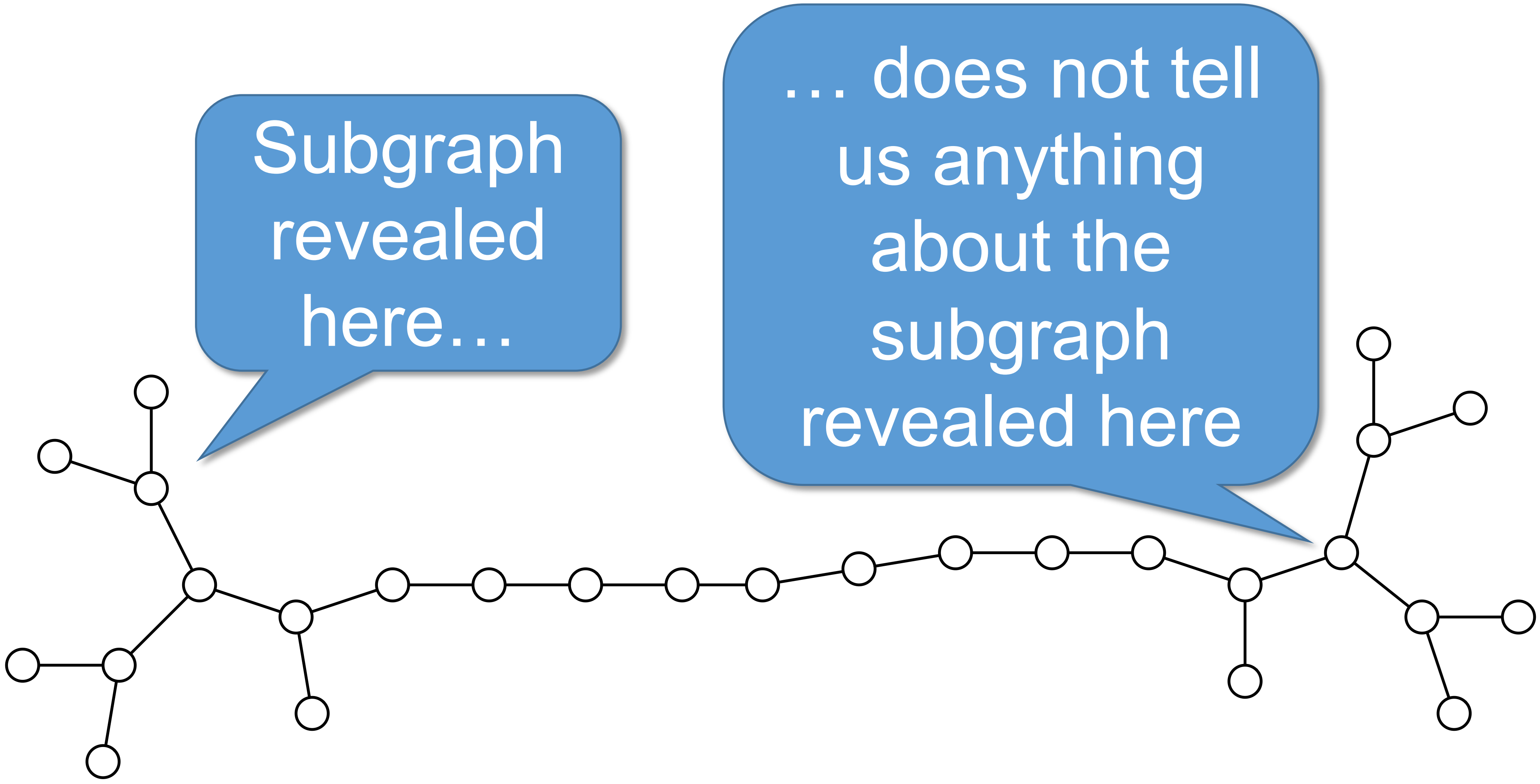


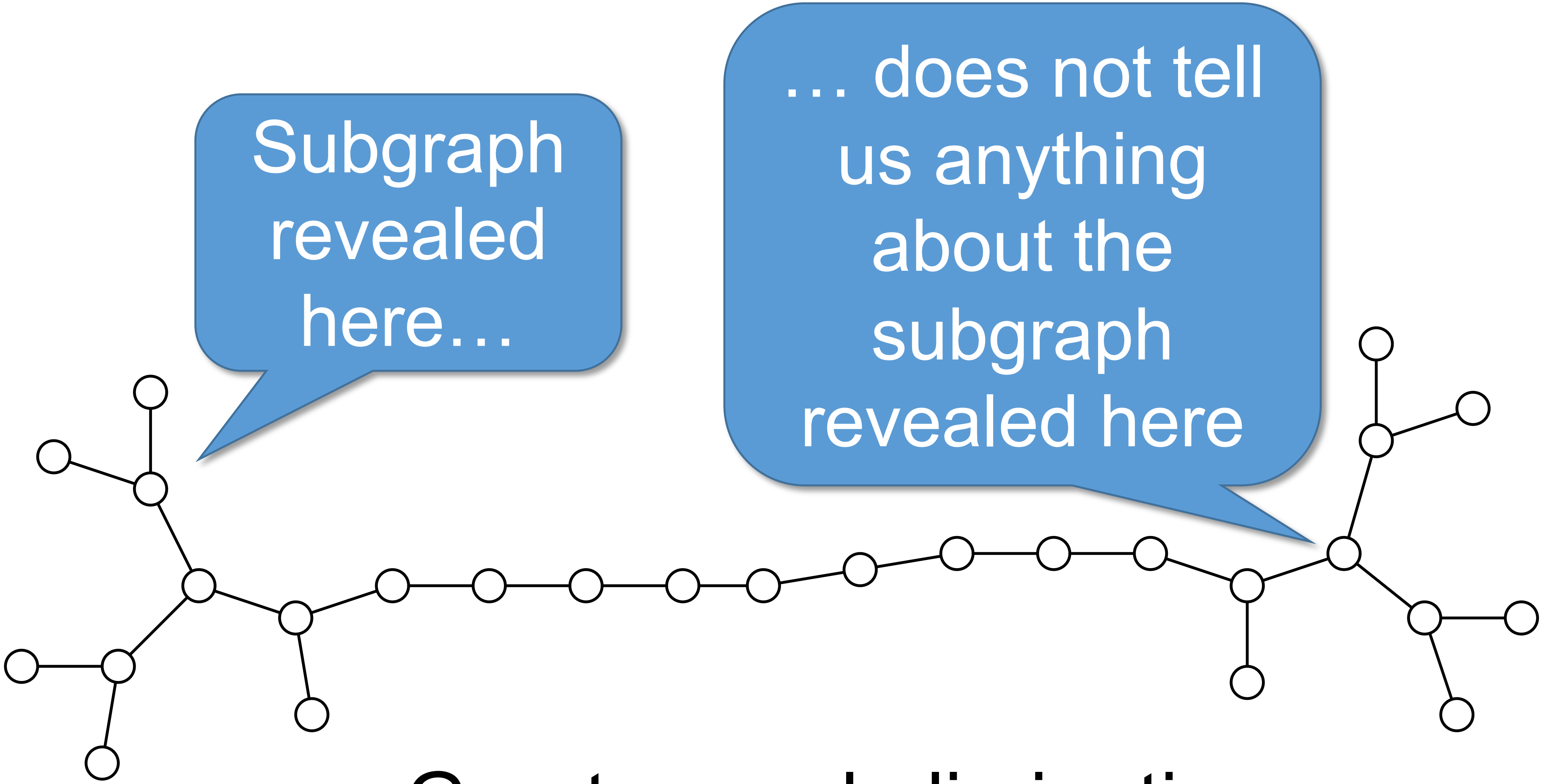
# Supported model

- Fix a 5-regular graph $G$
  - structure + identifiers globally known
- Reveal a **3-regular subgraph $H$**
  - only locally known
- Task: find a sinkless orientation in subgraph $H$




Subgraph revealed here…


Subgraph revealed here…
… does not tell us anything about the subgraph revealed here




Great, round elimination can be used!

# Summary

- **Sinkless orientation** problem
  - key problem for understanding distributed computing

# Summary

- **Sinkless orientation** problem
  - key problem for understanding distributed computing
- **Locality** known to be $\Omega(\log n)$, but hard to prove
  - cannot handle unique identifiers, go through randomness
- New much more **direct proof**
  - fix “support graph”, fix identifiers, reveal subgraph

# Summary

Also in the paper: $O(\log \log n)$ upper bound for the SLOCAL model: known result, **much simpler** algorithm

- **Sinkless orientation** problem
  - key problem for understanding distributed computing
- **Locality** known to be $\Omega(\log n)$, but hard to prove
  - cannot handle unique identifiers, go through randomness
- New much more **direct proof**
  - fix “support graph”, fix identifiers, reveal subgraph