Distributed Property Testing

Dennis Olivetti

GSSI, L'Aquila and IRIF, Paris

Overview

- Decision Problems
- Property Testing (Centralized)
  - Dense model
  - Sparse model
- Distributed Property Testing

Decision problems

- Definition:
  - Given a property P
  - Given a graph G
  - Does G satisfy the property P?

Decision problems

- Definition:
  - Given a property P
  - Given a graph G
  - Does G satisfy the property P?
- Example:
  - Given a graph G
  - Given an integer k
  - Is the graph k colorable?

Decision problems

- Definition:
  - Given a property P
  - Given a graph G
  - Does G satisfy the property P?
- Example:
  - Given a graph G
  - Given an integer k
  - Is the graph k colorable?
- Often decision problems are hard

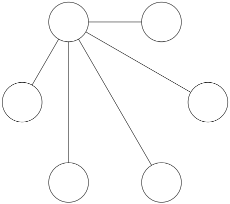
Decision problems

- Definition:
  - Given a property P
  - Given a graph G
  - Does G satisfy the property P?
- Example:
  - Given a graph G
  - Given an integer k
  - Is the graph k colorable?
- Often decision problems are hard
- Sometimes the input is huge, even linear time could be too much

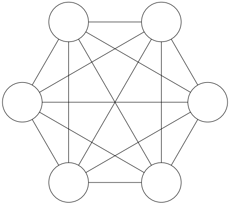
Property testing

- Relax the requirements
- Given a property P
- Given a graph G
- Distinguish whether:
  - Does G satisfy the property P?
  - Is G far from satisfying the property P?
- The input is huge:
  - Only a small part of the input can be seen
  - We want sublinear algorithms

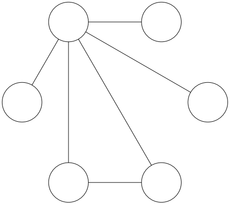
Example: 2 colorability



2 colorable



Far from being 2 colorable



Almost 2 colorable

How to measure how far is a graph from satisfying a property?

Let $G = (V, E)$, $n = |V|$, $m = |E|$. Let ϵ be a small constant in $(0, 1)$.
There exist two distinct models:

How to measure how far is a graph from satisfying a property?

Let $G = (V, E)$, $n = |V|$, $m = |E|$. Let ϵ be a small constant in $(0, 1)$. There exist two distinct models:

Dense model

A graph is ϵ-far from satisfying a property if at least ϵn^2 edges should be added or removed from G in order to make the property hold.

How to measure how far is a graph from satisfying a property?

Let $G = (V, E)$, $n = |V|$, $m = |E|$. Let ϵ be a small constant in $(0, 1)$. There exist two distinct models:

Dense model

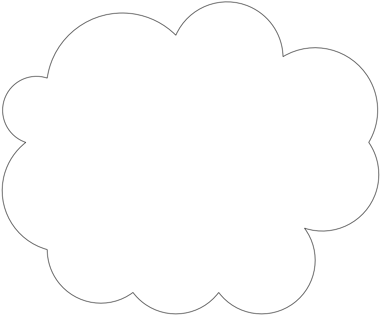
A graph is ϵ-far from satisfying a property if at least ϵn^2 edges should be added or removed from G in order to make the property hold.

Sparse model

A graph is ϵ-far from satisfying a property if at least ϵm edges should be added or removed from G in order to make the property hold.

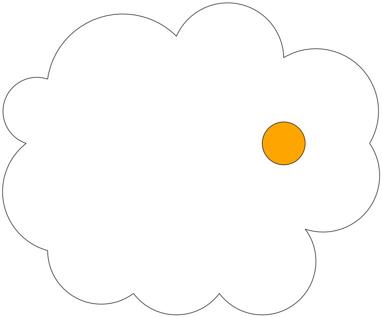
Complexity

- The complexity is measured in number of queries
- Different type of queries are allowed:



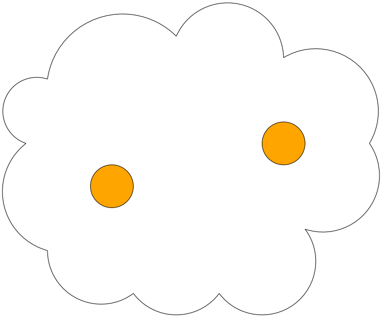
Complexity

- The complexity is measured in number of queries
- Different type of queries are allowed:
  - Give me the id of a random node



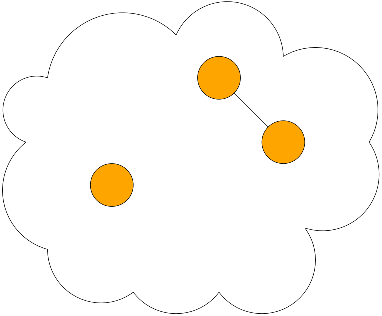
Complexity

- The complexity is measured in number of queries
- Different type of queries are allowed:
  - Give me the id of a random node



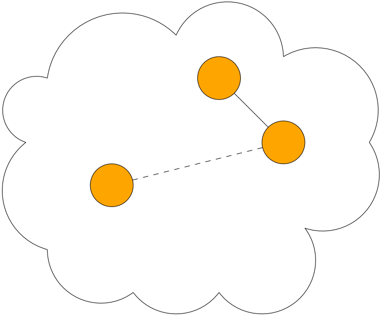
Complexity

- The complexity is measured in number of queries
- Different type of queries are allowed:
  - Give me the id of a random node
  - Give me a random neighbor of node x



Complexity

- The complexity is measured in number of queries
- Different type of queries are allowed:
  - Give me the id of a random node
  - Give me a random neighbor of node x
  - Are nodes x and y neighbors?



Definition

Property Tester (2 sided error)

A tester for a graph property P is a randomized algorithm A that is required to accept or reject any given network instance, under the following two constraints:

- G satisfies $P \Rightarrow Pr[A \text{ accepts } G] \geq \frac{2}{3}$
- G is ϵ-far from satisfying $P \Rightarrow Pr[A \text{ rejects } G] \geq \frac{2}{3}$

Definition

Property Tester (2 sided error)

A tester for a graph property P is a randomized algorithm A that is required to accept or reject any given network instance, under the following two constraints:

- G satisfies $P \Rightarrow Pr[A \text{ accepts } G] \geq \frac{2}{3}$
- G is ϵ-far from satisfying $P \Rightarrow Pr[A \text{ rejects } G] \geq \frac{2}{3}$

Property Tester (1 sided error)

A tester for a graph property P is a randomized algorithm A that is required to accept or reject any given network instance, under the following two constraints:

- G satisfies $P \Rightarrow A$ accepts G
- G is ϵ-far from satisfying $P \Rightarrow Pr[A \text{ rejects } G] \geq \frac{2}{3}$

Subgraph freeness

We want to know if G does not contain any copy of a subgraph H, or if it contains many copies of H, being H some small graph (e.g. K_5).

Subgraph freeness

We want to know if G does not contain any copy of a subgraph H, or if it contains many copies of H, being H some small graph (e.g. K_5).

- Easy in the dense model

Subgraph freeness

We want to know if G does not contain any copy of a subgraph H, or if it contains many copies of H, being H some small graph (e.g. K_5).

- Easy in the dense model

Graph removal lemma

For every k-node graph H, and every $\epsilon > 0$, there exists $\delta > 0$ such that every n-node graph containing at most δn^k copies of H can be transformed into an H-free graph by deleting at most ϵn^2 edges.

Subgraph freeness

We want to know if G does not contain any copy of a subgraph H, or if it contains many copies of H, being H some small graph (e.g. K_5).

- Easy in the dense model

Graph removal lemma

For every k-node graph H, and every $\epsilon > 0$, there exists $\delta > 0$ such that every n-node graph containing at most δn^k copies of H can be transformed into an H-free graph by deleting at most ϵn^2 edges.

- If a graph is far from being H free, it contains $\Omega(n^k)$ copies of H!
- Choose k nodes u.a.r., the probability to detect a copy of H is constant

Subgraph freeness

We want to know if G does not contain any copy of a subgraph H, or if it contains many copies of H, being H some small graph (e.g. K_5).

- Easy in the dense model

Graph removal lemma

For every k-node graph H, and every $\epsilon > 0$, there exists $\delta > 0$ such that every n-node graph containing at most δn^k copies of H can be transformed into an H-free graph by deleting at most ϵn^2 edges.

- If a graph is far from being H free, it contains $\Omega(n^k)$ copies of H!
- Choose k nodes u.a.r., the probability to detect a copy of H is constant

Lemma

H freeness can be tested in constant time, for any H of constant size.

A weaker lemma that holds in the sparse model

Lemma [Fraigniaud, Rapaport, Salo, Todinca '16]

Let H be any graph. Let G be an m-edge graph that is ϵ-far from being H-free. Then G contains at least $\epsilon m/|E(H)|$ edge-disjoint copies of H.

A weaker lemma that holds in the sparse model

Lemma [Fraigniaud, Rapaport, Salo, Todinca '16]

Let H be any graph. Let G be an m-edge graph that is ϵ-far from being H-free. Then G contains at least $\epsilon m/|E(H)|$ edge-disjoint copies of H.

The number of copies of H is proportional to m instead of $n^{|V(H)|}$, in the sparse model the problem is harder, in fact:

Lemma [Alon, Kaufman, Krivelevich, Ron '08]

Testing triangle freeness requires $\Omega(n^{\frac{1}{3}})$ queries.

Distributed property testing

Definition

A distributed tester for a graph property P is a distributed randomized algorithm A that satisfies the following conditions:

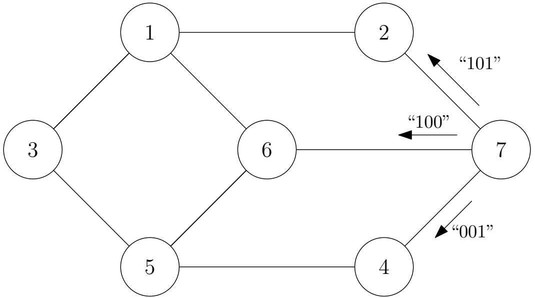
- G satisfies $P \Rightarrow$ every node outputs "accept"
- G is ϵ-far from satisfying $P \Rightarrow$ Pr[at least one node outputs "reject"] $\geq \frac{2}{3}$

The Congest Model

- All nodes start the computation at the same round
- The computation proceeds in phases
- At each phase each node:

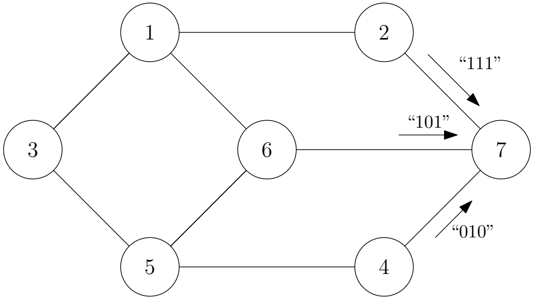
The Congest Model

- All nodes start the computation at the same round
- The computation proceeds in phases
- At each phase each node:
  - sends (possibly different) messages to its neighbors



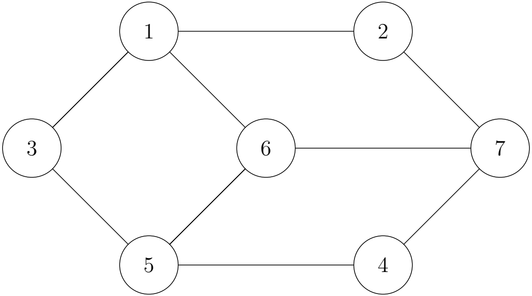
The Congest Model

- All nodes start the computation at the same round
- The computation proceeds in phases
- At each phase each node:
  - sends (possibly different) messages to its neighbors
  - receives messages sent by its neighbors



The Congest Model

- All nodes start the computation at the same round
- The computation proceeds in phases
- At each phase each node:
  - sends (possibly different) messages to its neighbors
  - receives messages sent by its neighbors
  - performs some local computation



The Congest Model

- The main constraint of the Congest model is that the exchanged messages should be small, typically $O(\log n)$.
- For example, messages of size $O(\log n)$ are enough to transmit, in a single round, a constant number of IDs of neighbors.

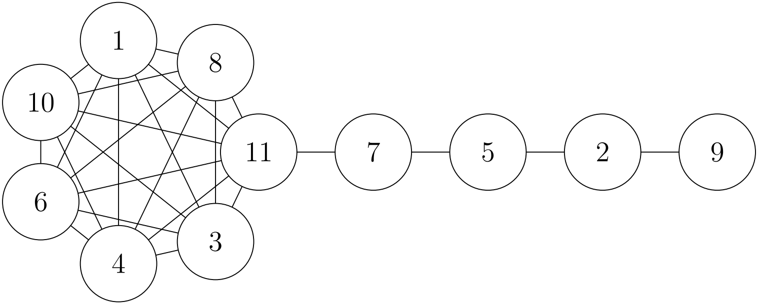
Decision problems in the Congest model

It is difficult to decide distributedly if a graph satisfies a property in constant time because of:

- Locality: two nodes can communicate in a time proportional to their distance
- Congestion: a node can not communicate all its neighbors to a single neighbor because they could be many

Example: knowing if a ring is 2 colorable requires to know the parity of its size.

Knowing the 2-hop neighborhood is hard



Dense model

Lemma [Censor-Hillel, Fischer, Schwartzman, Vasudev '16]

Any ϵ-tester for the dense model (for a non-disjointed property) that makes q queries can be converted to a distributed ϵ-tester that requires $O(q^2)$ rounds in the distributed setting.

Dense model

Lemma [Censor-Hillel, Fischer, Schwartzman, Vasudev '16]

Any ϵ-tester for the dense model (for a non-disjointed property) that makes q queries can be converted to a distributed ϵ-tester that requires $O(q^2)$ rounds in the distributed setting.

Example of properties testable in constant time:

- Is G H-free?
- Is G k-colorable?
- Is G a perfect graph?

Sparse model

[Censor-Hillel, Fischer, Schwartzman, Vasudev '16]

- Triangle freeness can be tested in $O(1/\epsilon^2)$
- Cycle freeness can be tested $O(\log n/\epsilon)$
- Cycle freeness requires at least $\Omega(\log n)$
- Bipartiteness can be tested in in $O(poly(\log \frac{n}{\epsilon}/\epsilon))$ in bounded degree graphs

[Fraigniaud, Rapaport, Salo, Todinca '16]

- H-freeness can be tested in constant time for any H s.t. $|V(H)| \leq 4$

Example: triangle freeness [Censor-Hillel, Fischer, Schwartzman, Vasudev '16]

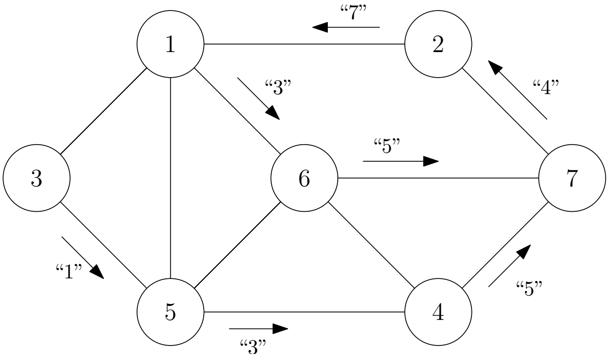
Each node repeats $32\epsilon^{-2}$ times the following procedure:

- select two neighbors u and v uniformly at random
- ask to u if v is his neighbor

Example: triangle freeness [Censor-Hillel, Fischer, Schwartzman, Vasudev '16]

Each node repeats $32\epsilon^{-2}$ times the following procedure:

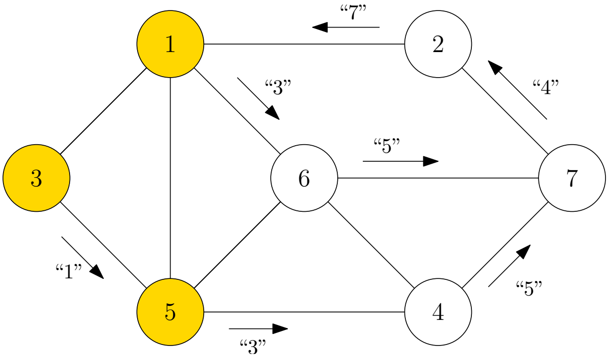
- select two neighbors u and v uniformly at random
- ask to u if v is his neighbor



Example: triangle freeness [Censor-Hillel, Fischer, Schwartzman, Vasudev '16]

Each node repeats $32\epsilon^{-2}$ times the following procedure:

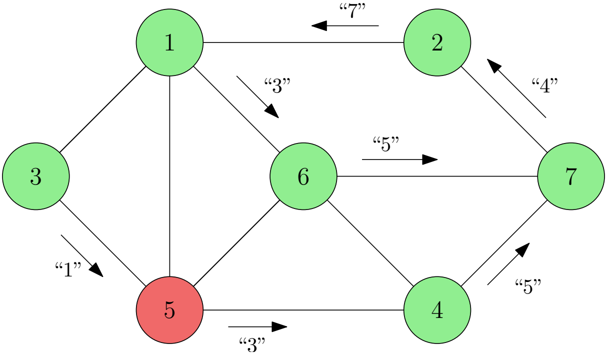
- select two neighbors u and v uniformly at random
- ask to u if v is his neighbor



Example: triangle freeness [Censor-Hillel, Fischer, Schwartzman, Vasudev '16]

Each node repeats $32\epsilon^{-2}$ times the following procedure:

- select two neighbors *u* and *v* uniformly at random
- ask to *u* if *v* is his neighbor



Generalization of the procedure: DFS and BFS testers

Each node:

- chooses some neighbor at random
- sends it to some random neighbor
- samples and propagates the received information

[Fraigniaud, Rapaport, Salo, Todinca '16]

BFS and DFS testers can not detect C_k and K_k for $k \geq 5$.

C_k detection

[Fraigniaud, O. '17]

There exists an ϵ-tester for C_k freeness, for any constant $k \geq 3$, that requires $O(\frac{1}{\epsilon})$ rounds in the CONGEST model.

C_k detection

[Fraigniaud, O. '17]

There exists an ϵ-tester for C_k freeness, for any constant $k \geq 3$, that requires $O(\frac{1}{\epsilon})$ rounds in the CONGEST model.

Procedure:

- Choose an edge u.a.r.
- Check if there is a cycle of length k passing through that edge
  - It can be done deterministically

Choose an edge at random

Lemma [Fraigniaud, Rapaport, Salo, Todinca '16]

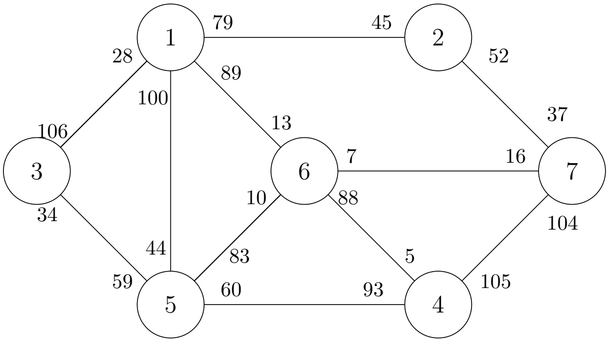
Let H be any graph. Let G be an m-edge graph that is ϵ-far from being H-free. Then G contains at least $\epsilon m/|E(H)|$ edge-disjoint copies of H.

This implies that by choosing a random edge we have probability $\Omega(\epsilon)$ to choose an edge that is part of some copy of H.

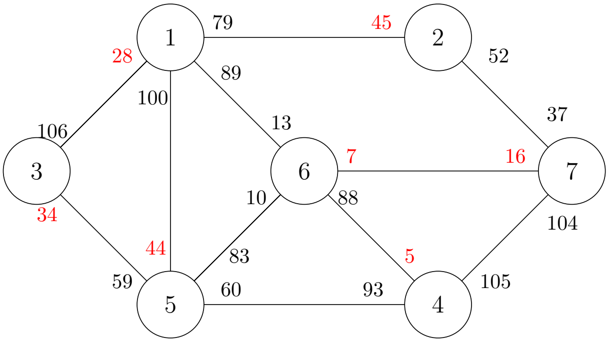
Choose an edge at random

- Each node picks a random weight w from $[1, m^2]$ for each edge incident to him
- The "leader" of each edge is the endpoint that chose the smaller weight
- If a node is the leader of multiple edges, choose the one with smaller weight
- Broadcast the edge of known minimum weight, and its weight, for a constant number of rounds

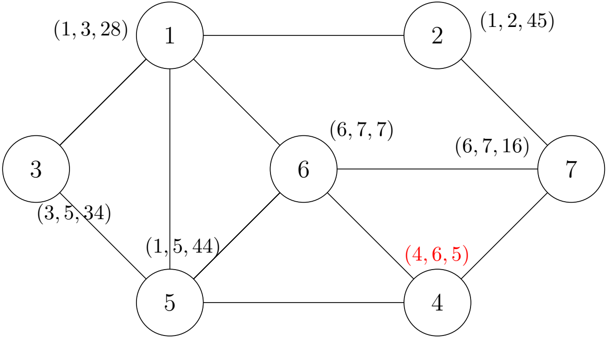
Choose an edge at random



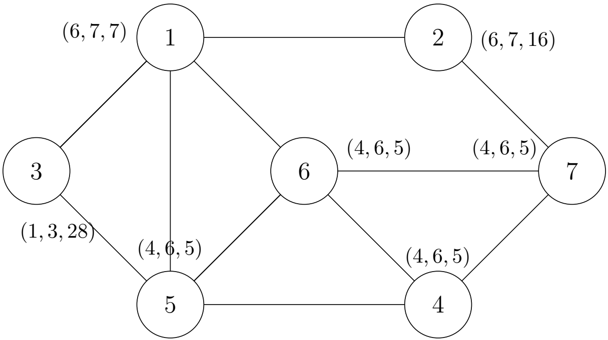
Choose an edge at random



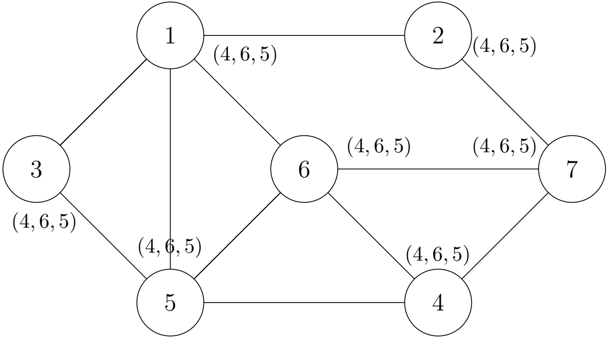
Choose an edge at random



Choose an edge at random



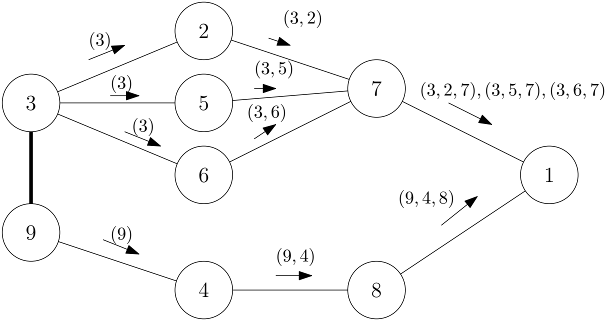
Choose an edge at random



Check the presence of a cycle

Naïve solution:

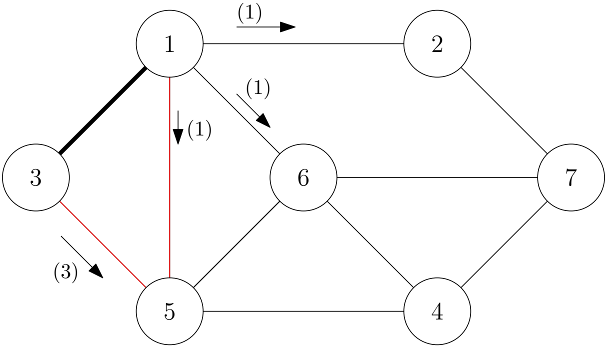
- The endpoints of the chosen edge broadcast their id
- Repeat
  - Append my id to each received sequence
  - Broadcast the new sequences just created
  - Check if two sequences are disjoint and form a cycle of desired length



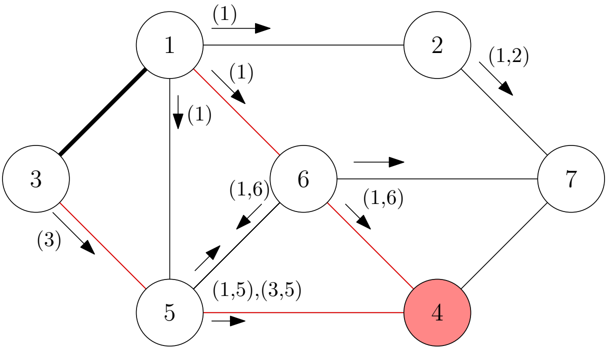
Example: triangle freeness

Repeat $O(\frac{1}{\epsilon})$ times:

- Choose an edge (u, v) as described before
- u and v broadcast
- If a node receives two messages a triangle is detected



C_5 detection

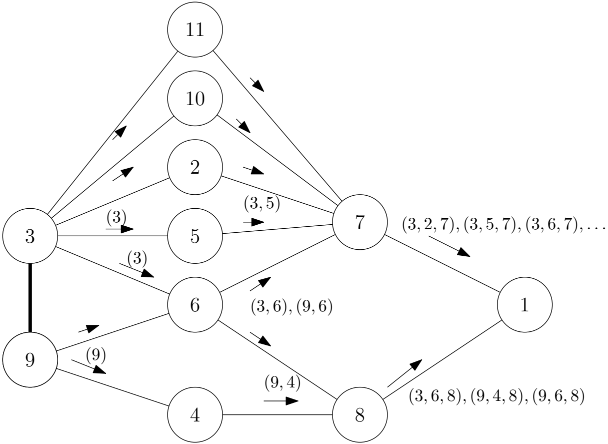


Node 4:

- receives $(1,5),(3,5),(1,6)$
- detects $(1,6,4,5,3)$

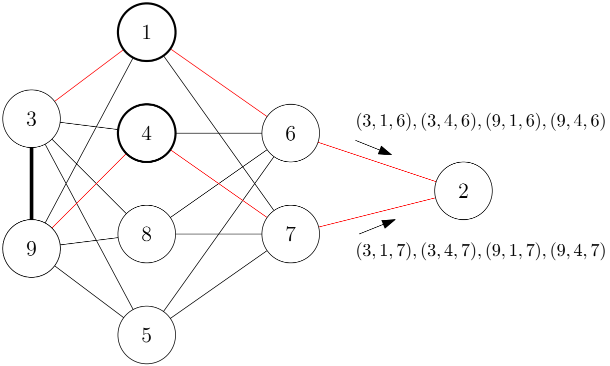
C_7 detection

- Nodes at distance 2 could potentially receive $\Theta(n)$ messages
- The previous procedure could require a lot of bandwidth



C_7 detection

- The partial solution can be sparsified
- For C_7, 3 subpaths (for each inital node) are enough



Sparsification of the intermediate solution

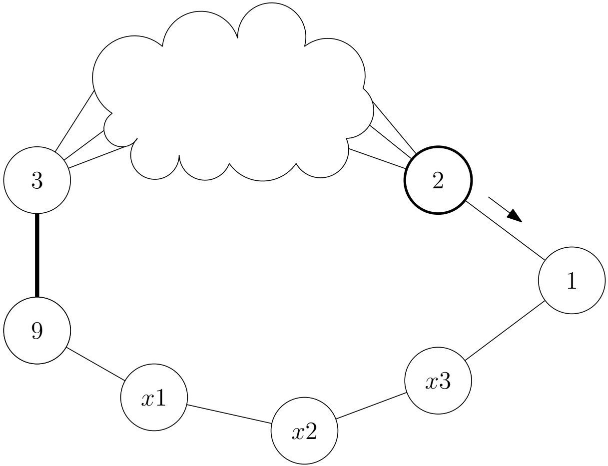
Lemma [Erdős, Hajnal, Moon '64]

Let V be a set of size n, and consider two integer parameters p and q. For any set $F \subseteq \mathcal{P}(V)$ of subsets of size at most p of V, there exists a *compact* (p, q)*-representation* of F, i.e., a subset $\hat{F}$ of F satisfying:

1. For each set $C \subseteq V$ of size at most q, if there is a set $L \in F$ such that $L \cap C = \emptyset$, then there also exists $\hat{L} \in \hat{F}$ such that $\hat{L} \cap C = \emptyset$;
2. The cardinality of $\hat{F}$ is at most $\binom{p+q}{p}$, for any $n \geq p + q$.

In other words, the number of subpaths that must be forwarded at each round do not depend on the size of the graph.

Sparsification of the intermediate solution

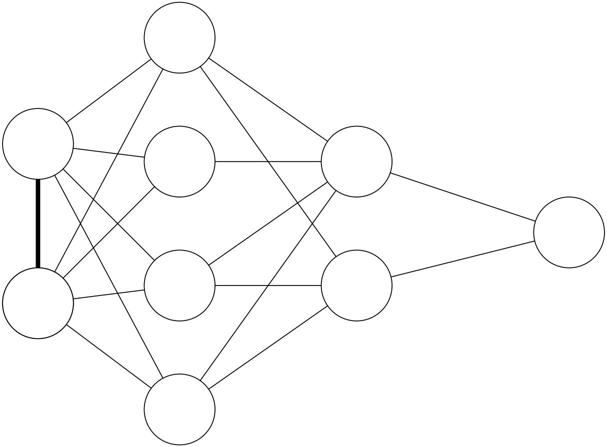


- Node 2 should send at least one sequence that does not contain $x1, x2$ and $x3$
- A constant number of sequences are enough

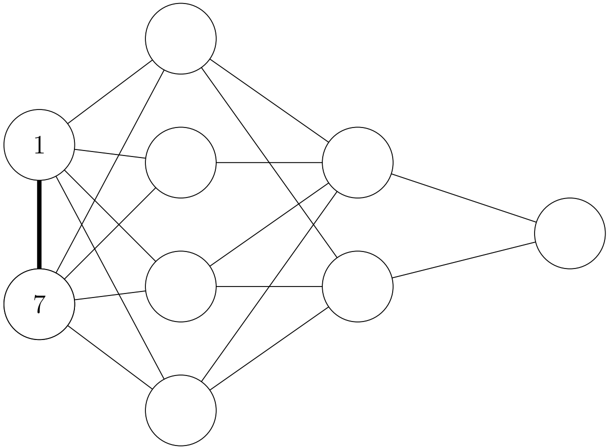
An easier (randomized) solution

- Pick a random edge (u, v)
- Each node picks a random color from $[1, k]$
- with constant probability the nodes of a cycle going from u to v will have colors $1, 2, \ldots, k$
- Start a BFS from u, that at round i can pass only on nodes with color i
- If the BFS reaches v at round k, a cycle is detected

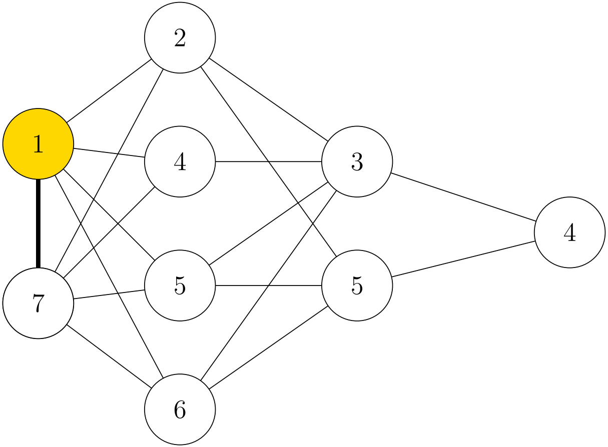
An easier (randomized) solution



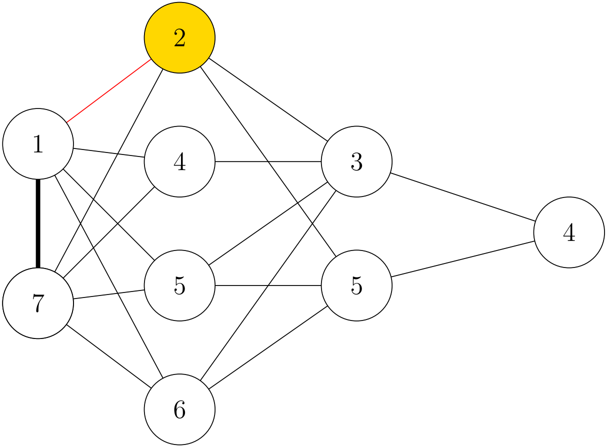
An easier (randomized) solution



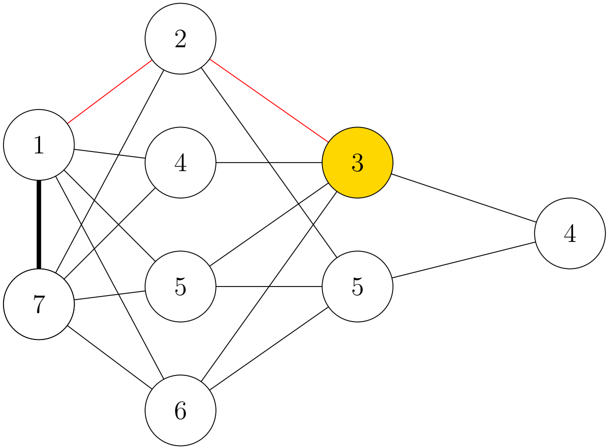
An easier (randomized) solution



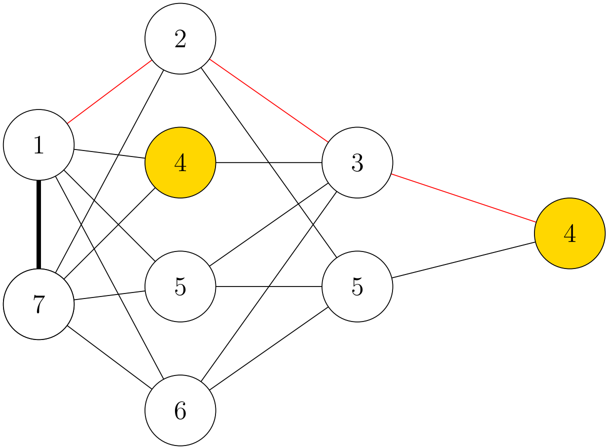
An easier (randomized) solution



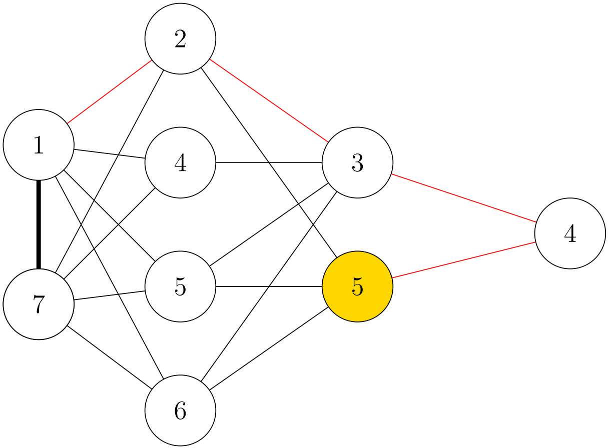
An easier (randomized) solution



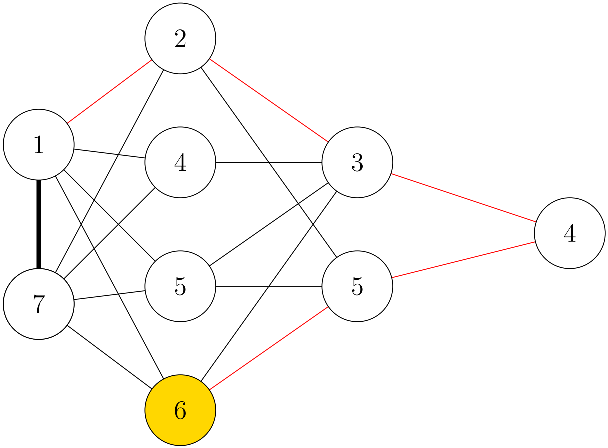
An easier (randomized) solution



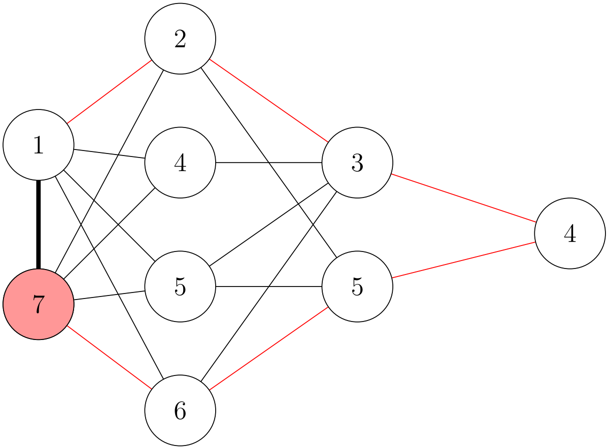
An easier (randomized) solution



An easier (randomized) solution



An easier (randomized) solution

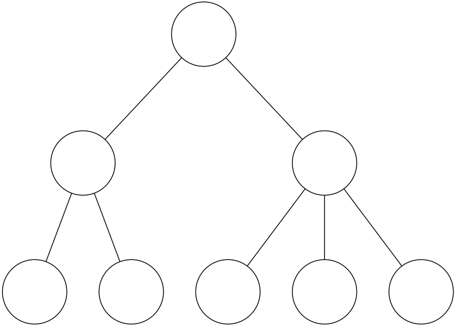


Tree detection

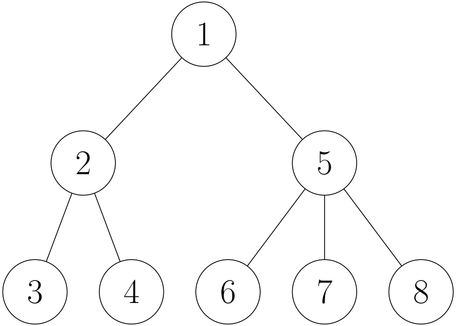
[Fraigniaud, Montealegre, O., Rapaport, Todinca '17]

In the CONGEST model, it is possible to check the presence of a fixed tree T of constant size, in $O(1)$ rounds, deterministically.

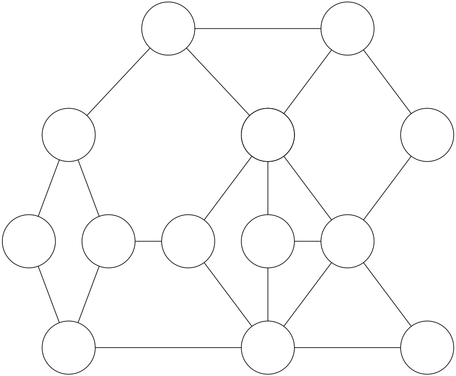
Tree detection: example

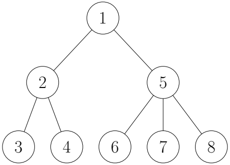


Tree detection: example

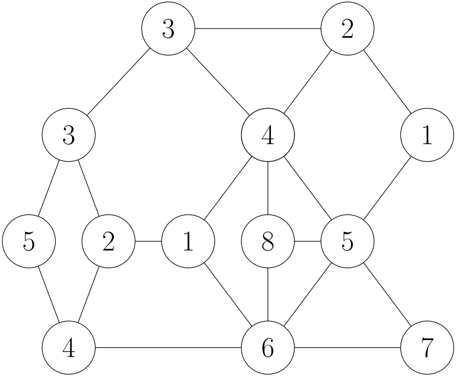


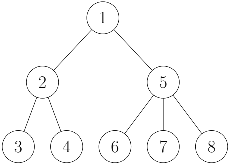
Tree detection: example



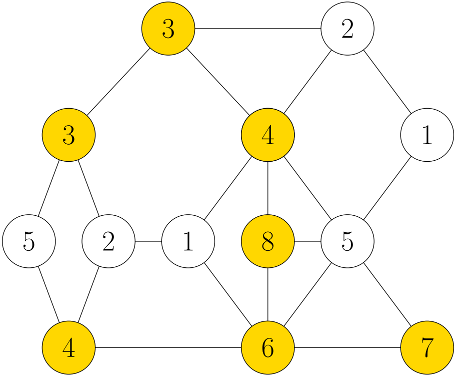


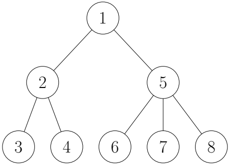
Tree detection: example



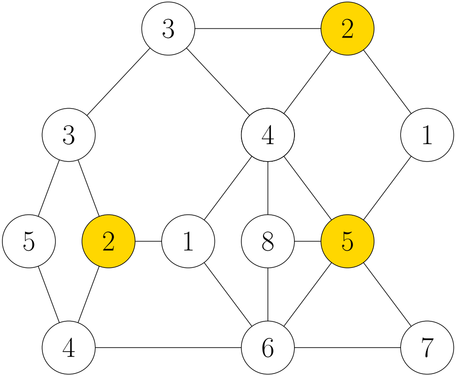


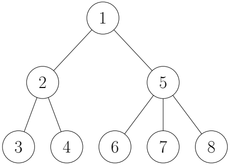
Tree detection: example



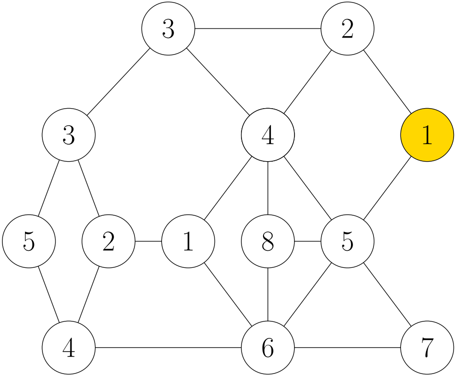


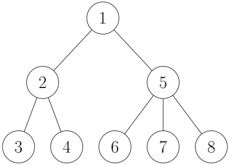
Tree detection: example



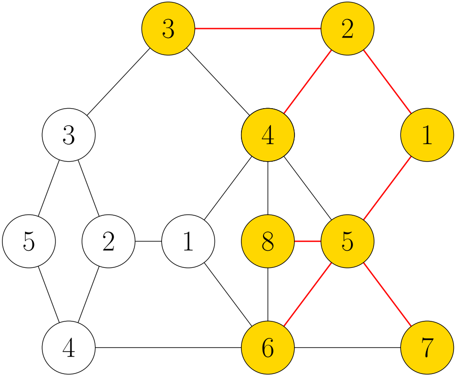


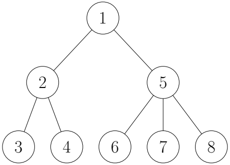
Tree detection: example



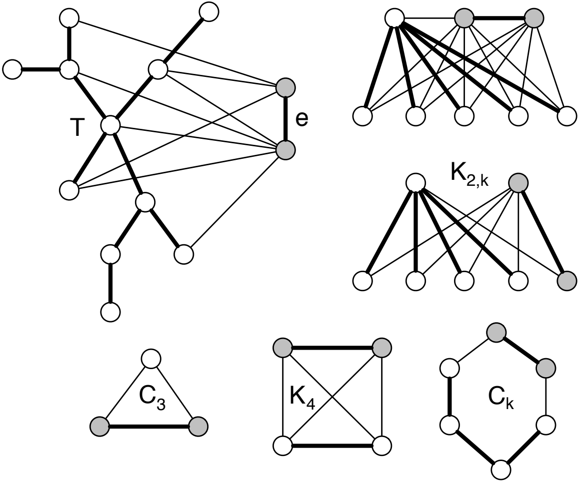


Tree detection: example





Tree + 1 edge

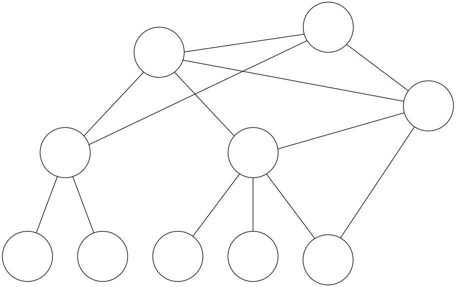


Tree + 1 edge

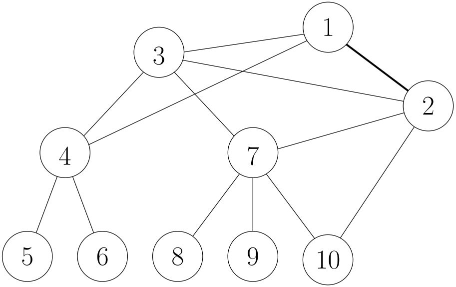
[Fraigniaud, Montealegre, O., Rapaport, Todinca '17]

There exists an ϵ-tester for H freeness, for any graph H of constant size composed by a tree, an edge, and arbitrary connections between the endpoints of the edge and the nodes of the tree, that requires $O(\frac{1}{\epsilon})$ rounds in the CONGEST model.

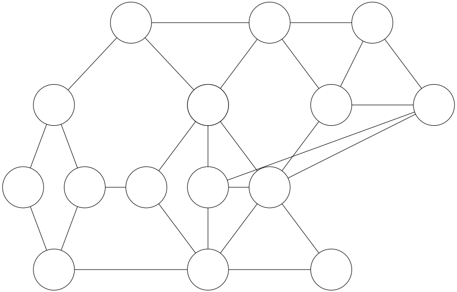
Tree + 1 edge: example

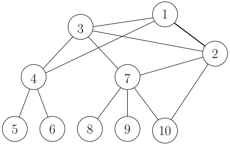


Tree + 1 edge: example

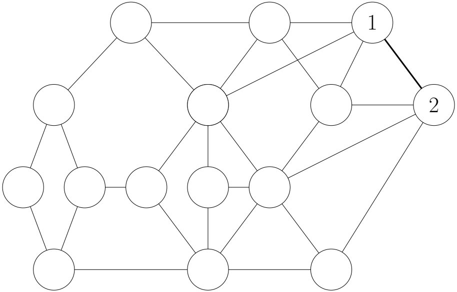


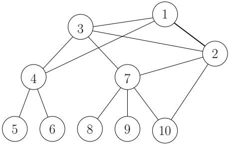
Tree + 1 edge: example



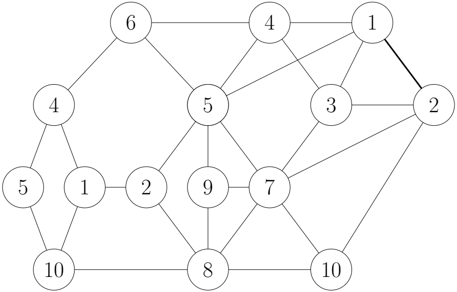


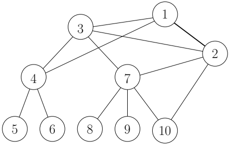
Tree + 1 edge: example



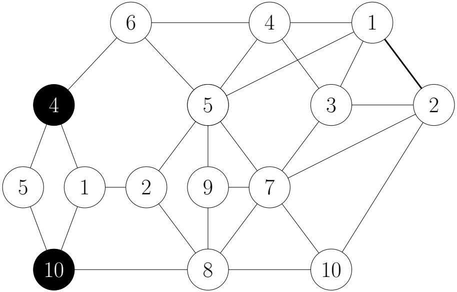


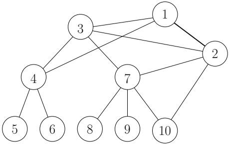
Tree + 1 edge: example



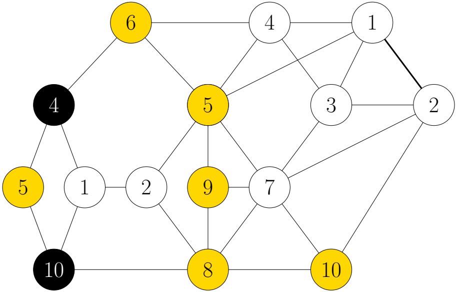


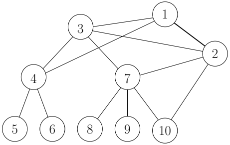
Tree + 1 edge: example



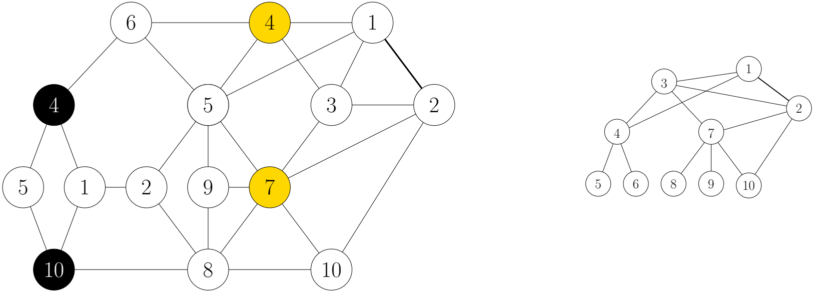


Tree + 1 edge: example

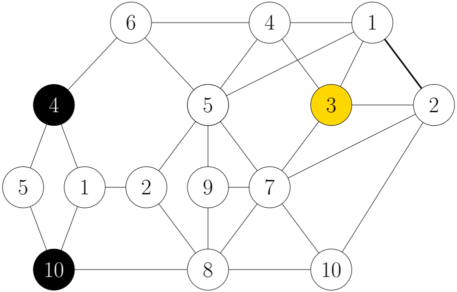


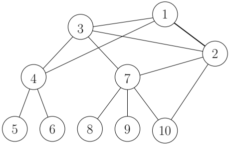


Tree + 1 edge: example

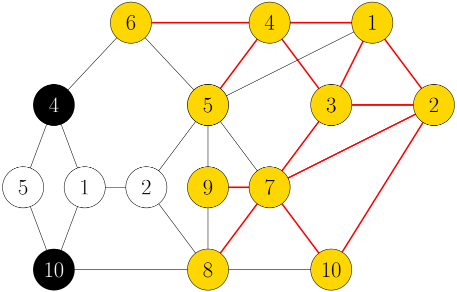


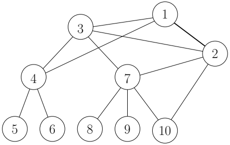
Tree + 1 edge: example



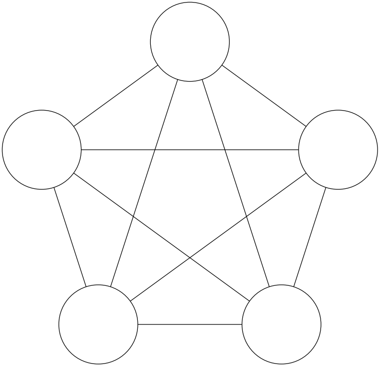


Tree + 1 edge: example

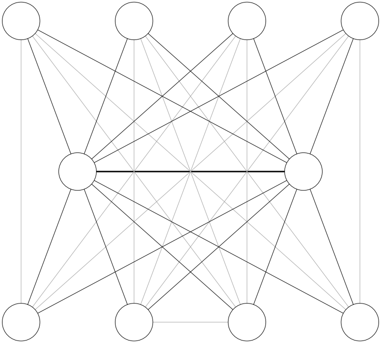




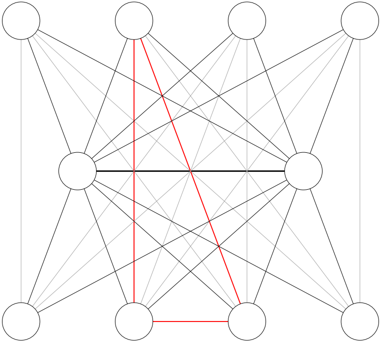
Open problems



Open problems



Open problems



Does there exist an ϵ-tester for K_5-freeness?

Conclusions

- There exists a deterministic algorithm for the CONGEST model that can check the presence of a fixed tree in a constant number of rounds
- There exists an ϵ-tester for the CONGEST model that can check the presence of a fixed tree + 1 edge in $O(1/\epsilon)$
- The minimal graph not testable with the above algorithms is K_5
- For distributed property testing, no lower bounds are known!

Thank you