# Three Notes on Distributed Property Testing

Guy Even[1,*], Orr Fischer[2], Pierre Fraigniaud[3], **Tzlil Gonen[2]**, Reut Levi[4,*], **Moti Medina[5,*]**, Pedro Montealegre[6], **Dennis Olivetti[7]**, Rotem Oshman[2], Ivan Rapaport[8], & Ioan Todinca[9]

1. Tel-Aviv University, EE, Israel
2. Tel Aviv University, CS, Israel
3. CNRS and University Paris Diderot, France
4. Weizmann Institute of Science, CS and Applied Math, Israel
5. Ben-Gurion University of the Negev, ECE, Israel
6. Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Chile
7. Gran Sasso Science Institute, L'Aquila, Italy
8. DIM-CMM (UMI 2807 CNRS), Universidad de Chile, Chile
9. Université d'Orléans, INSA Centre Val de Loire, LIFO EA 4022, France

*. Work done in Max Planck Institute for Informatics, Saarland Informatics Campus, Germany

# The Distributed **CONGEST** Model [Peleg 2000]

- A synched network $G = (V, E)$
- $V$ are the processors
  - Each processor has a distinct ID.
- $E$ are the communication links.
- Each processor is given a local input.
- In each round, each processor performs the following steps:
  1. **Receive** messages from neighbors.
  2. **Execute** a local (randomized) computation.
  3. **Sends** messages of $O(\log n)$ **bits** to every neighbor.
- Last round: all processors stop and output a local output.
- Complexity measure: $\#rounds$.

# Distributed Property Testing in the **CONGEST** Model (General Model ver.)

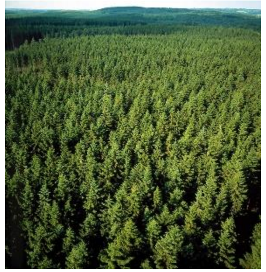
[Censor-Hillel, Fischer, Schwartzman, Vasudev. 2016]

- A graph $G = (V, E)$.
- Edge-distance: $dist(G, G') \triangleq |E \Delta E'|$
  - The edge-distance between two graphs = #edges in the symmetric diff.
- A graph property $P$.
  - Set of all graphs that have the property $P$.
- Distance from $P$: $dist(G, P) \triangleq \min_{G' \in P} dist(G, G')$.
- $G$ is $\epsilon$-far from $P$ : $dist(G, P) \geq \epsilon \cdot |E|$.

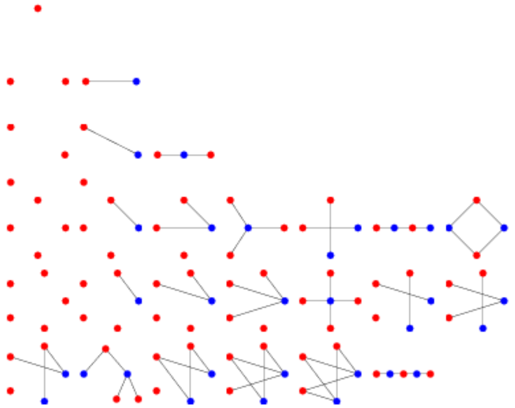

- $\epsilon$-tester for $P$ : $\begin{cases} G \in P, & \forall v \in V \ output\ \boldsymbol{ACCEPT} \\ G \text{ is } \epsilon\text{–far from } P, & \exists v \in V\ output\ \boldsymbol{REJECT}\ w.p.\ 2/3 \end{cases}$

# Studied Problems


- **The Subgraph-Freeness Problem.**
  - Given a graph $H$, s.t. $H = O(1)$.
  - $P = \{All\ the\ graphs\ that\ does\ not\ contain\ H\ as\ a\ subgraph\}.$
  - Examples: $T$-freeness, $K_s$-freeness, $C_s$-freeness.



- **Cycle-freeness**
  - $P = \{All\ the\ graphs\ that\ are\ acyclic\}.$
- **Bipartiteness**
  - $P = \{All\ the\ graphs\ that\ are\ Bipartite\}.$



# Overview of Previous Results

- Initiated by Brakerski, Patt-Shamir 2011.
  - Testing algorithm for finding large *near-cliques* in the graph.
- Censor-Hillel, Fischer, Schwartzman, and Vasudev, DISC 2016.
  - Property testing in **CONGEST**
  - Triangle-freeness, cycle-freeness, bipartiteness.
  - Lower bounds $\Omega(\log|V|)$ for Bipartiteness, and Cycle-freeness.
- Fraigniaud, Rapaport, Salo, and Todinca, DISC 2016.
  - Tester for $H$-freeness, $|V(H)| \leq 4$
  - For $|V(H)| > 4$ presented a “hard” family for algs with “natural” properties.
- Pierre Fraigniaud and Dennis Olivetti, SPAA 2017.
  - Tester for $C_s$-freeness, $s \geq 4$.

# Overview of Main Results

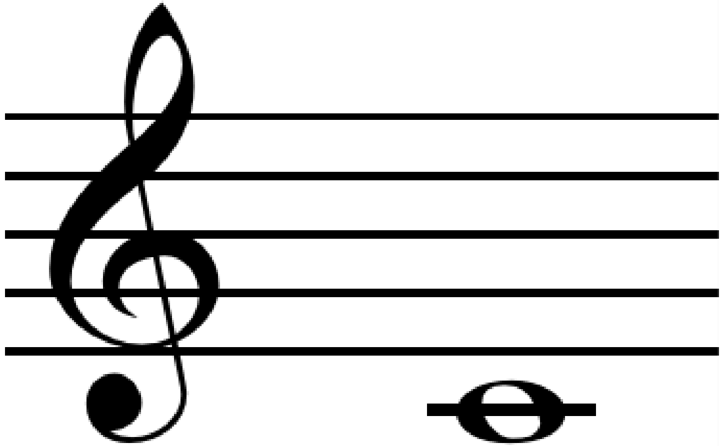
- $H$-freeness:
  - $O(1/\epsilon)$ #rounds,
  - For a large family of graphs $H$, where $|H| = O(1)$.
- $T$-freeness:
  - A deterministic **CONGEST** alg.
  - Decision alg.
  - Constant #rounds.
- $K_s$-freeness:
  - $s \geq 3$,
  - $O\left(|E|^{\frac{1}{2}-\frac{1}{s-2}} \cdot \epsilon^{-\frac{1}{2}-\frac{1}{s-2}}\right)$ #rounds.
- Reducing the dependency on the diameter
  - **Bipartiteness**: $O((\log|V|)/\epsilon)$ #rounds.
  - Testing and **correcting Cycle-freeness**: $O((\log|V|)/\epsilon)$ #rounds.

# First Note

Guy Even

Reut Levi

Moti Medina



- Introducing **Distributed Correction**
- Reducing the Dependency on the Diameter and Applications
  - Testing **Bipartiteness**,
  - Testing **Cycle-freenes**,
  - **Corrector** for Cycle-freeness.

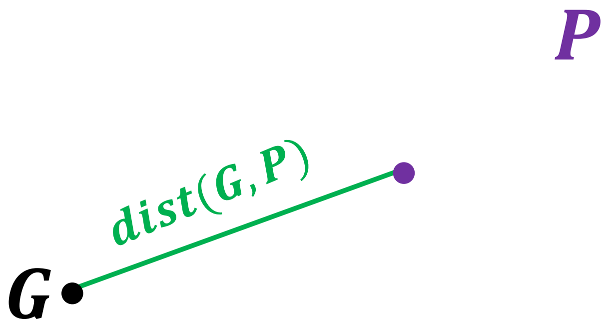

- Testers for $H$-freeness for $|V(H)| \leq 4$.
  - $O(\epsilon^{-1})$ rounds.
- $T$-freeness
  - Centralized testing for any tree $T$.
  - Distributed simulation: $\epsilon$-tester with $O(k^{k^2+1} \cdot \epsilon^{-k})$ rounds.

# Distributed Correctors: Motivation

- $\epsilon$-tester
  - $G$ is $\epsilon$-far from $P \rightarrow \exists v \in V$ that outputs **REJECT** w.p $\geq 2/3$.
  - That is, $dist(G, P) \geq \epsilon \cdot |E|$.
- $\Rightarrow$ 1 vertex shouts **"NO"** even though there are $\geq \epsilon \cdot |E|$ "violations" .
  - **Lots of edges** to add or remove!
- We prefer:
  - Having that $\epsilon$ fraction of $|V|$ output **REJECT**.
  - Having that $dist(G, P)$ vertices output **REJECT**.
- Or even better, that $G$ locally **"correct"** itself!

# Distributed Corrector

A graph property $\boldsymbol{P}$ is ***edge-monotone* (EM)** if $\boldsymbol{G} \in \boldsymbol{P}$ and $\boldsymbol{G}'$ is obtained from $\boldsymbol{G}$ by the removal of edges, then $\boldsymbol{G}' \in \boldsymbol{P}$.

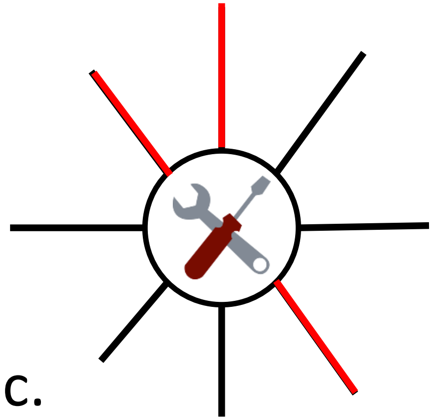


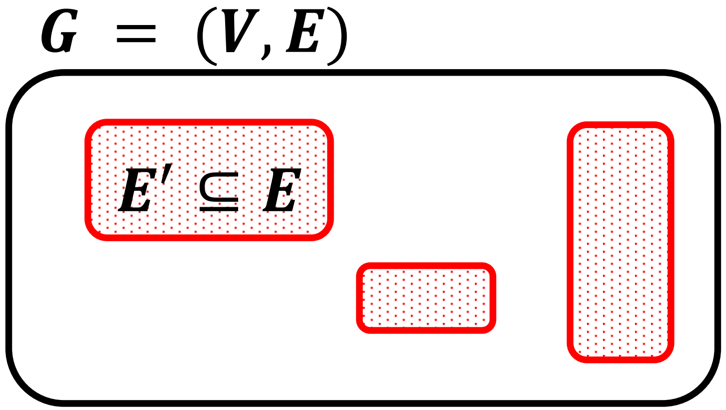

$\boldsymbol{dist(G,P)}$ min #edges that should be removed from $\boldsymbol{G}$ in order to obtain the property $\boldsymbol{P}$.

An algorithm is **$\epsilon$-corrector** for property $\boldsymbol{P}$ if:

- $E' \subseteq E$,
- $G(V, E \setminus E') \in P$,
- $|E'| \leq dist(G,P) + \epsilon \cdot |E|$,
- Upon termination $\forall v \in V : knows\ E'(v)$.

**Example:** Cycle-freeness corrector: $E \setminus E'$ is acyclic.





# Prelim. I: $(\boldsymbol{\beta}, \boldsymbol{d})$-decomposition

[Miller, Peng, Chen Xu 2013]

$V_1$

$\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$

Partition of $V$ into disjoint subsets $\boldsymbol{V_1}, \dots, \boldsymbol{V_k}$:

- $\forall 1 \le i \le k$: $\boldsymbol{G[V_i]}$ is connected.
  - $G[V_i]$: vertex induced subgraph of $G$, induced by $V_i$.
- $\forall 1 \le i \le k$: $\boldsymbol{diam(G[V_i]) \le d}$,
- $\boldsymbol{\#cut\ edges \le \beta \cdot |E|}$.

$\boldsymbol{V}_1$

$\boldsymbol{V}_2$

$\boldsymbol{V}_3$

# Prelim. II: Alg $(\epsilon, (\log n)/\epsilon)$-decomposition in **CONGEST**

[Elkin & Neiman 2017]

**Thm.** An $(\epsilon, O(\log n/\epsilon))$-decomposition can be computed

- Randomized **CONGEST**-model,
- $O((\log n)/\epsilon)$ rounds,
- w.p. $\geq 1 - 1/\mathrm{Poly}(n)$.

$V_i$

At the end of the algorithm:

- There is a spanning rooted tree $T_i$ for each subset $V_i$.
- Each $v \in V_i$ knows: the root of $T_i$, its parent in $T_i$.
- Each $v \in V_i$ knows which of the edges incident to it are cut-edges.

Algorithms for $(\epsilon, (\log n)/\epsilon)$-decompositions were developed in the context of parallel algorithms:

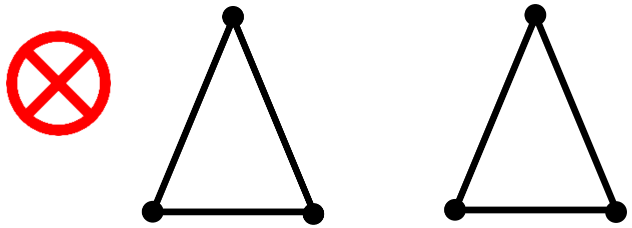

- Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Low-diameter graph decomposition is in NC. In Scandinavian Workshop on Algorithm Theory, pages 83–93.Springer, 1992.
- Guy E Blelloch, Anupam Gupta, Ioannis Koutis, Gary L Miller, Richard Peng, and Kanat Tangwongsan. Nearly-linear work parallel sdd solvers, low-diameter decomposition, and low-stretch subgraphs. Theory of Computing Systems, 55(3):521–554, 2014.
- Gary L Miller, Richard Peng, and Shen Chen Xu. Parallel graph decompositions using random shifts. In Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures, pages 196–203. ACM, 2013.

A graph property $\boldsymbol{P}$ is ***edge-monotone*** **(EM)** if $\boldsymbol{G} \in \boldsymbol{P}$ and $\boldsymbol{G}'$ is obtained from $\boldsymbol{G}$ by the removal of edges, then $\boldsymbol{G}' \in \boldsymbol{P}$.

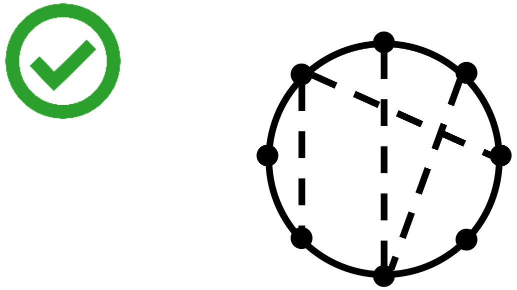
# Our Result:
# Reducing #rounds $O(Diam) \rightarrow O(\epsilon^{-1}\log n)$

- A graph property $\boldsymbol{P}$ is **non-disjointed (ND)** if for every witness $\boldsymbol{G}'$ against $\boldsymbol{G} \in \boldsymbol{P}$, there exists an induced subgraph $\boldsymbol{G}''$ of $\boldsymbol{G}'$ that is **connected** such that $\boldsymbol{G}''$ is also a witness against $\boldsymbol{G} \in \boldsymbol{P}$.
- **Verifier** for $\boldsymbol{P}$: a distributed algorithm in which **all** vertices **accept** iff $\boldsymbol{G} \in \boldsymbol{P}$.

$\boldsymbol{P_1 = \{graphs\ with\ at\ most\ one\ triangle\}}$



$\boldsymbol{P_1 = \{acyclic\ graphs\}}$



**Thm.** Let $\boldsymbol{P}$ be an **edge-monotone non-disjointed** graph property, let $\boldsymbol{G}$ be the input graph.

- **Verifier** in $\boldsymbol{O(Diam(G))}$ rounds
- $\Rightarrow \exists$ **$\epsilon$-tester** in $\boldsymbol{O((\log n)/\epsilon)}$ rounds w.p. $\boldsymbol{\geq 1 - 1/\mathrm{Poly}(n)}$.

$\boldsymbol{\epsilon - tester}$ for $\boldsymbol{P}$
$\boldsymbol{\#rounds = O((\log n)/\epsilon)}$

**Verifier** for $\boldsymbol{P}$
$\boldsymbol{\#rounds = O(diam(G))}$

# Applications

**Corollary.** **$\epsilon$-tester** in the randomized CONGEST-model for:

- **Bipartiteness**. #rounds $= \boldsymbol{O((\log n)/\epsilon)}$,
  - Lower bound $\boldsymbol{\Omega(\log n)}$ [Censor-Hillel, Fischer, Schwartzman, Vasudev. 2016].
  - Improves over $\mathbf{Poly}(\boldsymbol{\epsilon^{-1}\log n})$ in the bounded degree model of [CHFSV 2016]
- **Cycle-Freeness**. #rounds $= \boldsymbol{O((\log n)/\epsilon)}$.

**Theorem.** $\exists$ **$\epsilon$-corrector** for Cycle-Freeness in the randomized CONGEST-model.

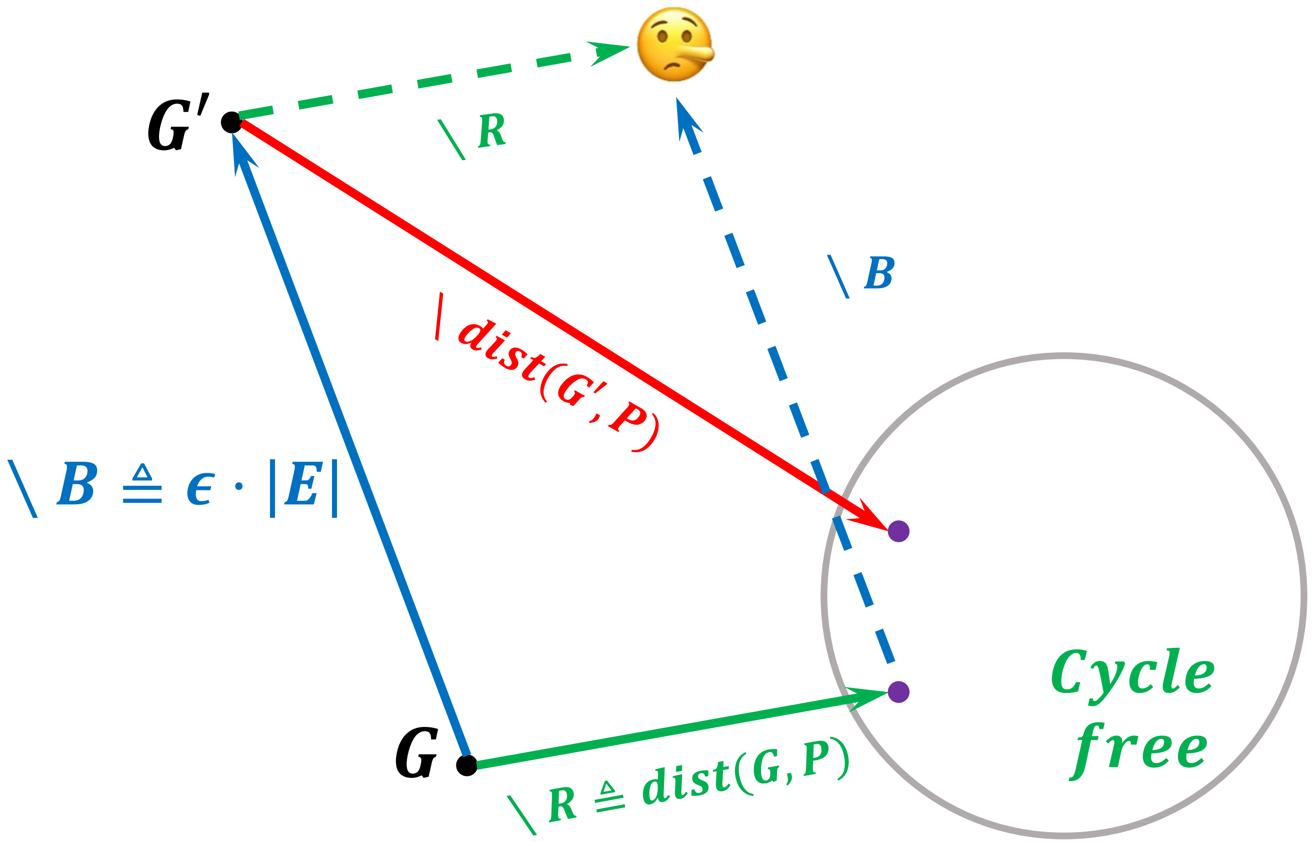
- #rounds $= \boldsymbol{O((\log n)/\epsilon)}$.

An algorithm is **$\epsilon$-corrector** for property $\boldsymbol{P}$ if:

- $E' \subseteq E$,
- $G(V, E \setminus E') \in P$,
- $|E'| \le dist(G,P) + \epsilon \cdot |E|$,
- Upon termination $\forall v \in V$ : $knows\ E'(v)$.

**Proof sketch.** Need to show:

$$\epsilon \cdot |E| + dist(\boldsymbol{G'}, P) \le \epsilon \cdot |E| + dist(\boldsymbol{G}, P)$$




**A Corrector.**

$\boldsymbol{V}_1$ $\boldsymbol{V}_2$

$\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$

$\boldsymbol{V}_3$

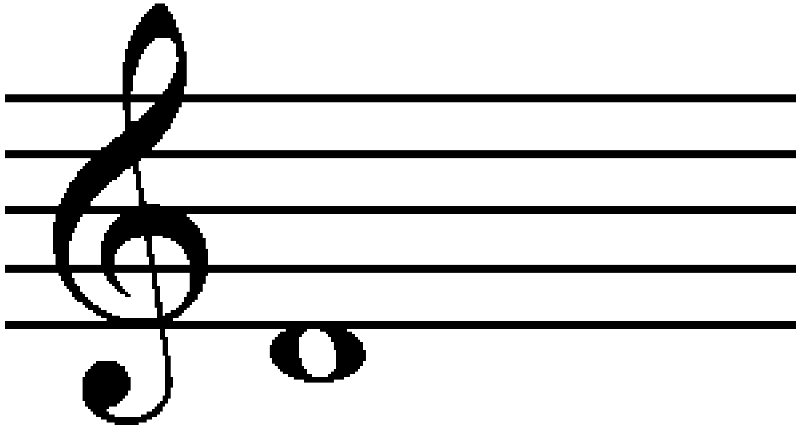
# 1st Intermezzo

*Questions?*

- My email: moti.medina@gmail.com
- Link to this note: https://arxiv.org/abs/1705.04898
  - “Faster and Simpler Distributed Algorithms for Testing and Correcting Graph Properties in the CONGEST-Model” by Guy Even, Reut Levi, and Moti Medina.

*Thank you!*

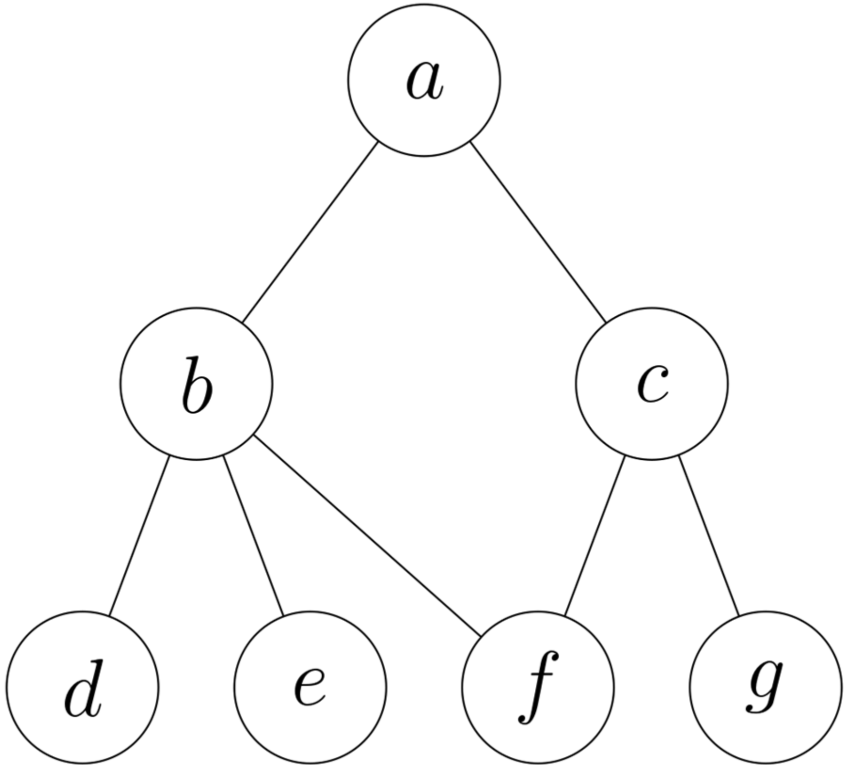
# Second Note

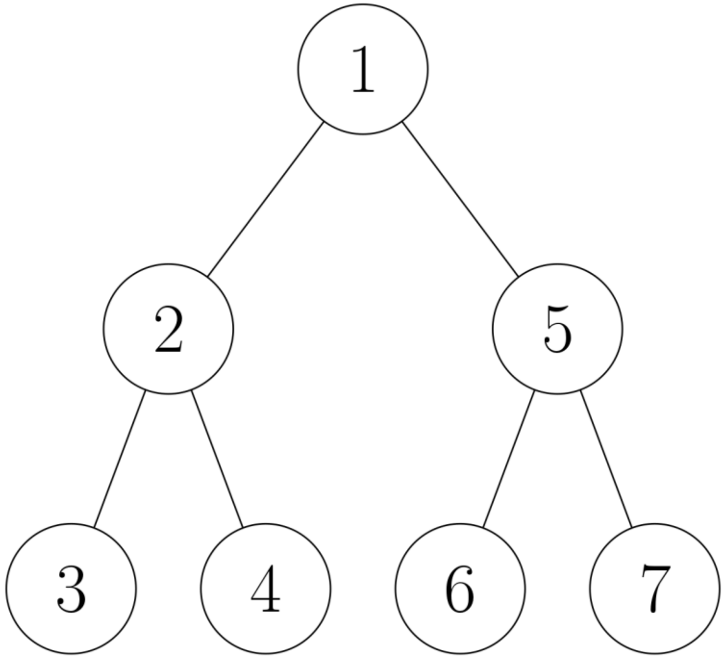


In the CONGEST model, it is possible to check the presence of a fixed tree **T** of constant size, in **O(1)** rounds, deterministically.

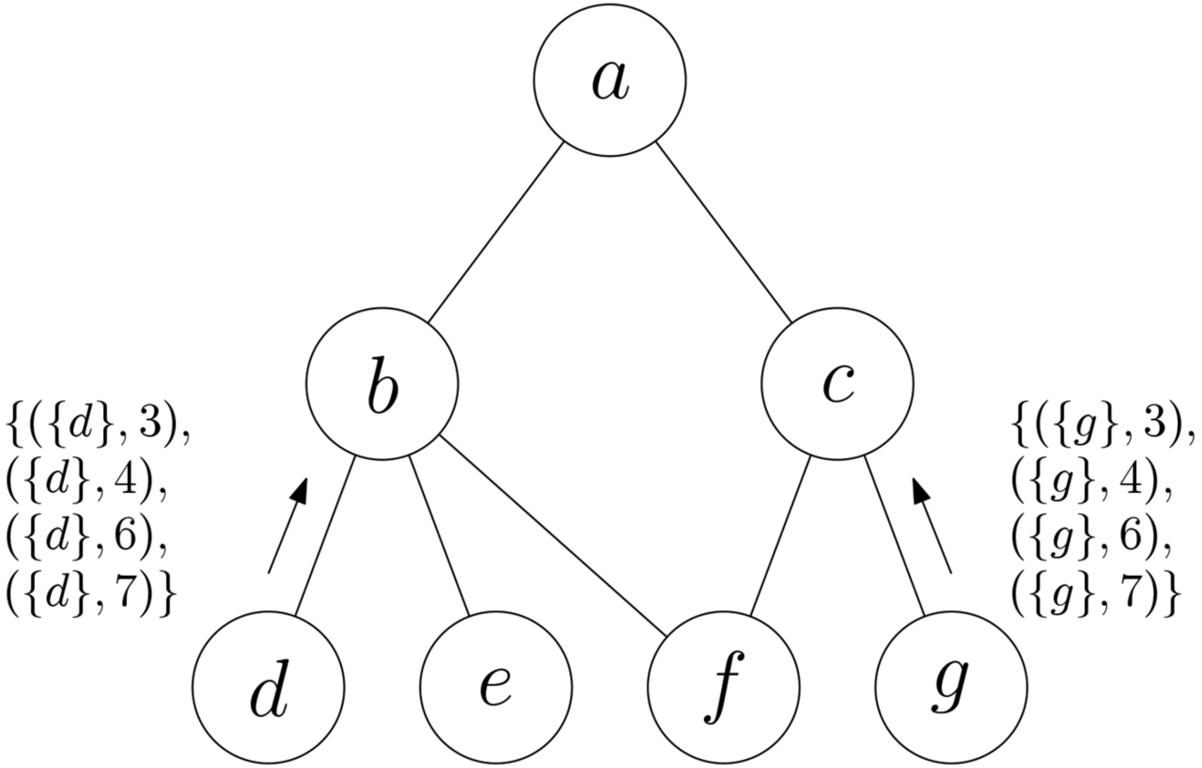
There exists an $\epsilon$-tester for **H** freeness, for any graph **H** of constant size composed by a tree, an edge, and arbitrary connections between the endpoints of the edge and the nodes of the tree, that requires $\mathbf{O(1/\epsilon)}$ rounds in the CONGEST model.

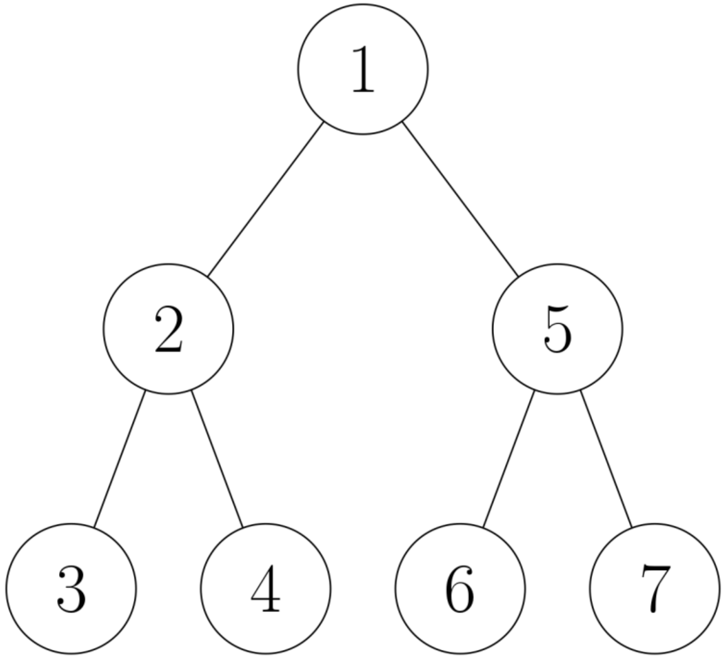
# Tree Detection



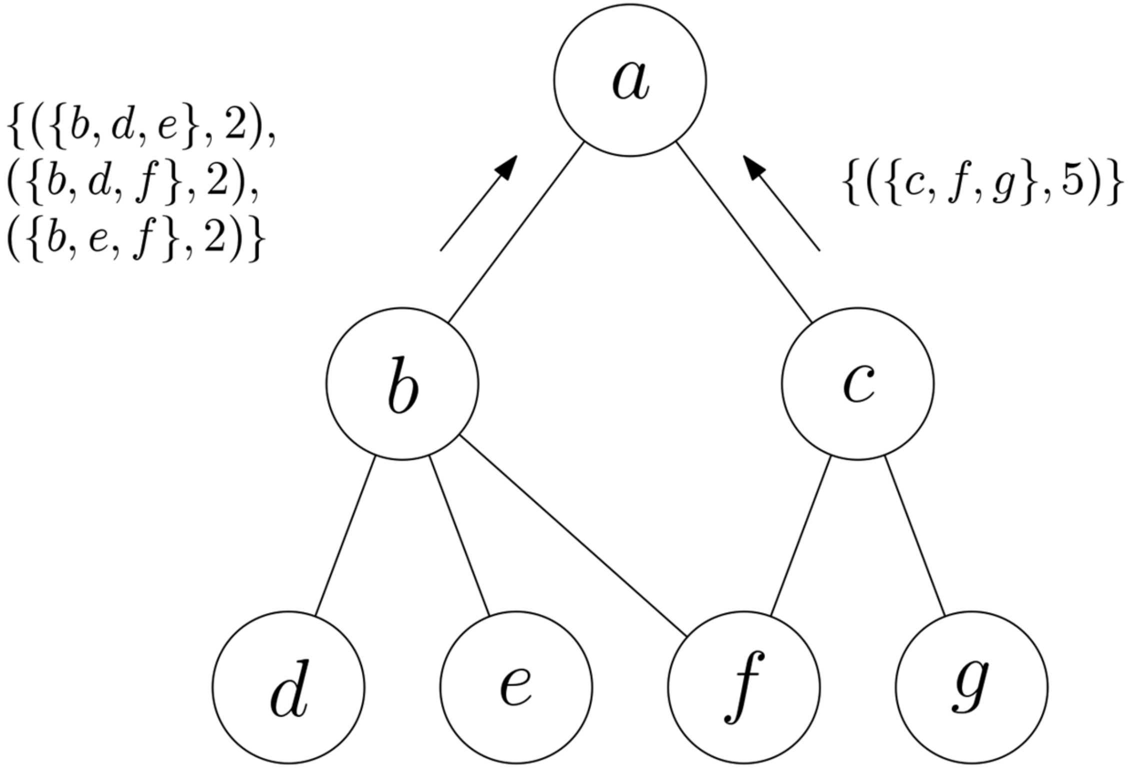



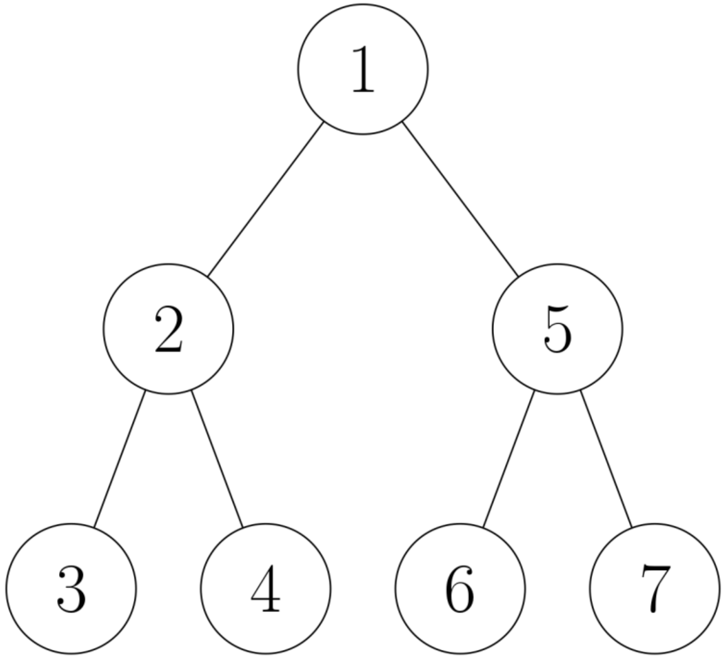
# Tree Detection



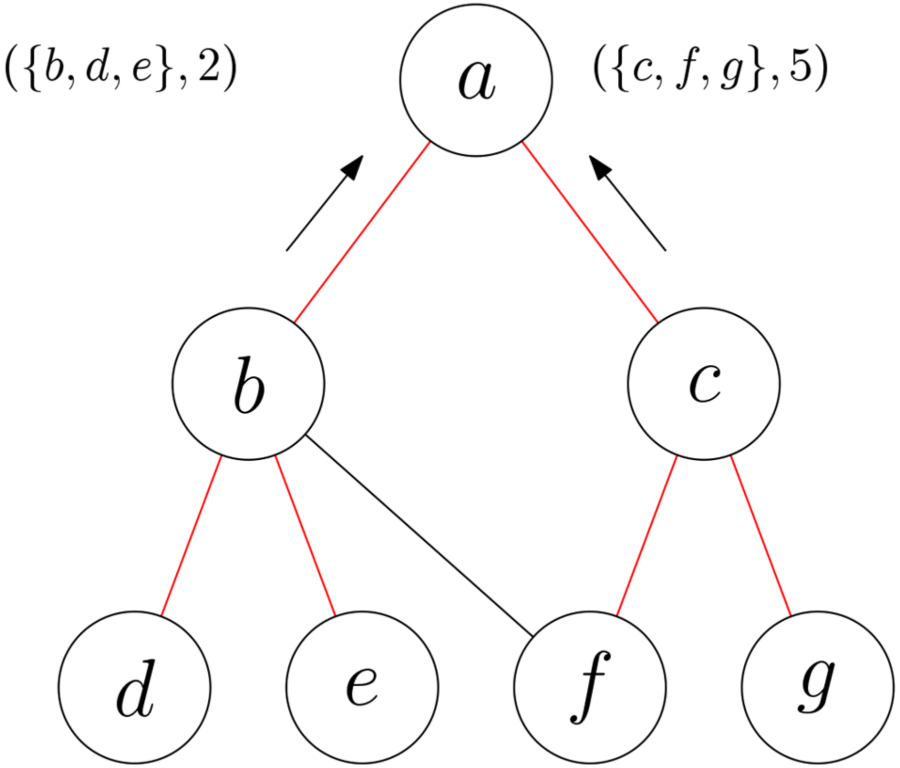



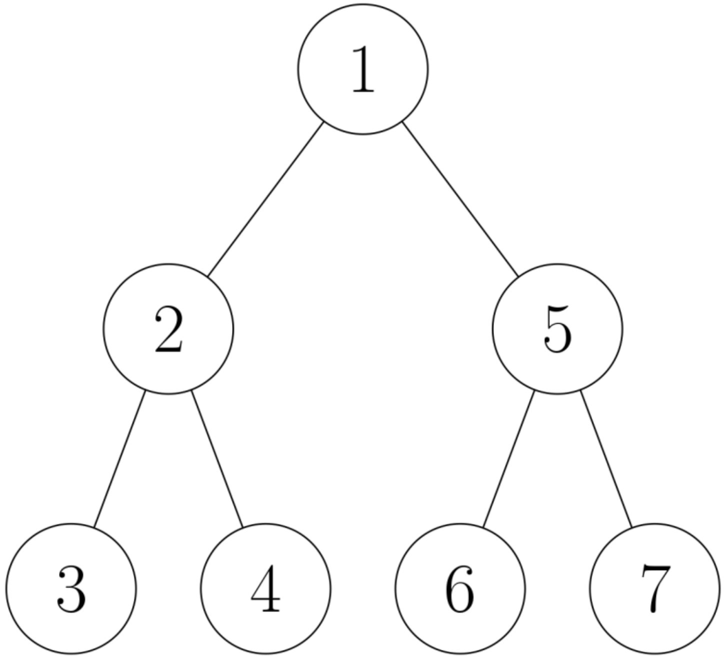
# Tree Detection



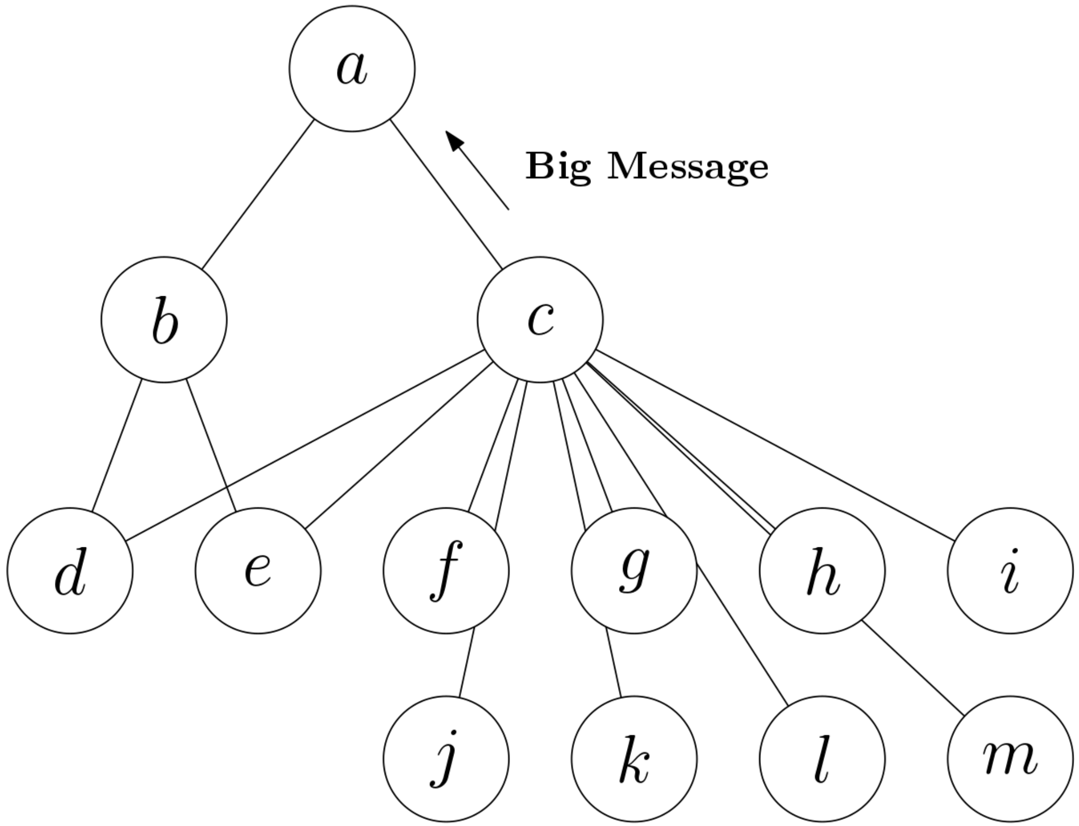



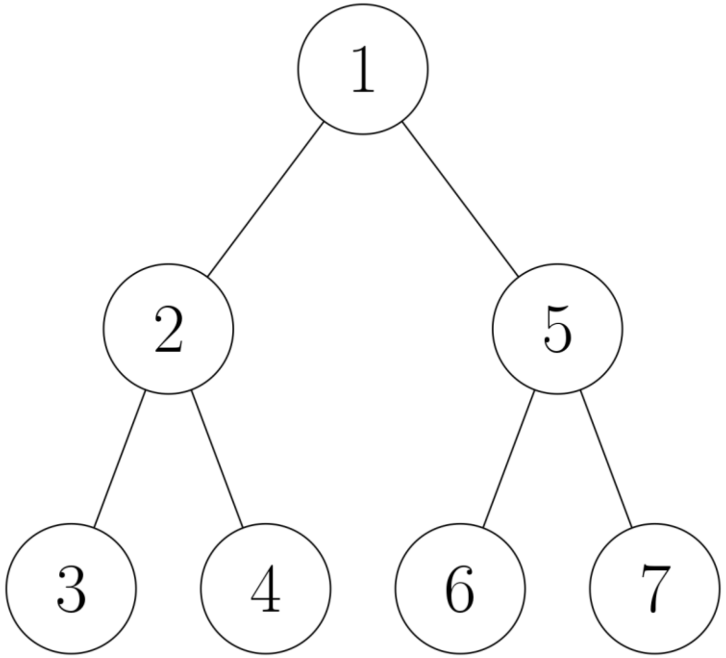
# Tree Detection



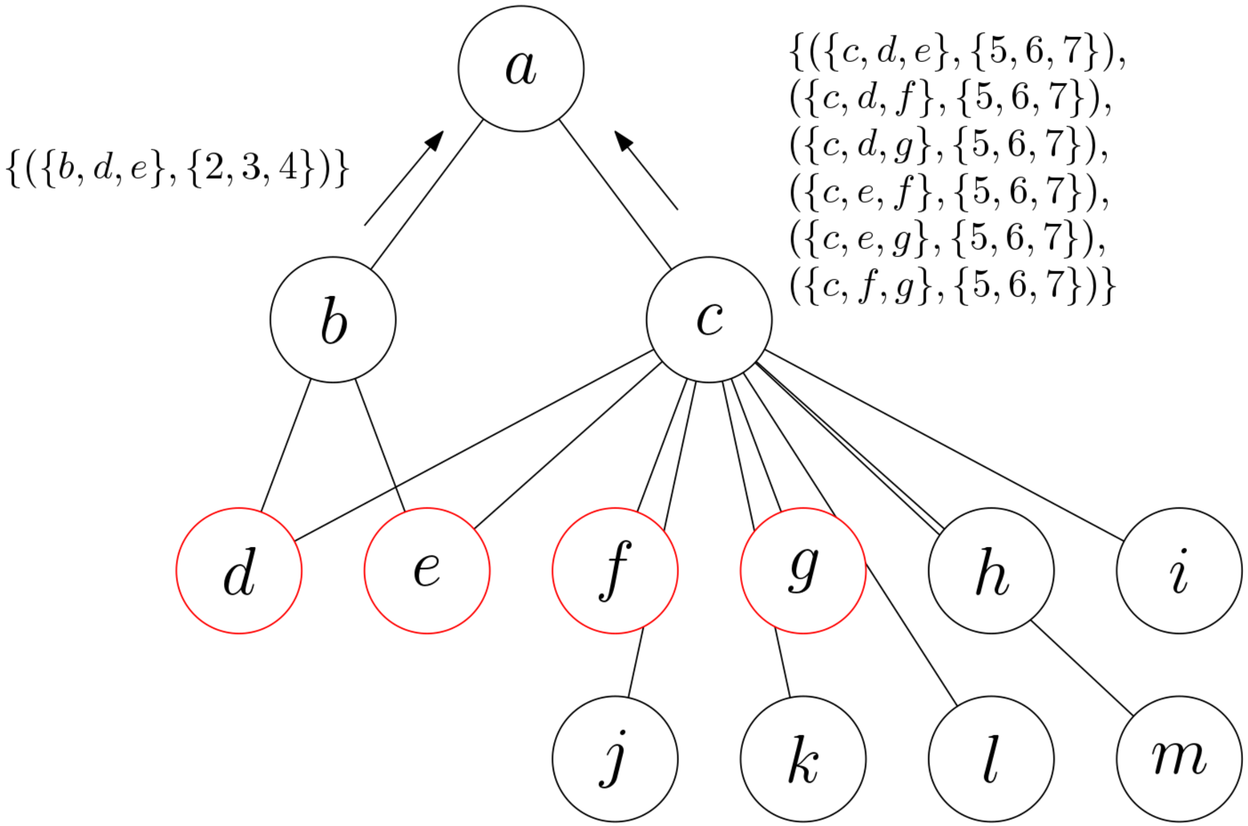



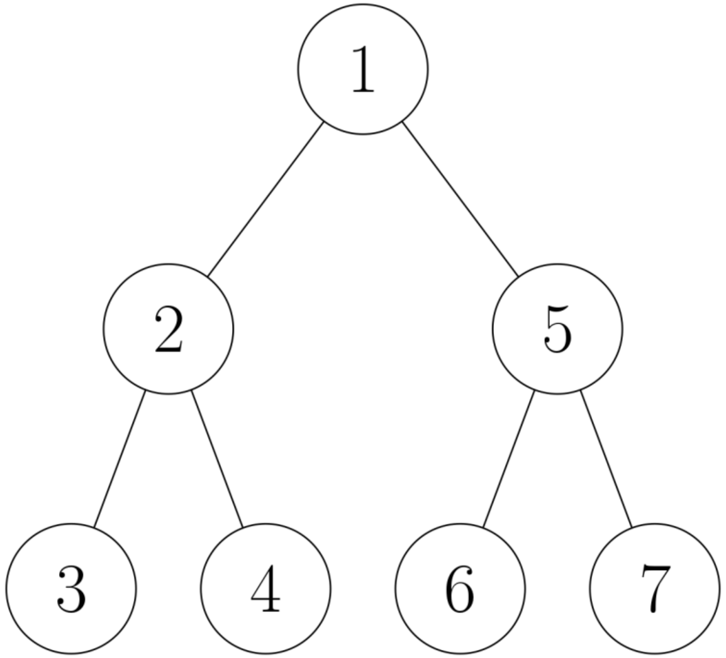
# Congestion



# Congestion


{({b, d, e}, {2, 3, 4})}
{({c, d, e}, {5, 6, 7}),
({c, d, f}, {5, 6, 7}),
({c, d, g}, {5, 6, 7}),
({c, e, f}, {5, 6, 7}),
({c, e, g}, {5, 6, 7}),
({c, f, g}, {5, 6, 7})}
a
b
c
d
e
f
g
h
i
j
k
l
m


1
2
5
3
4
6
7

# Sparsification of the intermediate solutions

Given a set of sets S, we need to find a representative set R, such that:

- It is small
- **R ⊆ S**
- For any other possible set **t** (of some constant fixed length), if there is a set **s∈S** disjoint with **t**, then there is also a set **r∈R** disjoint with **t**.

*Lemma [Erdős, Hajnal, Moon '64]:*
R is of constant size.

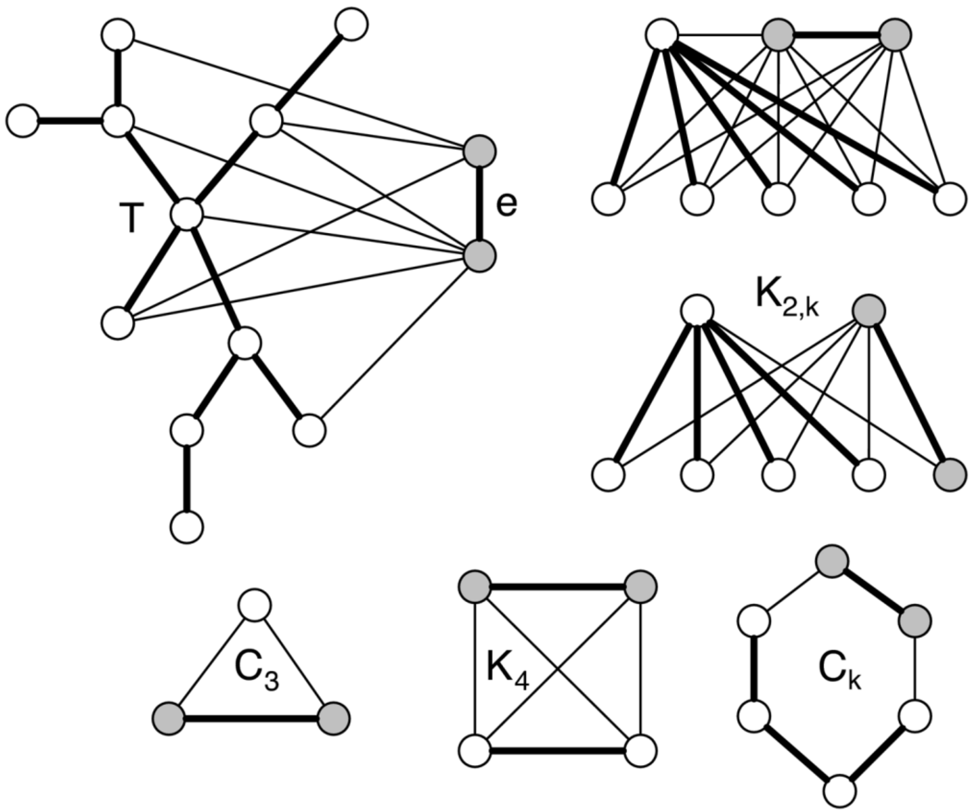
# Representative Sets

R = {(1, 2), (4, 5), (6, 7)} is a representative set of
S = {(1, 2), (1, 3), (4, 5), (6, 7), (8, 9), (8, 10), (8, 11), (9, 12)}

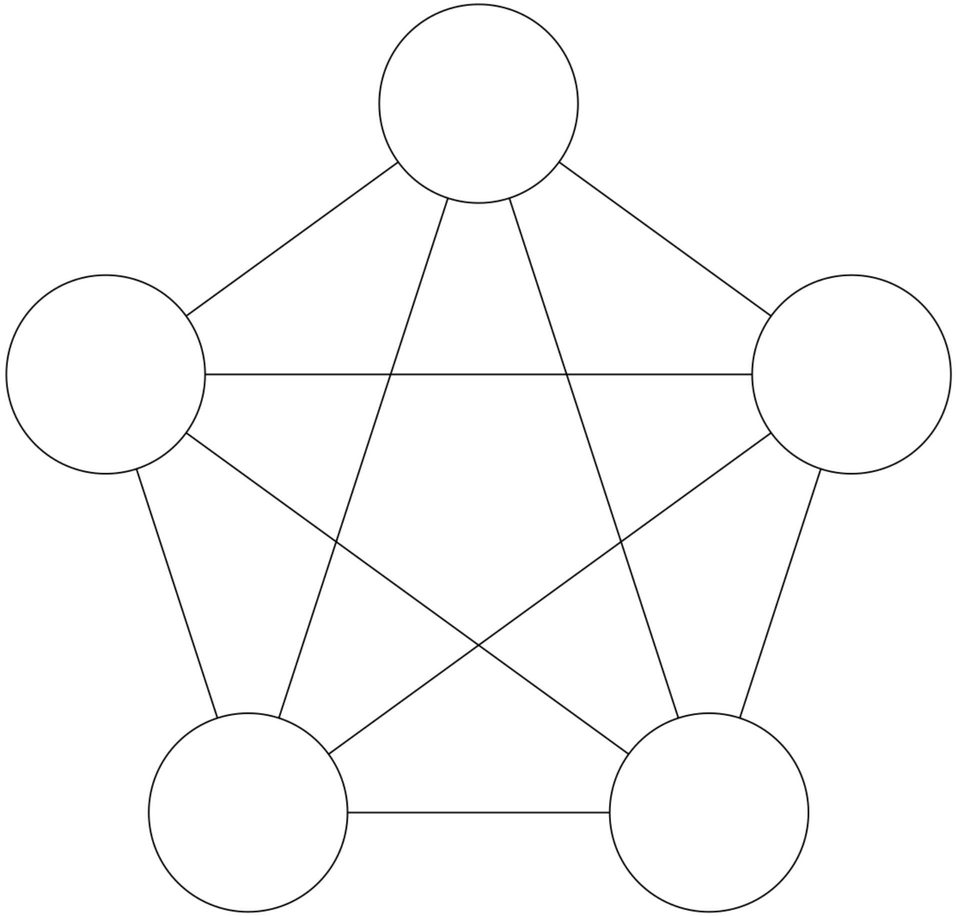
| Given: | Disjoint with it: |
|---|---|
| (1,2) | (4,5) |
| (4,5) | (6,7) |
| (1,4) | (6,7) |
| (10,20) | (1,2) |

# Property testing

1. Choose one edge uniformly at random
2. Execute (a slightly modified version of) the tree detection algorithm



# Open Problems

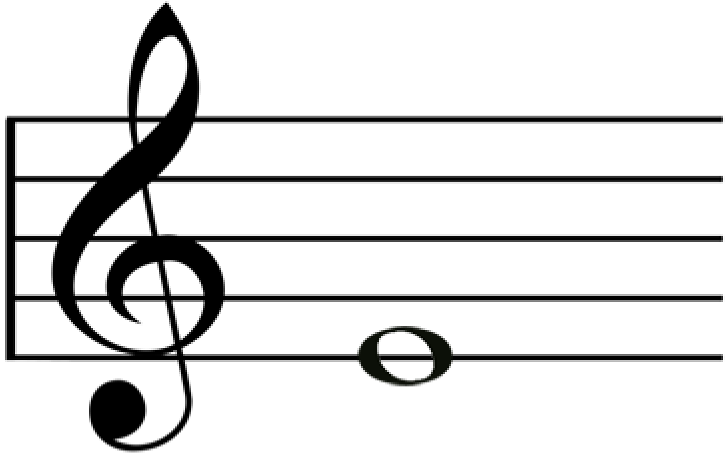


# 2nd Intermezzo

*Questions?*

*Thank you!*

# Third Note



1. Simpler algorithm for $C_k$-freeness in $O\left(\frac{1}{\epsilon}\right)$ rounds
2. Algorithm for finding any tree $T$ in $O(1)$ rounds (exact)
3. Combination: general class, including all 5-vertex graphs except $K_5$, in $O\left(\frac{1}{\epsilon}\right)$ rounds
4. Algorithm for $k$-clique freeness in $O\left(m^{\frac{1}{2}-\frac{1}{k-2}} \cdot \epsilon^{-\frac{1}{2}-\frac{1}{k-2}}\right)$ rounds
   - For triangles: if $\epsilon \geq \min\left\{m^{-\frac{1}{3}}, \frac{n}{m}\right\}$, in $O(1)$ rounds!

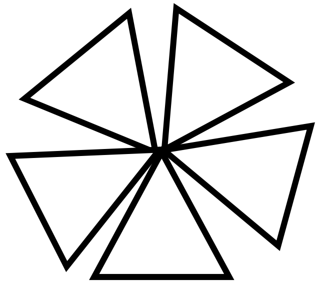
# Main Ingredient #1: Disjoint Copies

Well-known observation:

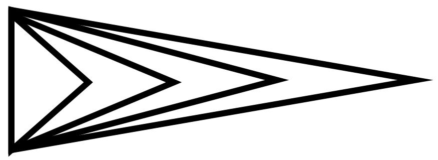
- If $G$ is $\epsilon$-far from $H$-free, then
- $G$ contains $\frac{\epsilon \cdot m}{|E(H)|}$ edge-disjoint copies of $H$

$\Rightarrow$ random edge participates in $H$ w.p. $\geq \epsilon$

YES

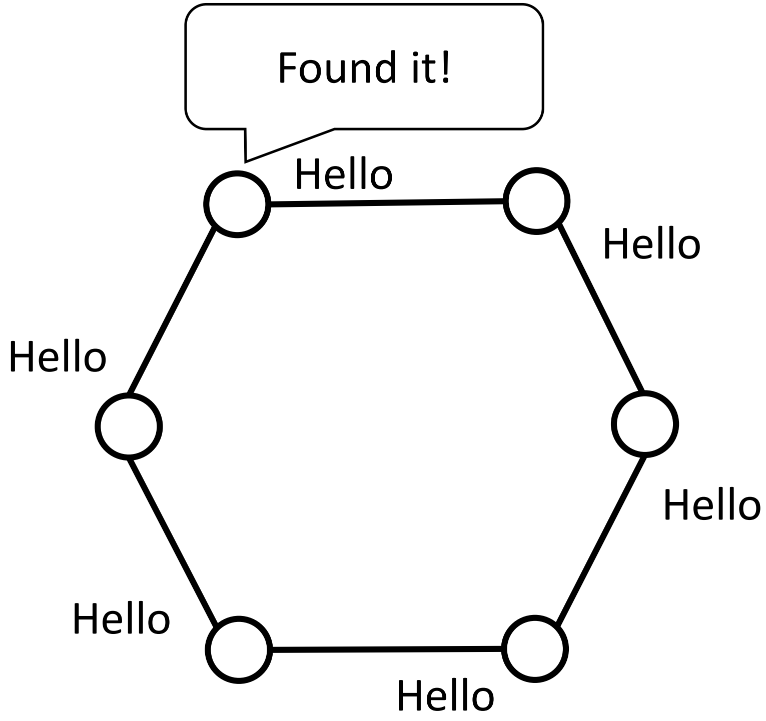


NO



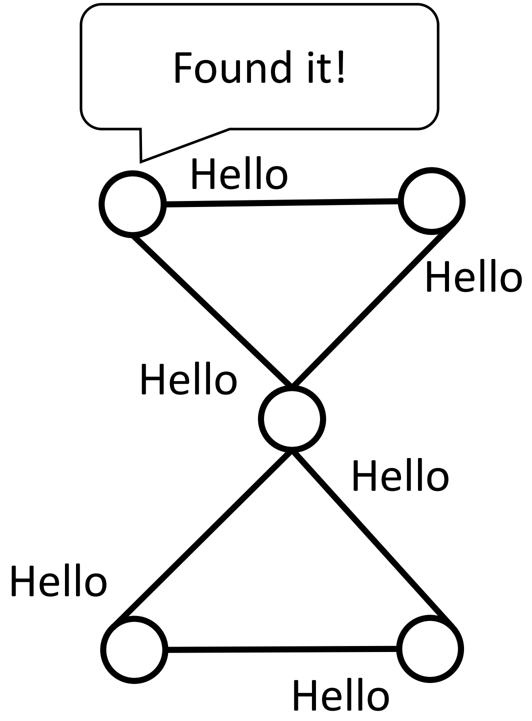
# Main Ingredient #2: Color Coding

- [ Alon, Yuster, Zwick '95 ]
- Idea: to find $C_k$,



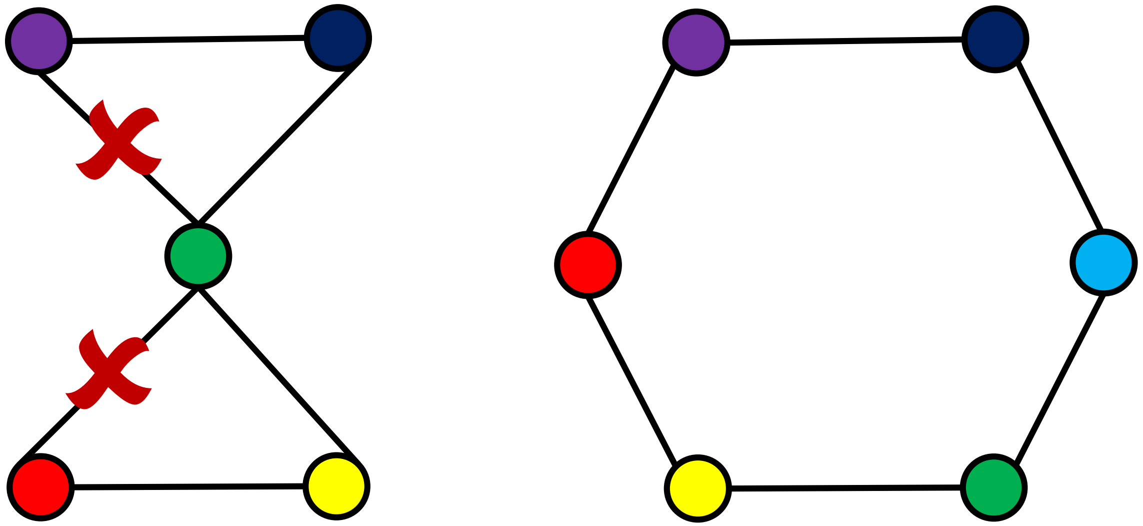
# Main Ingredient #2: Color Coding

- [ Alon, Yuster, Zwick '95 ]
- The problem…



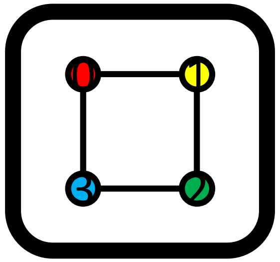
# Main Ingredient #2: Color Coding

- [ Alon, Yuster, Zwick '95 ]
- Solution:

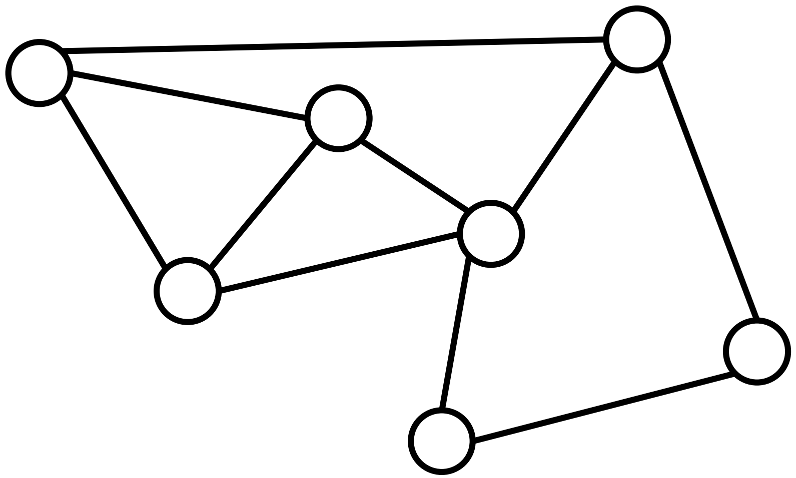


# Algorithm 1: $C_k$-freeness

$H$

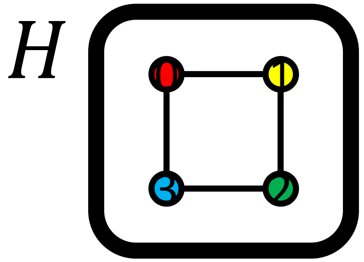


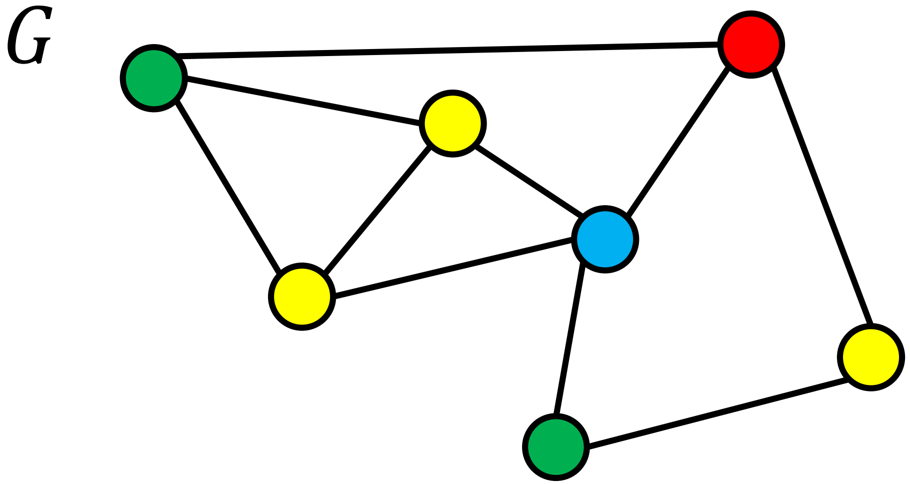
$G$



# Algorithm 1: $C_k$-freeness

- Step 1: color coding

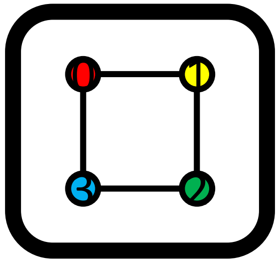


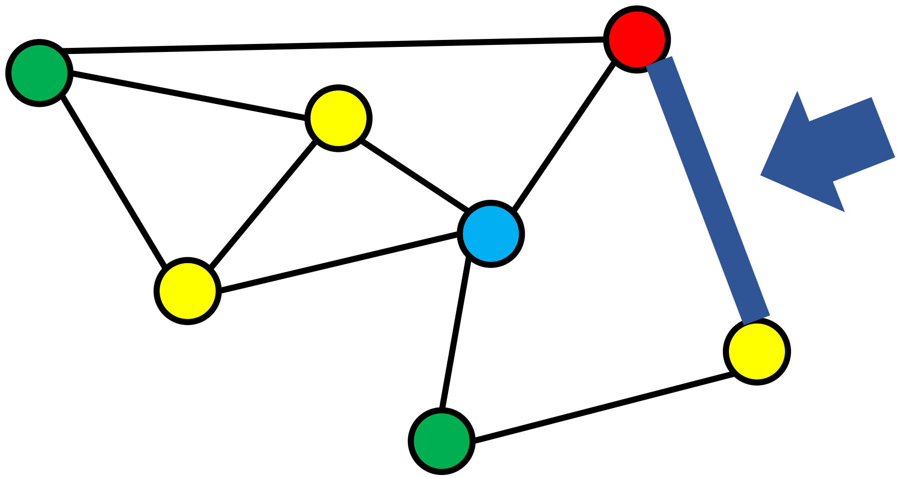
# Algorithm 1: $C_k$-freeness

- Step 1: color coding
- Step 2: select random directed edge colored (0,1)

$H$

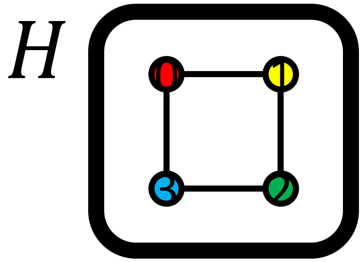


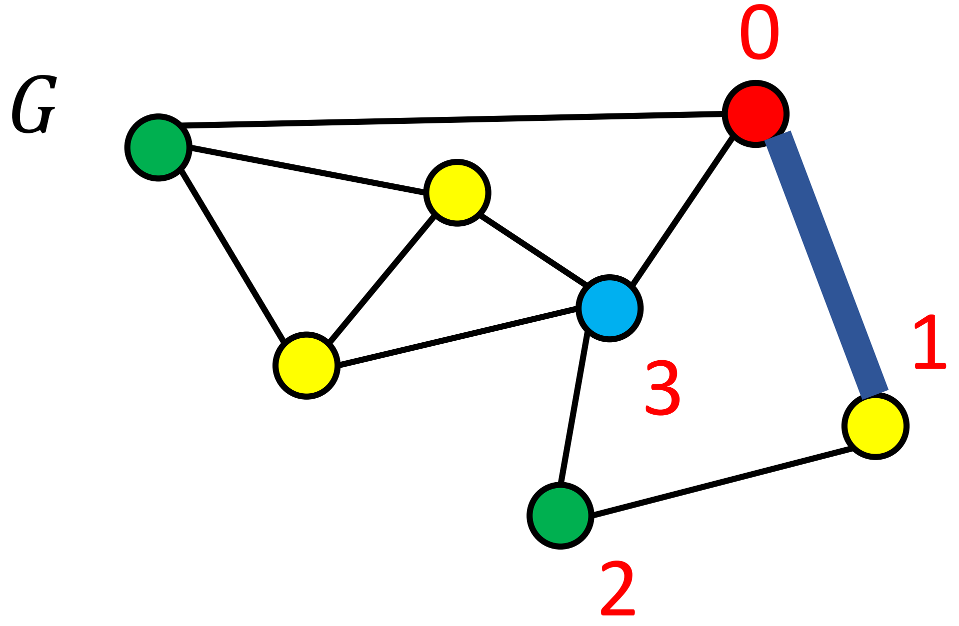
$G$



# Algorithm 1: $C_k$-freeness

- Step 1: color coding
- Step 2: select random directed edge colored (0,1)
- Step 3: color-coded BFS



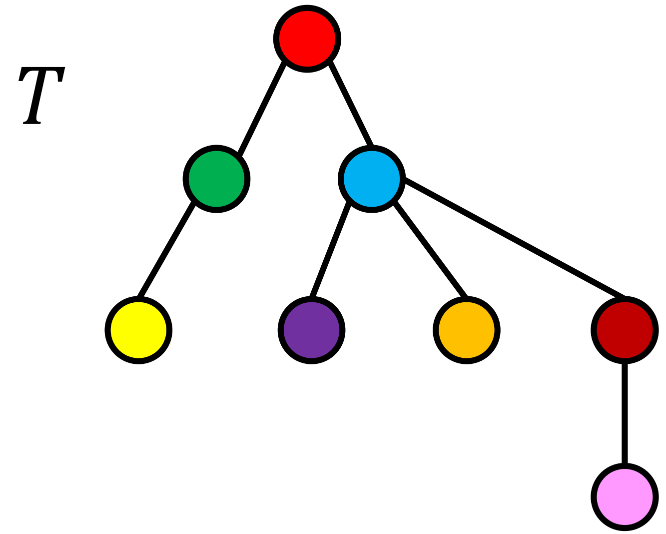
# Algorithm 1: $C_k$-freeness

- Step 1: color coding
- Step 2: ~~select random directed edge colored (0,1)~~
- Step 3: color-coded BFS

- Assign random weight to each edge
- Defer to lowest-weight edge

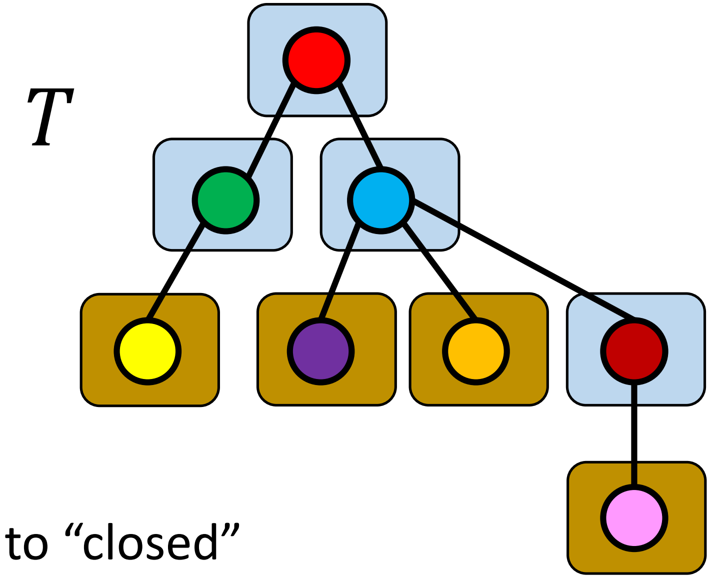
# Algorithm 2: $T$-freeness

- Step 1: color coding



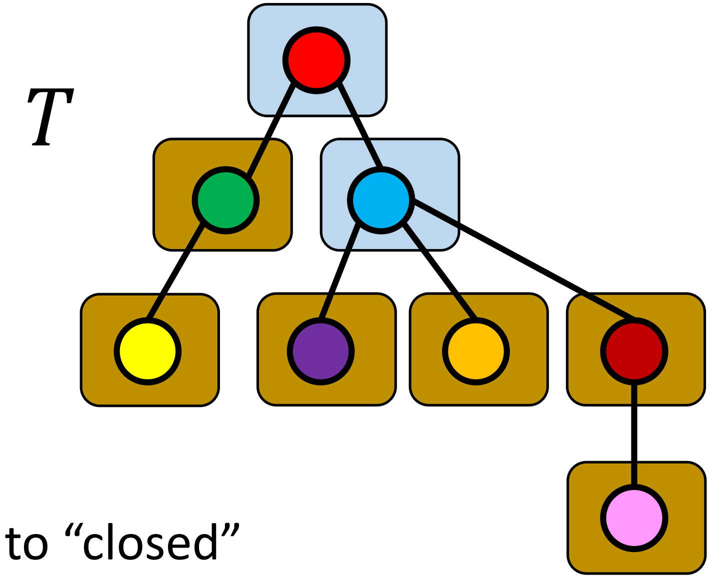
# Algorithm 2: $T$-freeness

- Step 1: color coding
- Step 2: convergecast
  - Initially:
    - State = “closed” if color = leaf of $T$
    - State = “open” otherwise
  - In each round: send (state, color)
    - If received (“closed”, $v$) for each child $v$ in $T$: set state to “closed”



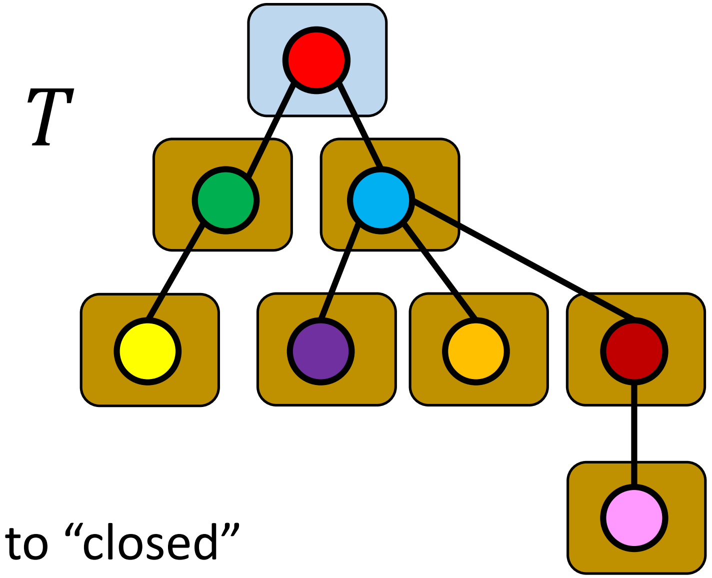
# Algorithm 2: $T$-freeness

- Step 1: color coding
- Step 2: convergecast
  - Initially:
    - State = “closed” if color = leaf of $T$
    - State = “open” otherwise
  - In each round: send (state, color)
    - If received (“closed”, $v$) for each child $v$ in $T$: set state to “closed”



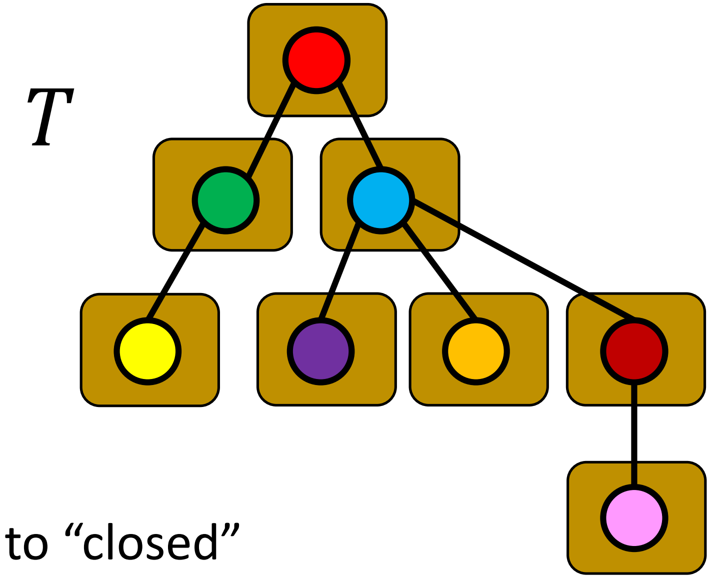
# Algorithm 2: $T$-freeness

- Step 1: color coding
- Step 2: convergecast
  - Initially:
    - State = “closed” if color = leaf of $T$
    - State = “open” otherwise
  - In each round: send (state, color)
    - If received (“closed”, $v$) for each child $v$ in $T$: set state to “closed”



# Algorithm 2: $T$-freeness

- Step 1: color coding
- Step 2: convergecast
  - Initially:
    - State = "closed" if color = leaf of $T$
    - State = "open" otherwise
  - In each round: send (state, color)
    - If received ("closed", $v$) for each child $v$ in $T$: set state to "closed"



# Combination

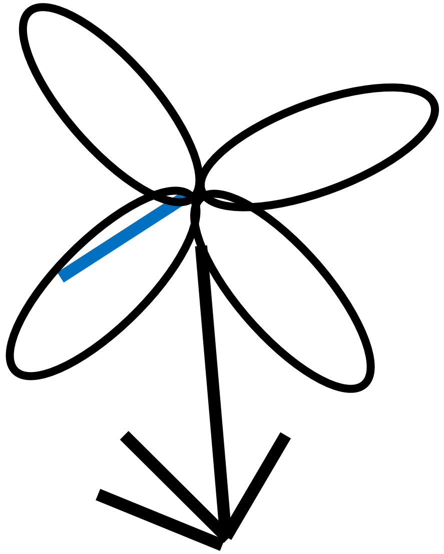
Characterization 1:

- $\exists$edge $\{u, v\}$ s.t. any cycle in $H$ contains $u$ or $v$ (or both)

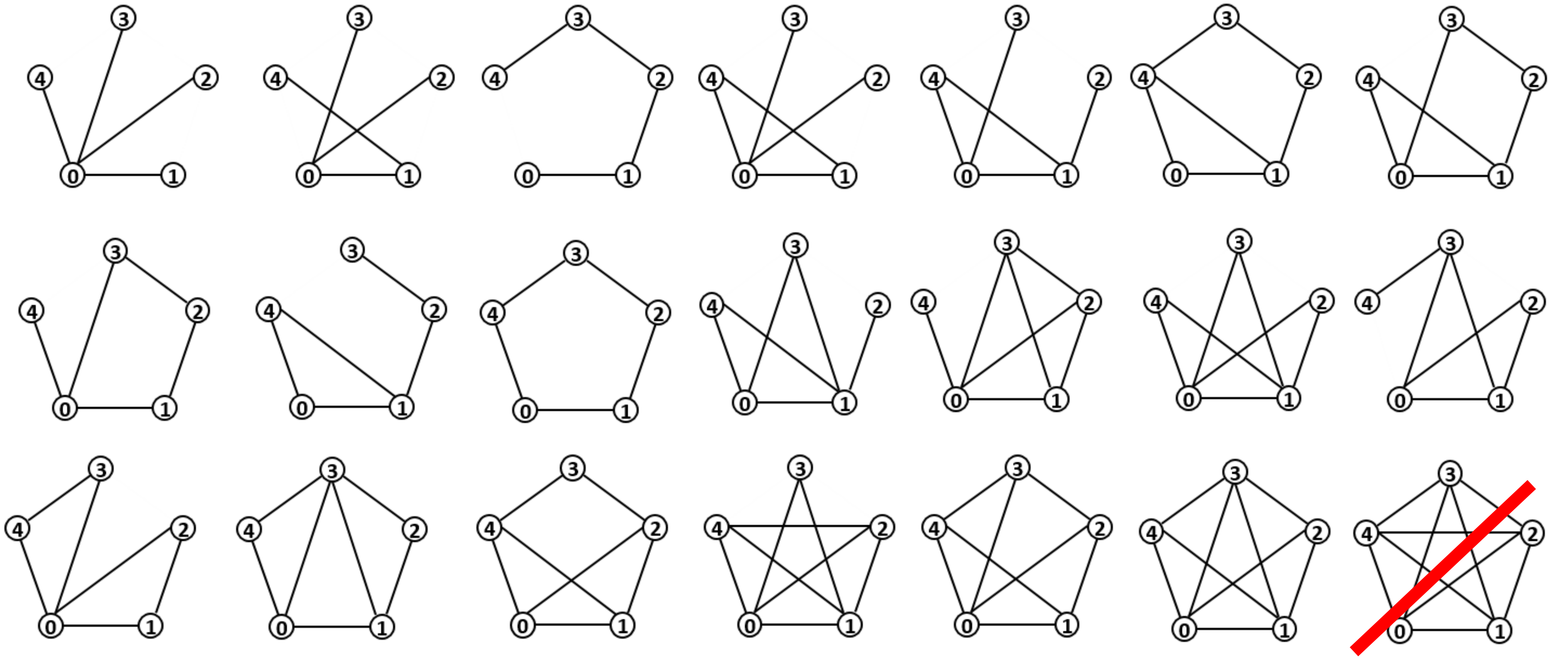
# Combination

Characterization 2:

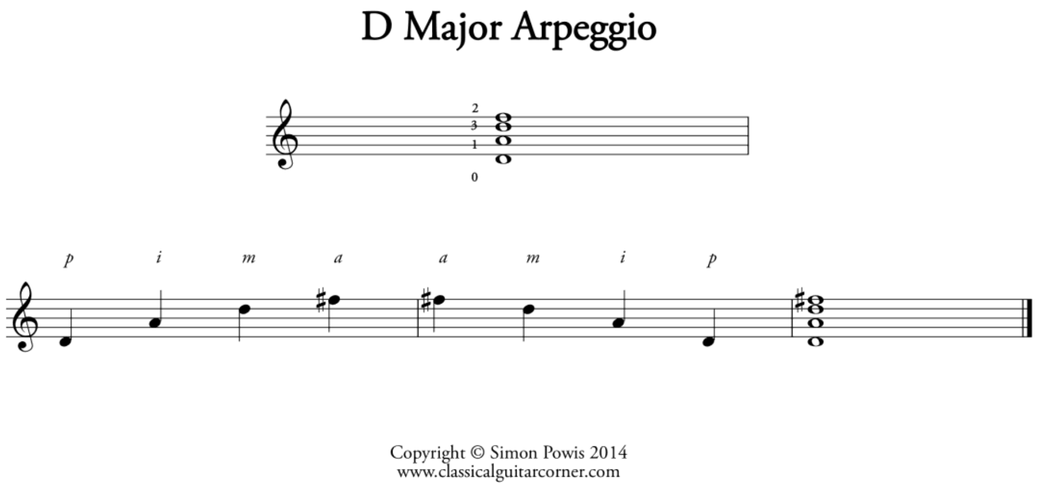
1. Start with edge $\{0,1\}$
2. Add “disjoint” cycles including $0, 1$ or both
3. Add “disjoint” trees rooted at prior nodes
4. Connect $0, 1$ freely



# Examples



# Finale

Questions?


Thank you from all of us!