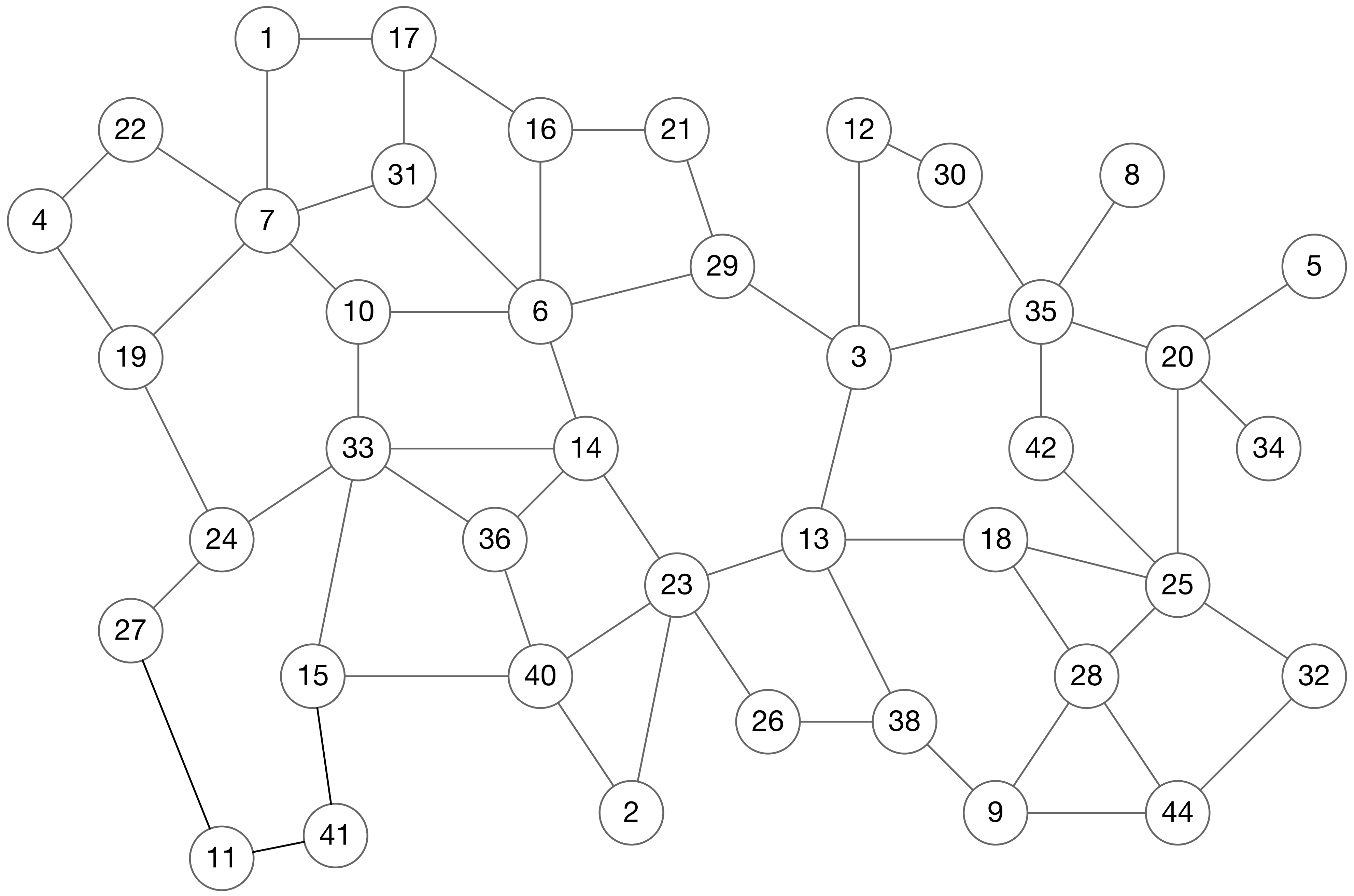
Distributed Edge Coloring in Time Polylogarithmic in Δ

Alkida Balliu [1], Sebastian Brandt [2], Fabian Kuhn [3], **Dennis Olivetti** [1]

[1] Gran Sasso Science Institute
[2] CISPA Helmholtz Center for Information Security
[3] University of Freiburg

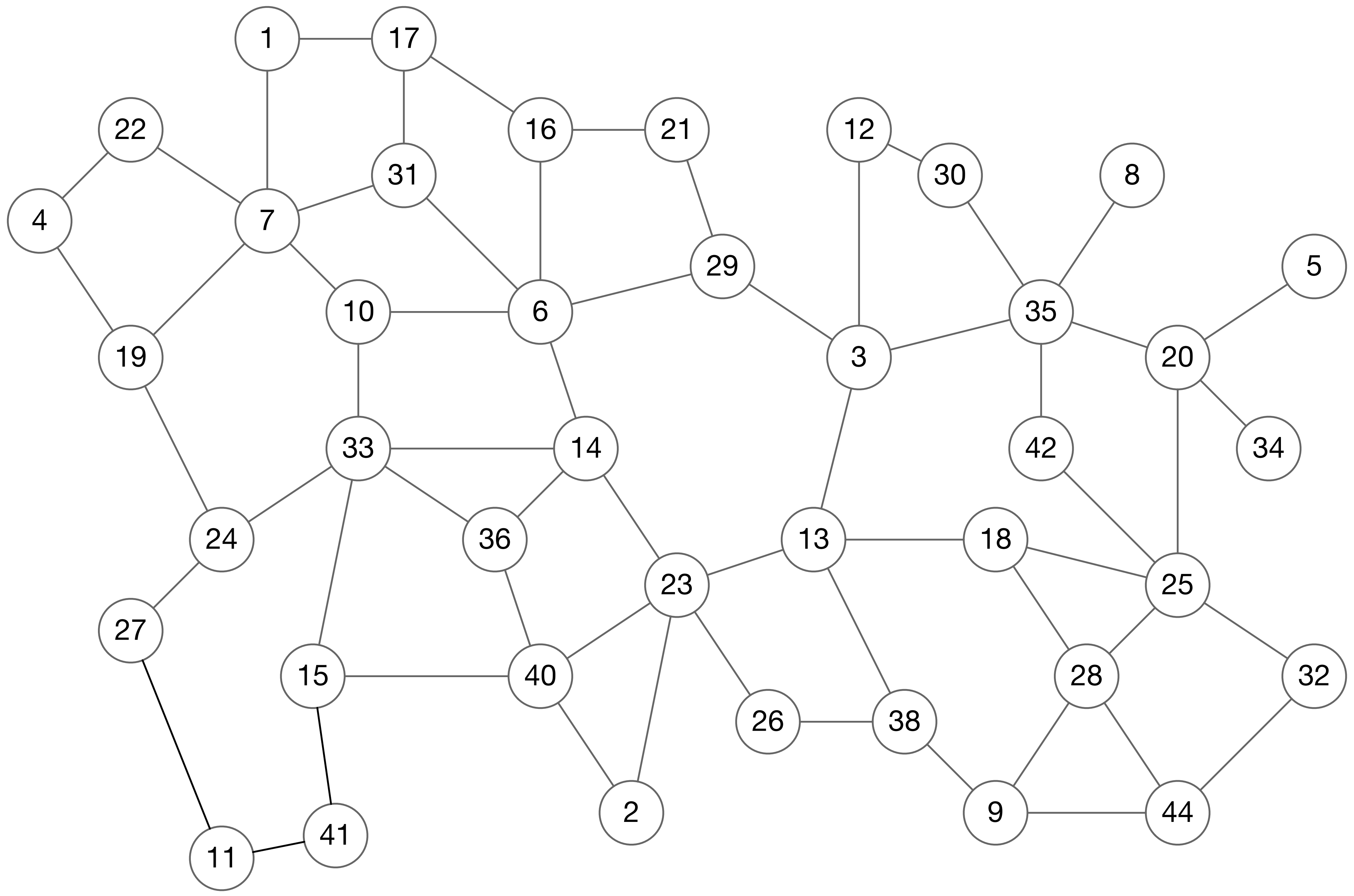
LOCAL model

- Undirected simple graph $G = (V, E)$ of n nodes and maximum degree Δ
- Each node has a unique ID
- Synchronous message passing model
- Unbounded computation
- Unbounded bandwidth

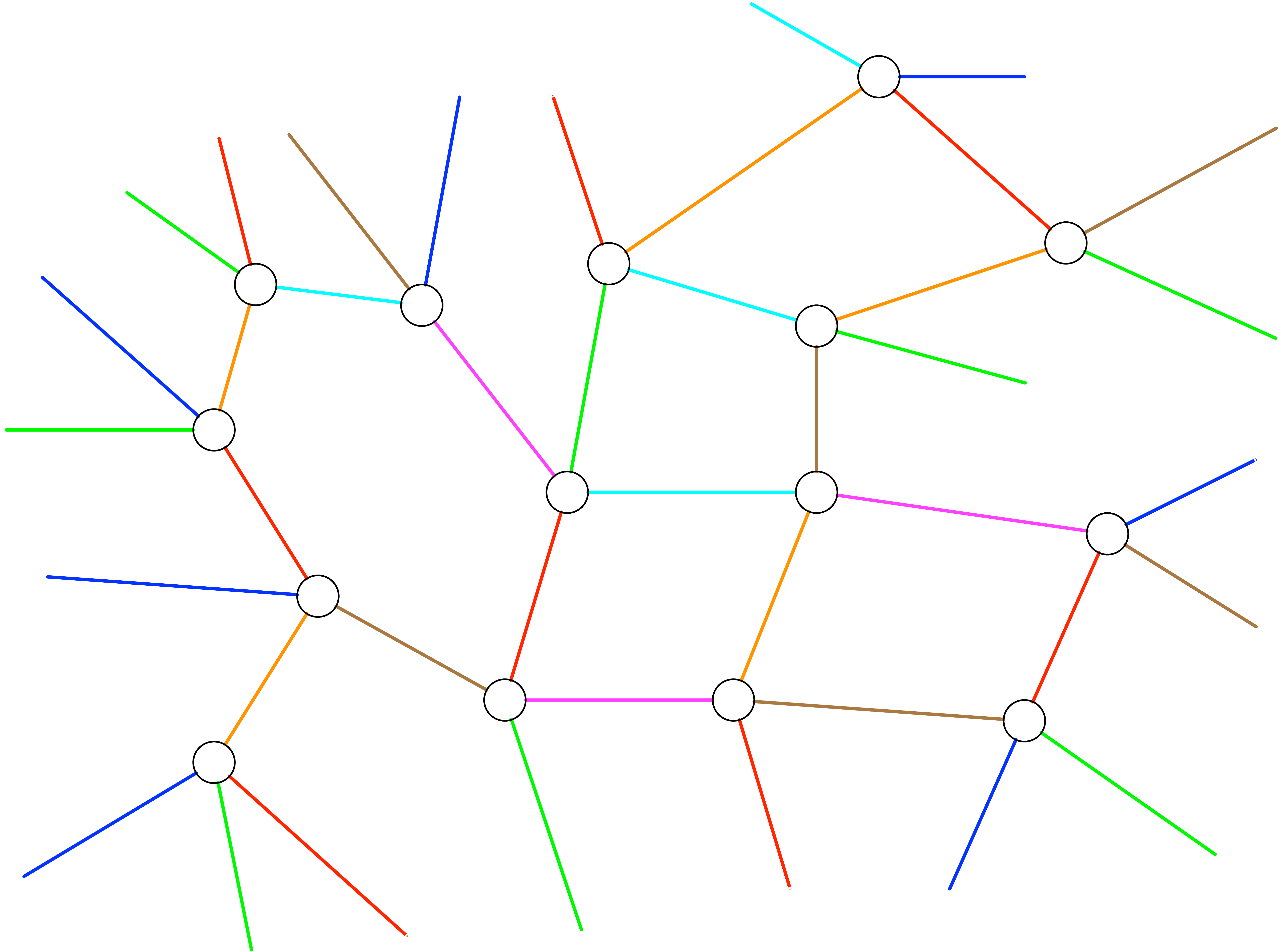


CONGEST model

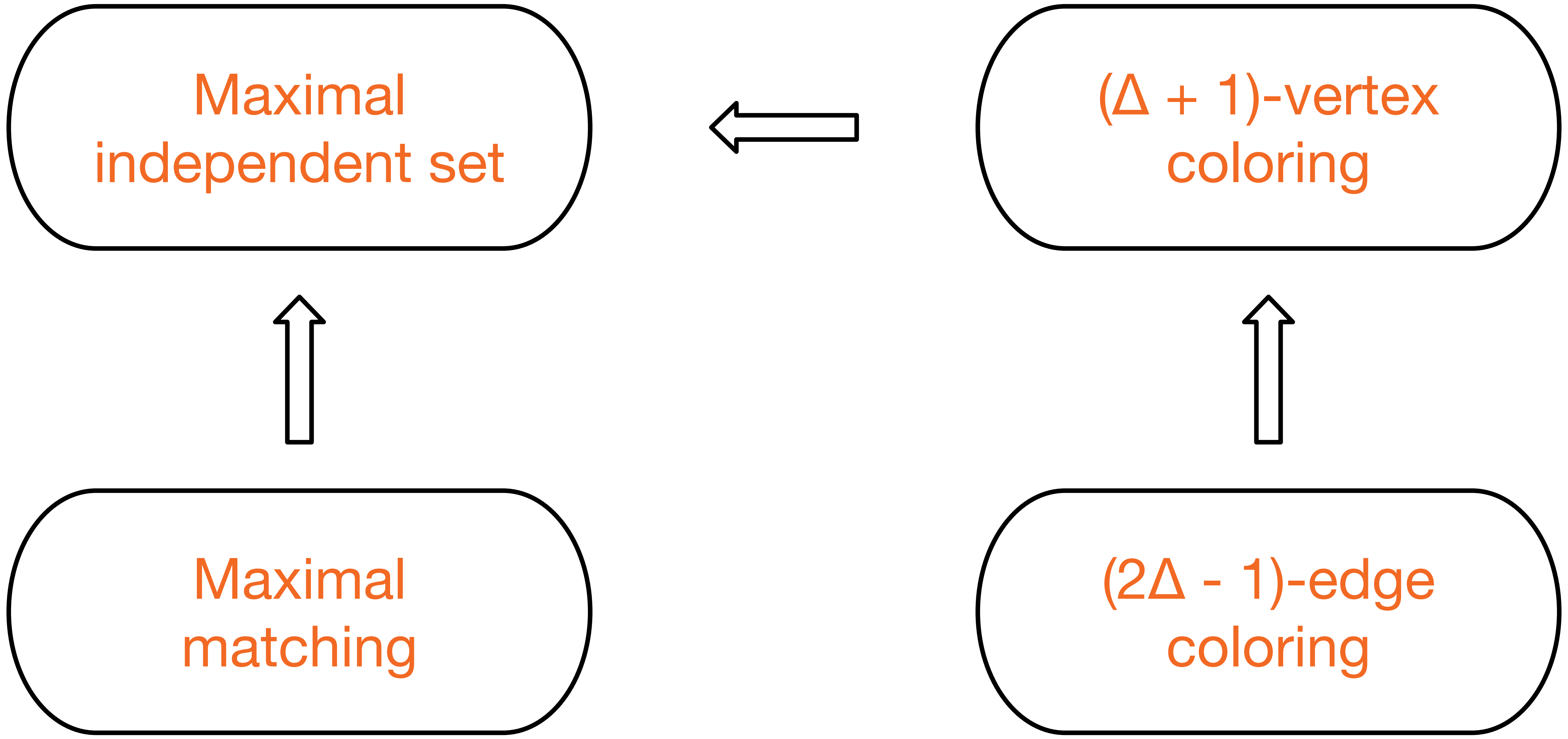
- Undirected simple graph $G = (V, E)$ of n nodes and maximum degree Δ
- Each node has a unique ID
- Synchronous message passing model
- Unbounded computation
- O(log n)-bit messages



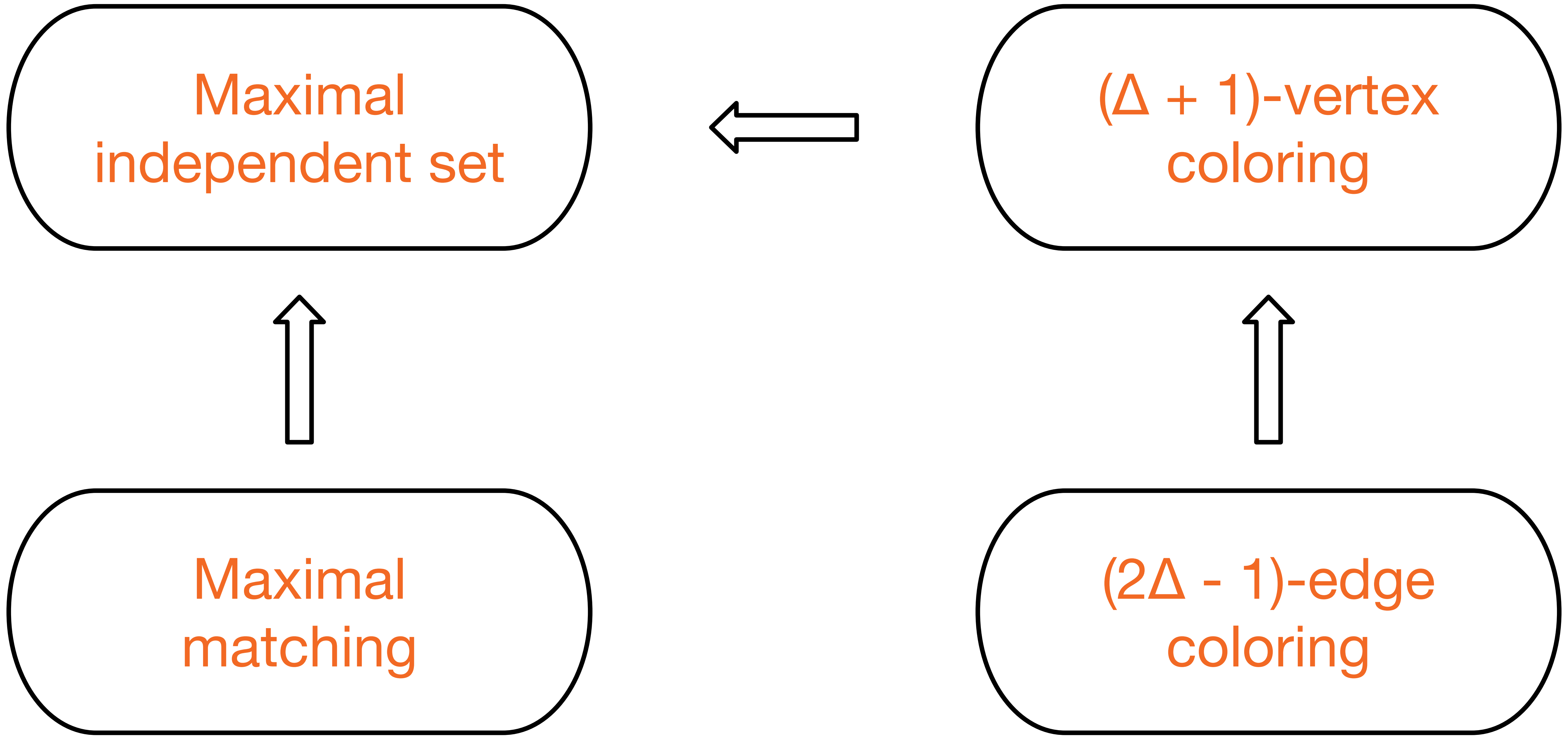
(2Δ - 1)-Edge Coloring



Four classical problems

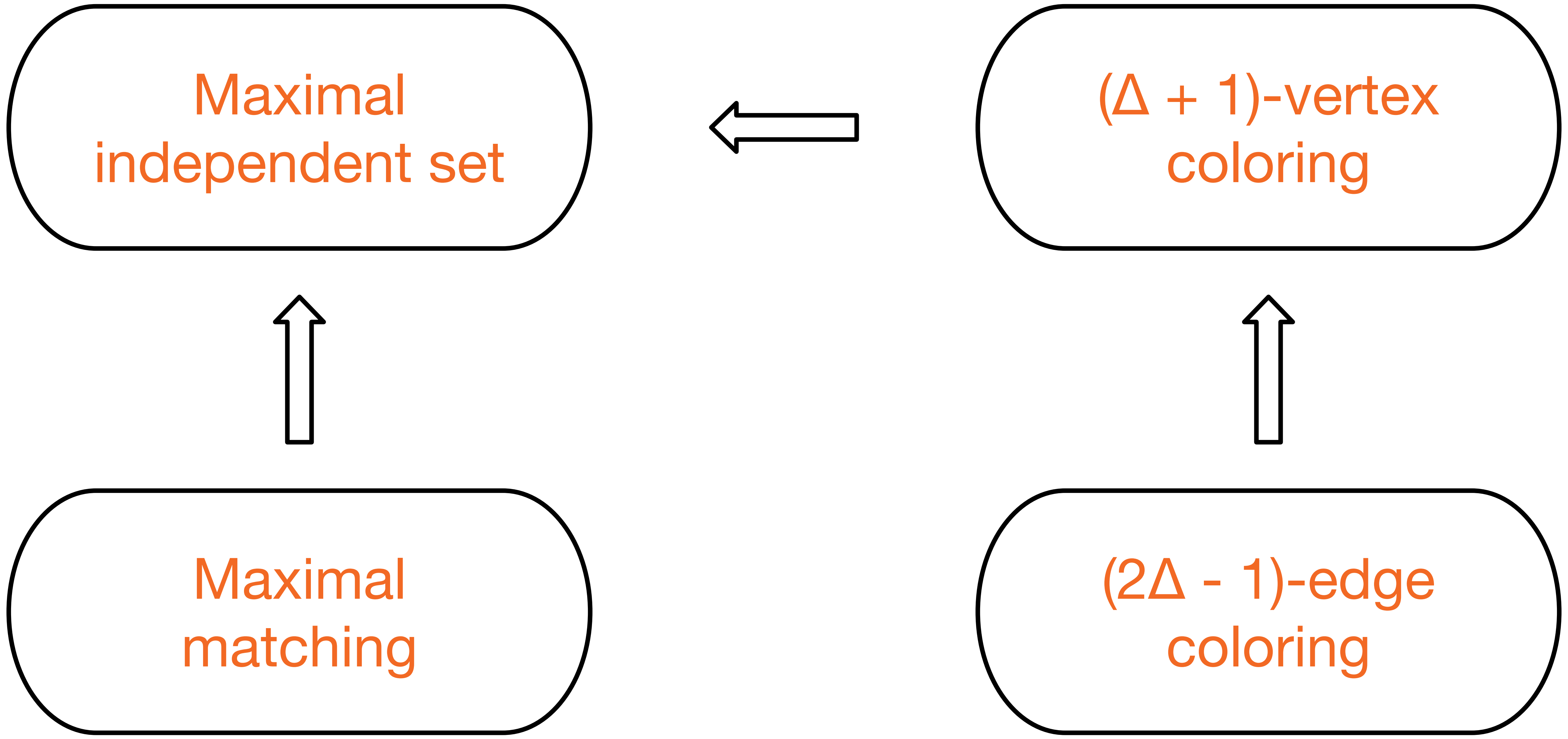


Four classical problems



These problems can be solved in $\operatorname{poly}\log n$ rounds [Rozhon, Ghaffari '20]

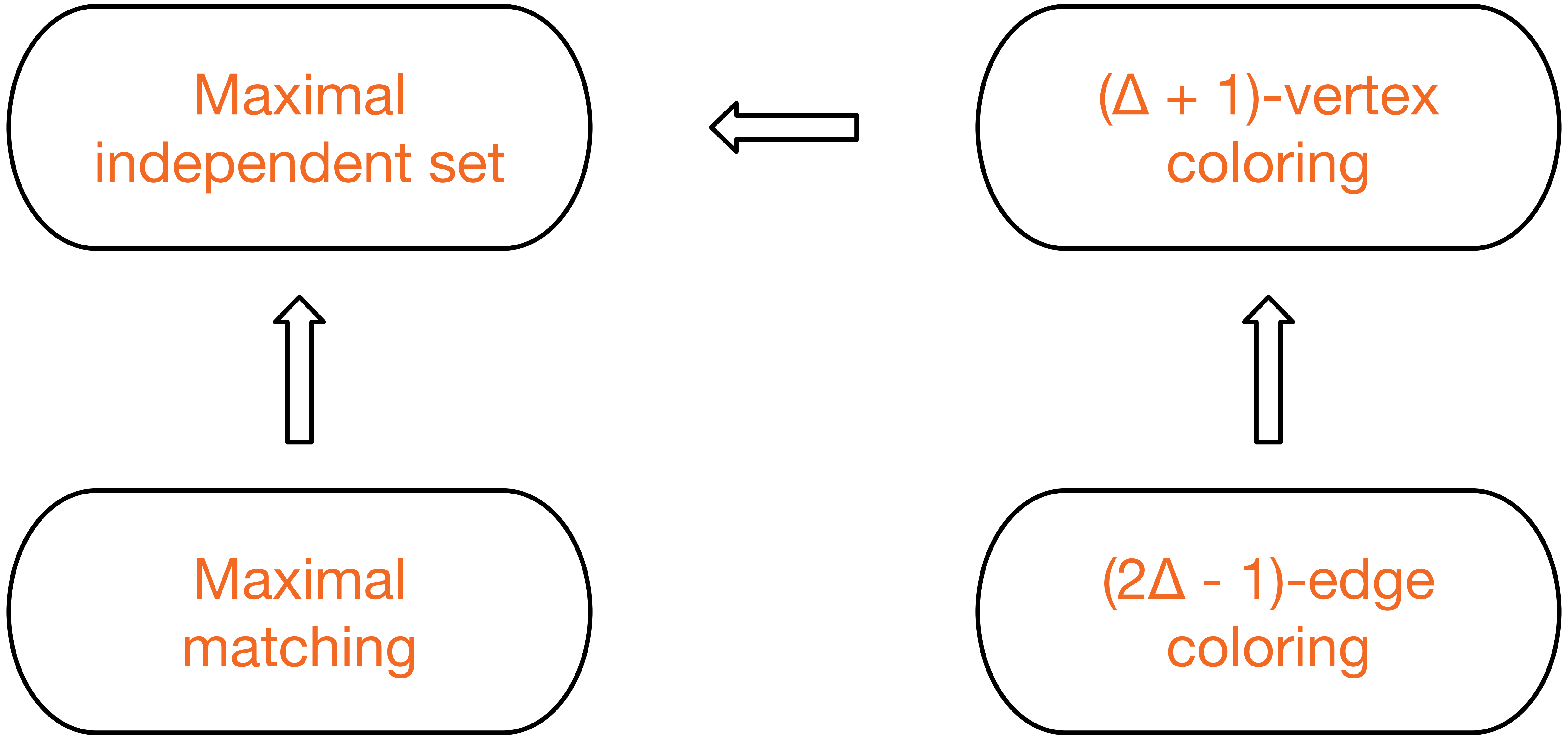
Four classical problems



These problems can be solved in $\text{poly} \log n$ rounds [Rozhon, Ghaffari '20]

These problems can be solved in $O(\log^2 \Delta \log n)$ rounds [Faour, Ghaffari, Grunau, Kuhn, Rozhon '23]

Four classical problems

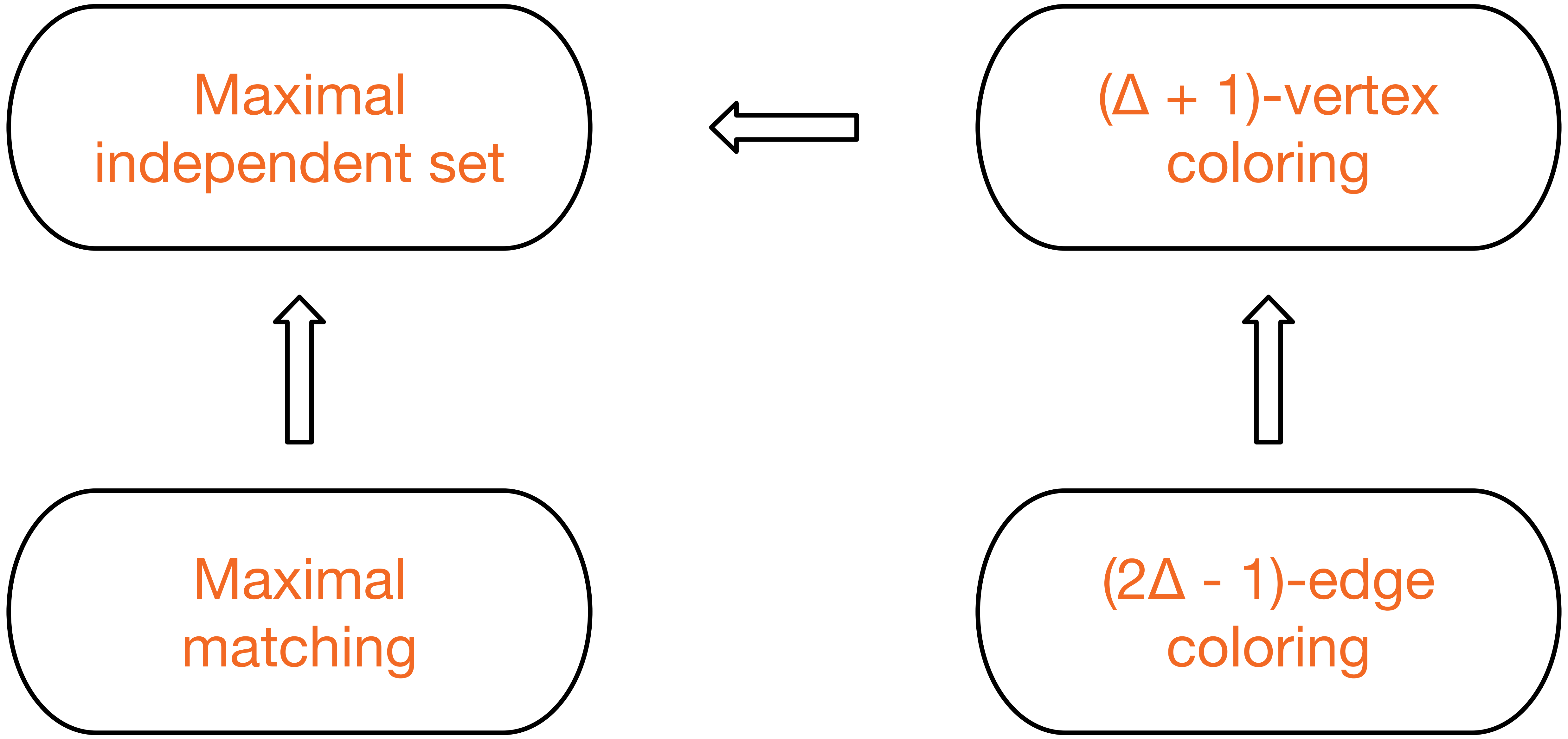


These problems can be solved in $\text{poly} \log n$ rounds [Rozhon, Ghaffari '20]

These problems can be solved in $O(\log^2 \Delta \log n)$ rounds [Faour, Ghaffari, Grunau, Kuhn, Rozhon '23]

These problems require $\Omega(\log^* n)$ rounds [Linial '87]

Four classical problems



These problems can be solved in $\mathrm{poly}\log n$ rounds [Rozhon, Ghaffari ’20]

These problems can be solved in $O(\log^2 \Delta \log n)$ rounds [Faour, Ghaffari, Grunau, Kuhn, Rozhon '23]

These problems require $\Omega(\log^* n)$ rounds [Linial ’87]

Big question: $f(\Delta) + O(\log^* n)$

Four classical problems

Maximal Matching	$O(\Delta + \log^* n)$ [Panconesi, Rizzi '01]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19]

Four classical problems

Maximal Matching	$O(\Delta + \log^* n)$ [Panconesi, Rizzi '01]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19]
Maximal Independent Set	$O(\Delta + \log^* n)$ [Barenboim, Elkin, Kuhn '09]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19] [BBKO '22]

Four classical problems

Maximal Matching	$O(\Delta + \log^* n)$ [Panconesi, Rizzi '01]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19]
Maximal Independent Set	$O(\Delta + \log^* n)$ [Barenboim, Elkin, Kuhn '09]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19] [BBKO '22]
$(\Delta + 1)$-Vertex Coloring	$O(\sqrt{\Delta \log \Delta} + \log^* n)$ [FHK '16] [BEG '18] [MT '20]	

Four classical problems

Maximal Matching	$O(\Delta + \log^* n)$ [Panconesi, Rizzi '01]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19]
Maximal Independent Set	$O(\Delta + \log^* n)$ [Barenboim, Elkin, Kuhn '09]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19] [BBKO '22]
$(\Delta + 1)$-Vertex Coloring	$O(\sqrt{\Delta \log \Delta} + \log^* n)$ [FHK '16] [BEG '18] [MT '20]	
$(2\Delta - 1)$-Edge Coloring	$(\log \Delta)^{O(\log \log \Delta)} + O(\log^* n)$ [Balliu, Kuhn, Olivetti '20]	

Four classical problems

Maximal Matching	$O(\Delta + \log^* n)$ [Panconesi, Rizzi '01]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19]
Maximal Independent Set	$O(\Delta + \log^* n)$ [Barenboim, Elkin, Kuhn '09]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19] [BBKO '22]
$(\Delta + 1)$-Vertex Coloring	$O(\sqrt{\Delta \log \Delta} + \log^* n)$ [FHK '16] [BEG '18] [MT '20]	
$(2\Delta - 1)$-Edge Coloring	$(\log \Delta)^{O(\log \log \Delta)} + O(\log^* n)$ [Balliu, Kuhn, Olivetti '20]	

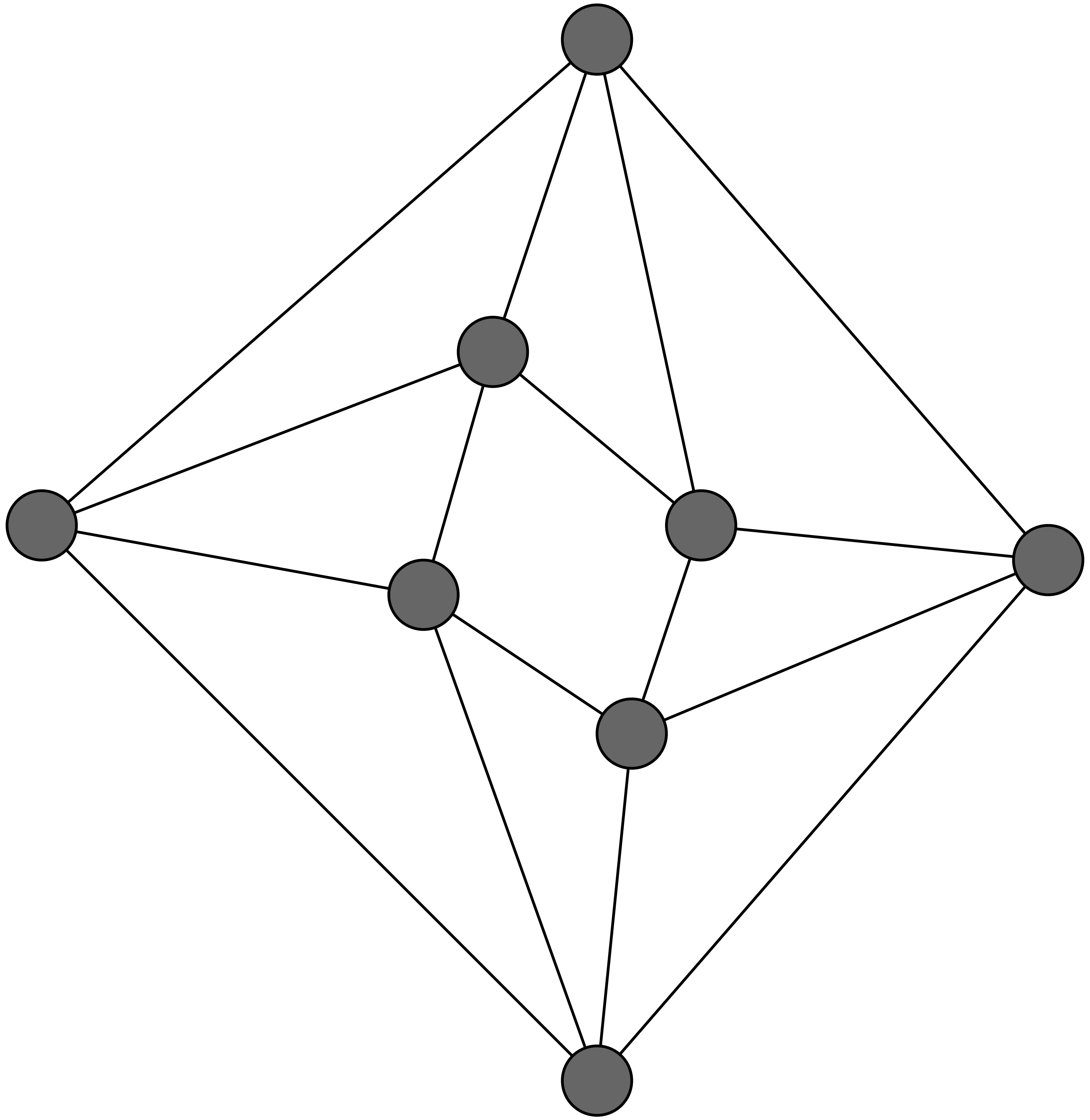
Our results

$(2\Delta - 1)$-Edge Coloring	$O(\text{poly} \log \Delta + \log^* n)$	LOCAL model
$O(\Delta)$-Edge Coloring	$O(\text{poly} \log \Delta + \log^* n)$	CONGEST model

Our results

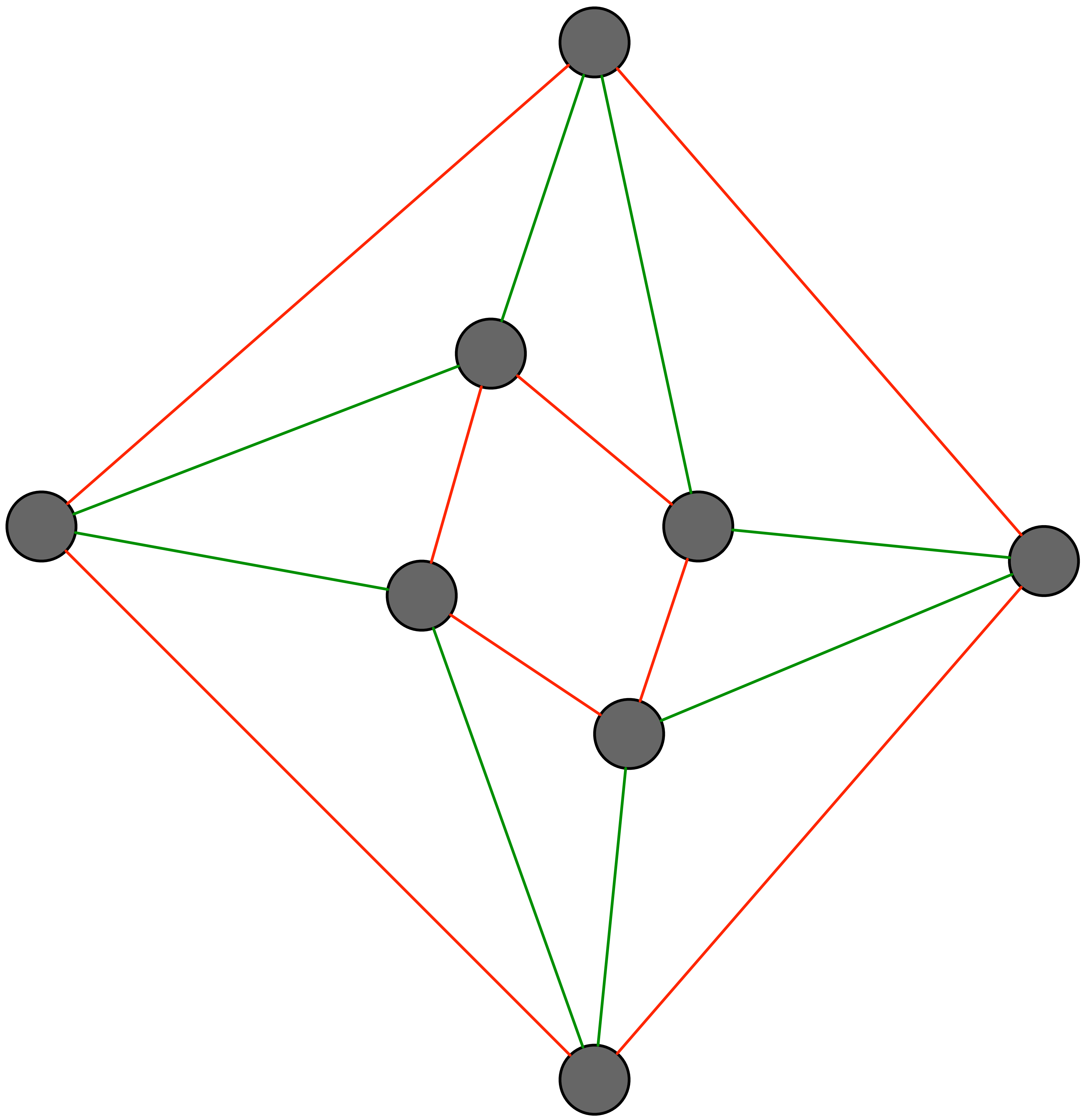
$(\text{degree} + 1)$-List Edge Coloring	$O(\log^7 C \cdot \log^5 \Delta + \log^* n)$	LOCAL model
$(2\Delta - 1)$-Edge Coloring	$O(\log^{12} \Delta + \log^* n)$	LOCAL model
$(8 + \varepsilon)\Delta$-Edge Coloring	$O\left(\frac{\log^{12} \Delta}{\varepsilon^6} + \log^* n\right)$	CONGEST model

A possible approach

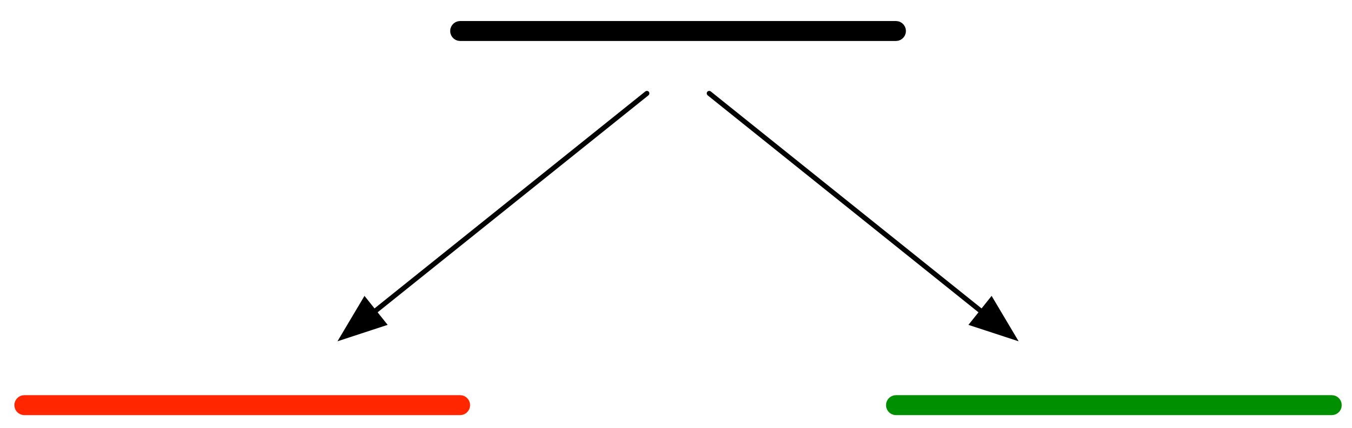


Try to **recursively color** the edges with **2 colors**, such that each node has roughly the **same amount of incident edges for each color**

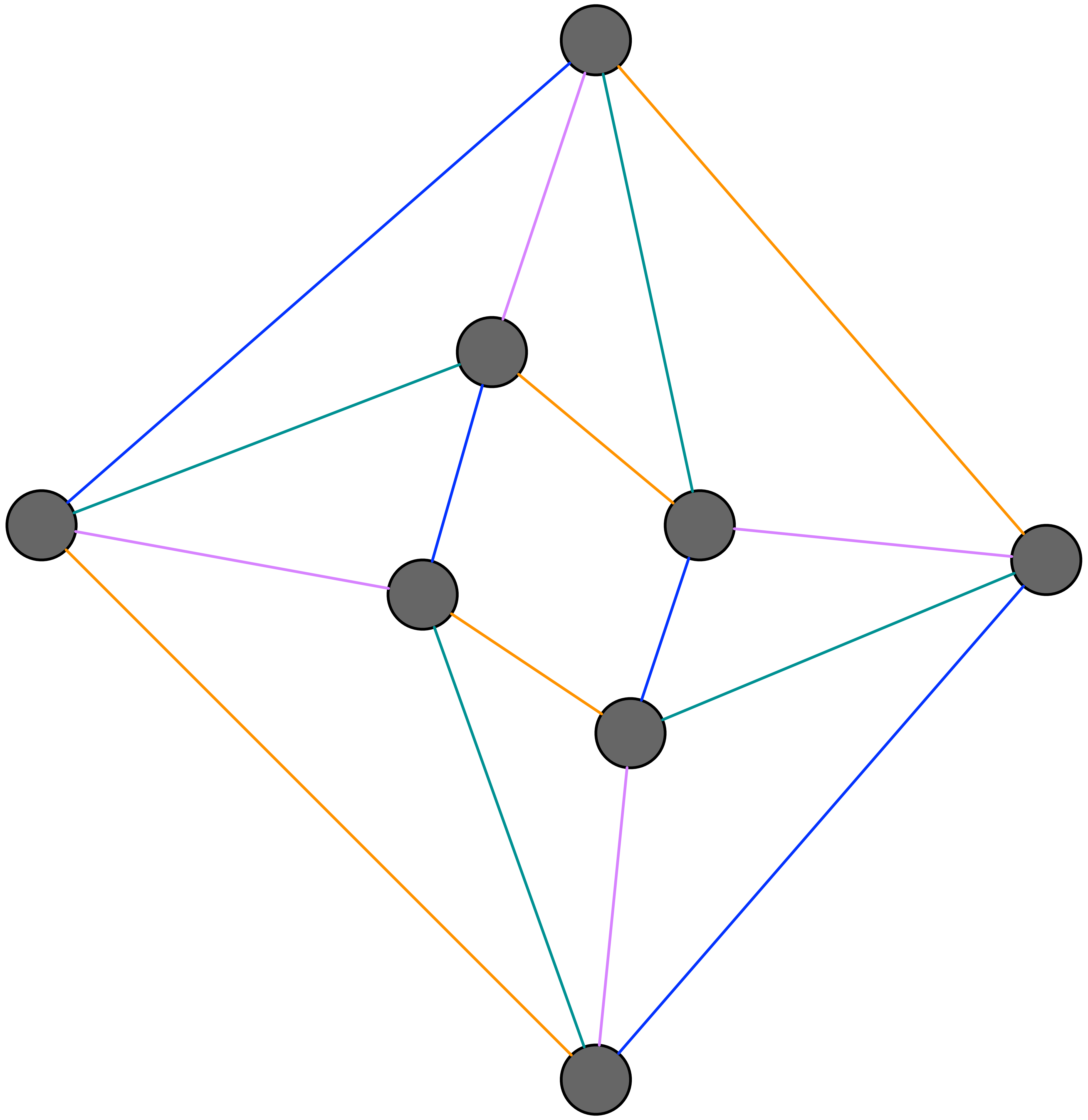
A possible approach



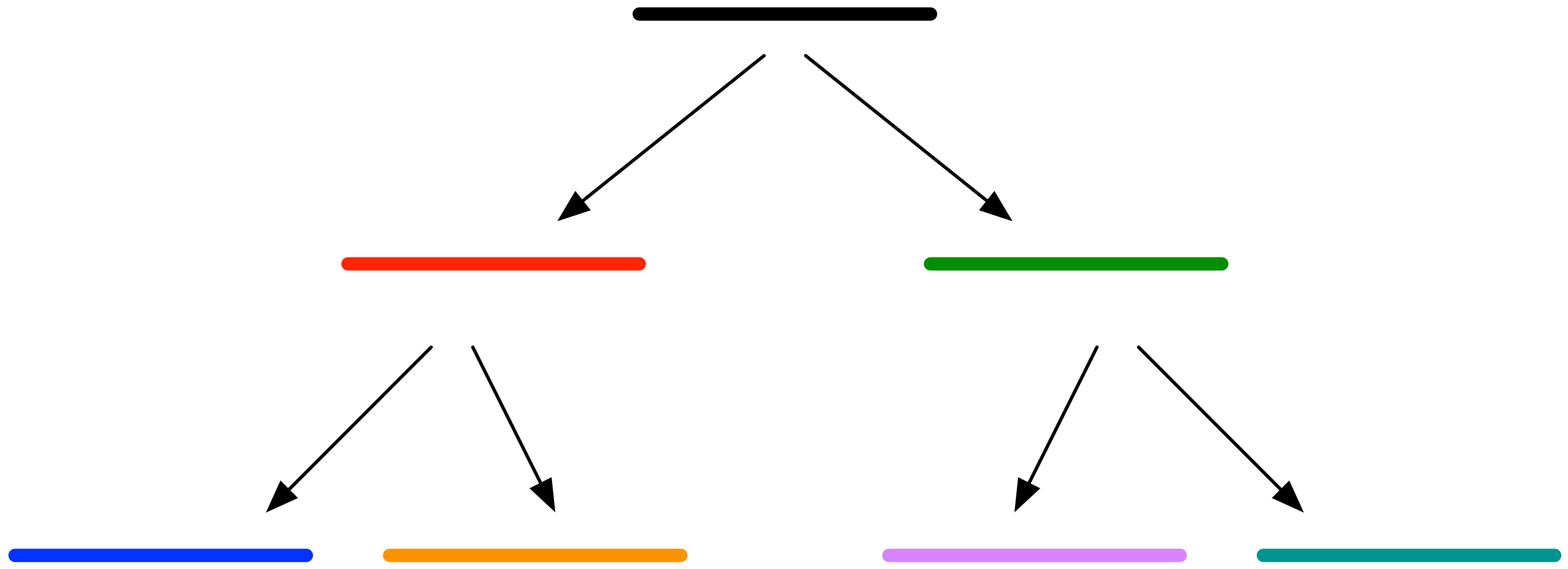
Try to **recursively color** the edges with **2 colors**, such that each node has roughly the **same amount of incident edges for each color**



A possible approach



Try to **recursively color** the edges with **2 colors**, such that each node has roughly the **same amount of incident edges for each color**



A possible approach

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly $\Delta/2$, Recurse** on each subgraph.

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.

- After $\mathbf{T}$ steps of recursion the maximum degree is $\boldsymbol{\Delta} \cdot \left(\frac{\mathbf{1}+\varepsilon}{\mathbf{2}}\right)^{\mathbf{T}}$

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.
- After $\mathbf{T}$ steps of recursion the maximum degree is $\mathbf{\Delta} \cdot \left(\frac{\mathbf{1}+\varepsilon}{\mathbf{2}}\right)^{\mathbf{T}}$
- Let us fix $\mathbf{T} = \log \mathbf{\Delta}$ and $\varepsilon = \mathbf{1}/\log \mathbf{\Delta}$

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.
- After $\mathbf{T}$ steps of recursion the maximum degree is $\boldsymbol{\Delta} \cdot \left(\frac{\mathbf{1}+\varepsilon}{\mathbf{2}}\right)^{\mathbf{T}}$
- Let us fix $\mathbf{T} = \log \boldsymbol{\Delta}$ and $\varepsilon = \mathbf{1}/\log \boldsymbol{\Delta}$
- After $\mathbf{T}$ steps each subgraph has **constant maximum degree**

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.
- After $\mathbf{T}$ steps of recursion the maximum degree is $\mathbf{\Delta} \cdot \left(\frac{\mathbf{1} + \varepsilon}{\mathbf{2}}\right)^{\mathbf{T}}$
- Let us fix $\mathbf{T} = \log \mathbf{\Delta}$ and $\varepsilon = \mathbf{1}/\log \mathbf{\Delta}$
- After $\mathbf{T}$ steps each subgraph has **constant maximum degree**
- **The number of colors is** $2^T \cdot O(1) = O(\Delta)$

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.
- After $\mathbf{T}$ steps of recursion the maximum degree is $\boldsymbol{\Delta} \cdot \left(\frac{\mathbf{1}+\varepsilon}{\mathbf{2}}\right)^{\mathbf{T}}$
- Let us fix $\mathbf{T} = \log \boldsymbol{\Delta}$ and $\varepsilon = \mathbf{1}/\log \boldsymbol{\Delta}$
- After $\mathbf{T}$ steps each subgraph has **constant maximum degree**
- **The number of colors is** $2^T \cdot O(1) = O(\Delta)$
- **Running time:** $\log \Delta \cdot T_{balanced\_2\_col} + T_{\text{final}}$

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.

- After $\mathbf{T}$ steps of recursion the maximum degree is $\boldsymbol{\Delta} \cdot \left(\frac{\mathbf{1}+\varepsilon}{\mathbf{2}}\right)^{\mathbf{T}}$

- Let us fix $\mathbf{T} = \log \boldsymbol{\Delta}$ and $\varepsilon = \mathbf{1}/\log \boldsymbol{\Delta}$
- After $\mathbf{T}$ steps each subgraph has **constant maximum degree**
- **The number of colors is** $2^T \cdot O(1) = O(\Delta)$
- **Running time:** $\log \Delta \cdot T_{balanced\_2\_col} + T_{\text{final}}$

Can be done in just $O(\log^* n)$

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.
- After $\mathbf{T}$ steps of recursion the maximum degree is $\mathbf{\Delta} \cdot \left(\frac{\mathbf{1}+\varepsilon}{\mathbf{2}}\right)^{\mathbf{T}}$
- Let us fix $\mathbf{T} = \log \mathbf{\Delta}$ and $\varepsilon = \mathbf{1}/\log \mathbf{\Delta}$
- After $\mathbf{T}$ steps each subgraph has **constant maximum degree**
- **The number of colors is** $2^T \cdot O(1) = O(\Delta)$
- **Running time:** $\log \Delta \cdot T_{balanced\_2\_col} + T_{\text{final}}$

This requires too much!

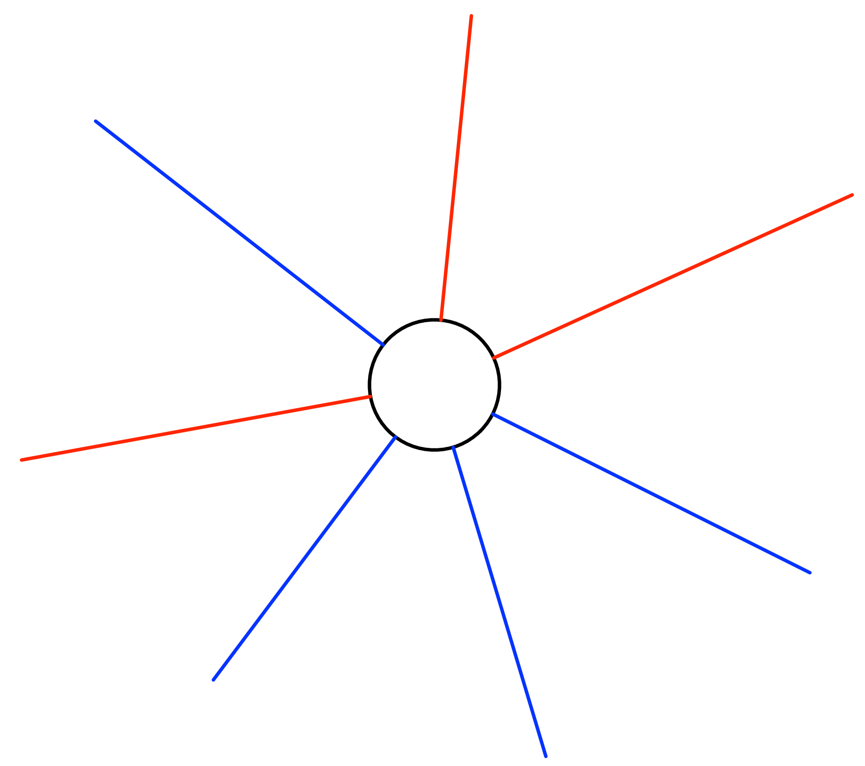
Can be done in just $O(\log^* n)$

A possible approach: the issue

- **d**-**node**-defective **2**-edge-coloring: **color the edges** with **2** colors such that **each node** has at most **d** incident **edges of the same color**.

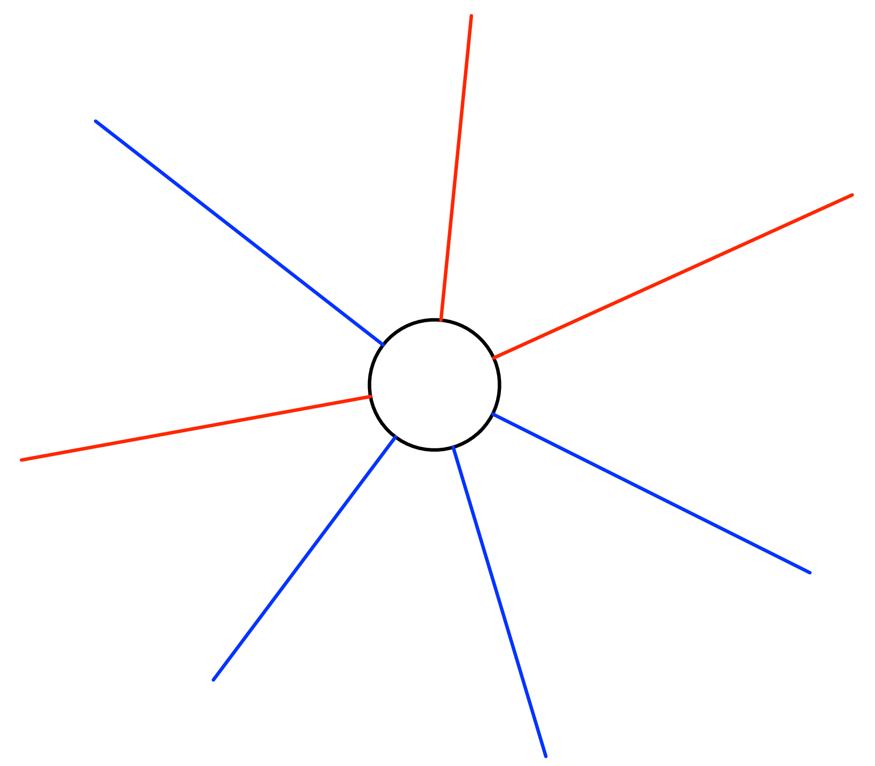
  This is **hard,** even for $d \leq \Delta - 1.$

  It requires $\Omega(\log n)$ rounds!



A different subroutine

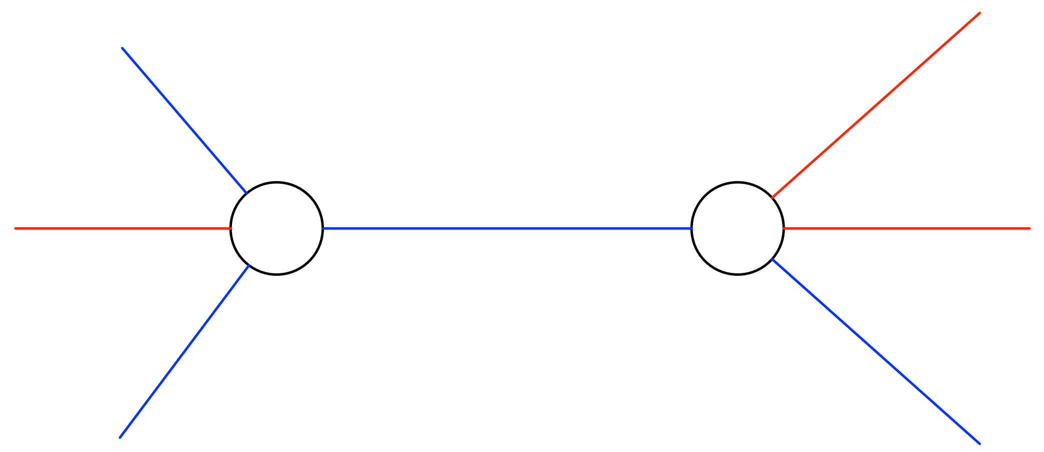
- **d-node**-defective **2**-edge-coloring: **color the edges** with **2** colors such that **each node** has at most **d** incident **edges of the same color**.



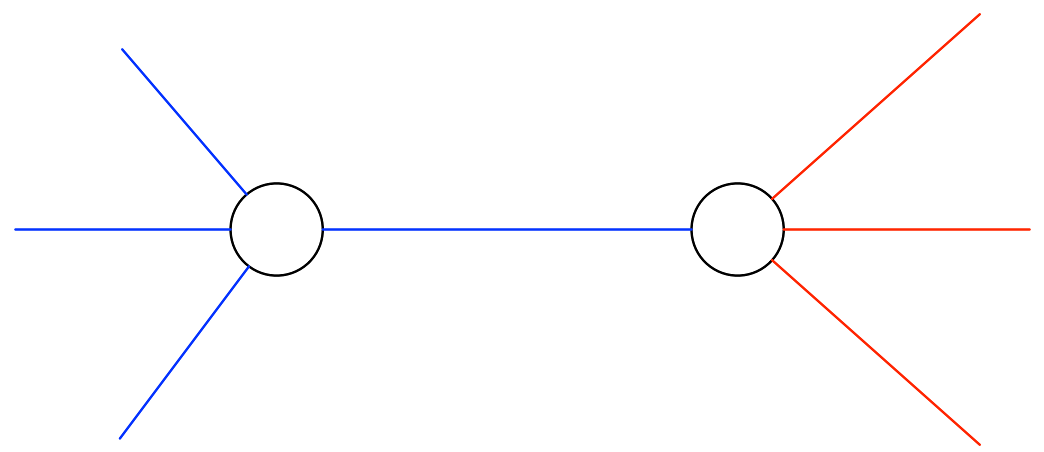
This is **hard,** even for $d \leq \Delta - 1.$

It requires $\Omega(\log n)$ rounds!

- **d-edge**-defective **2**-edge-coloring: **color the edges** with **2** colors such that **each edge** has at most **d** incident **edges of its color**.



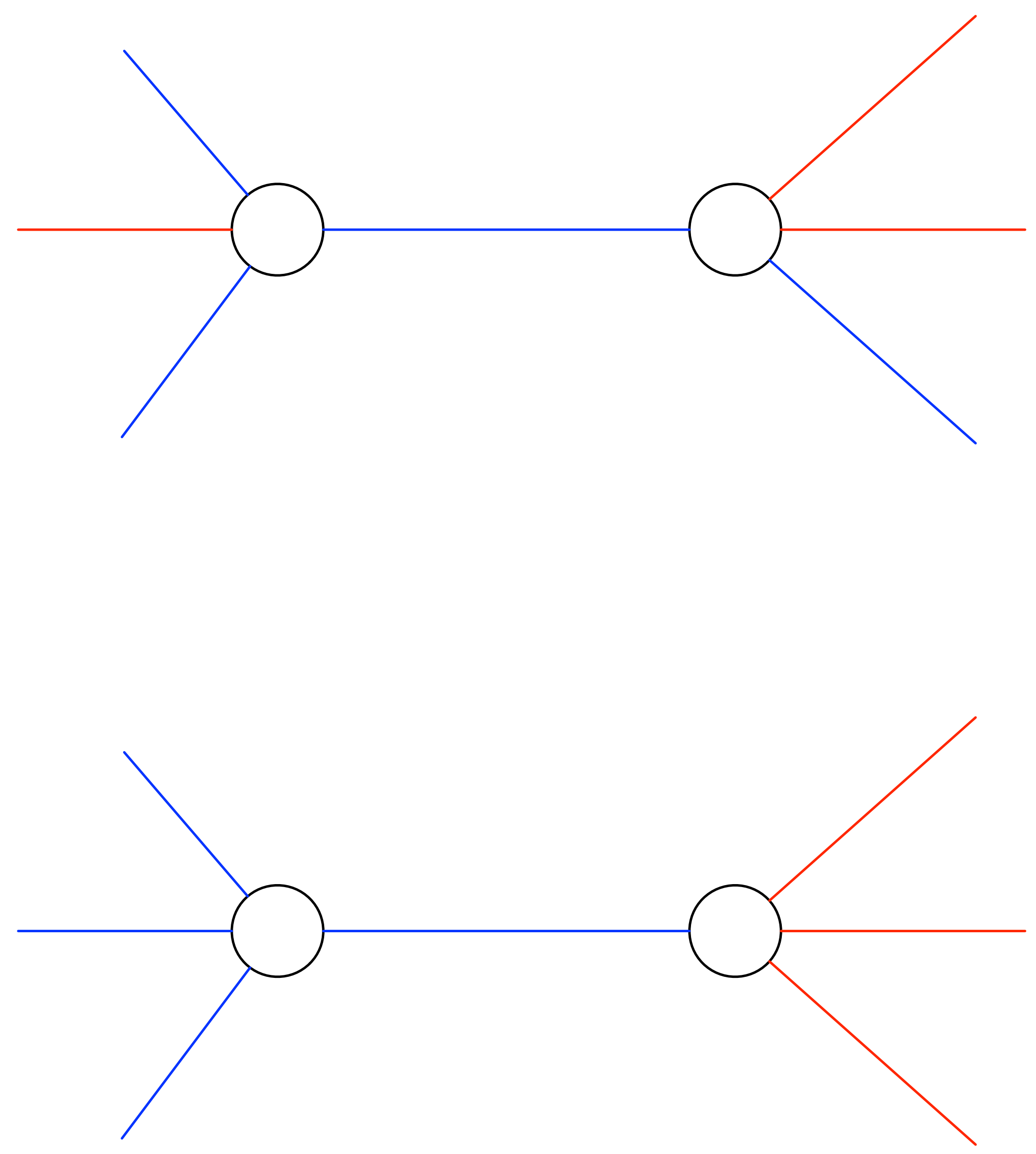
This is the problem that we tried to solve, for $d \leq (1 + \varepsilon)\Delta$



Main Ingredient

d-edge-defective **2**-edge-coloring:

> **color the edges** with **2** colors such that **each edge** has at most **d** incident **edges of its color**.

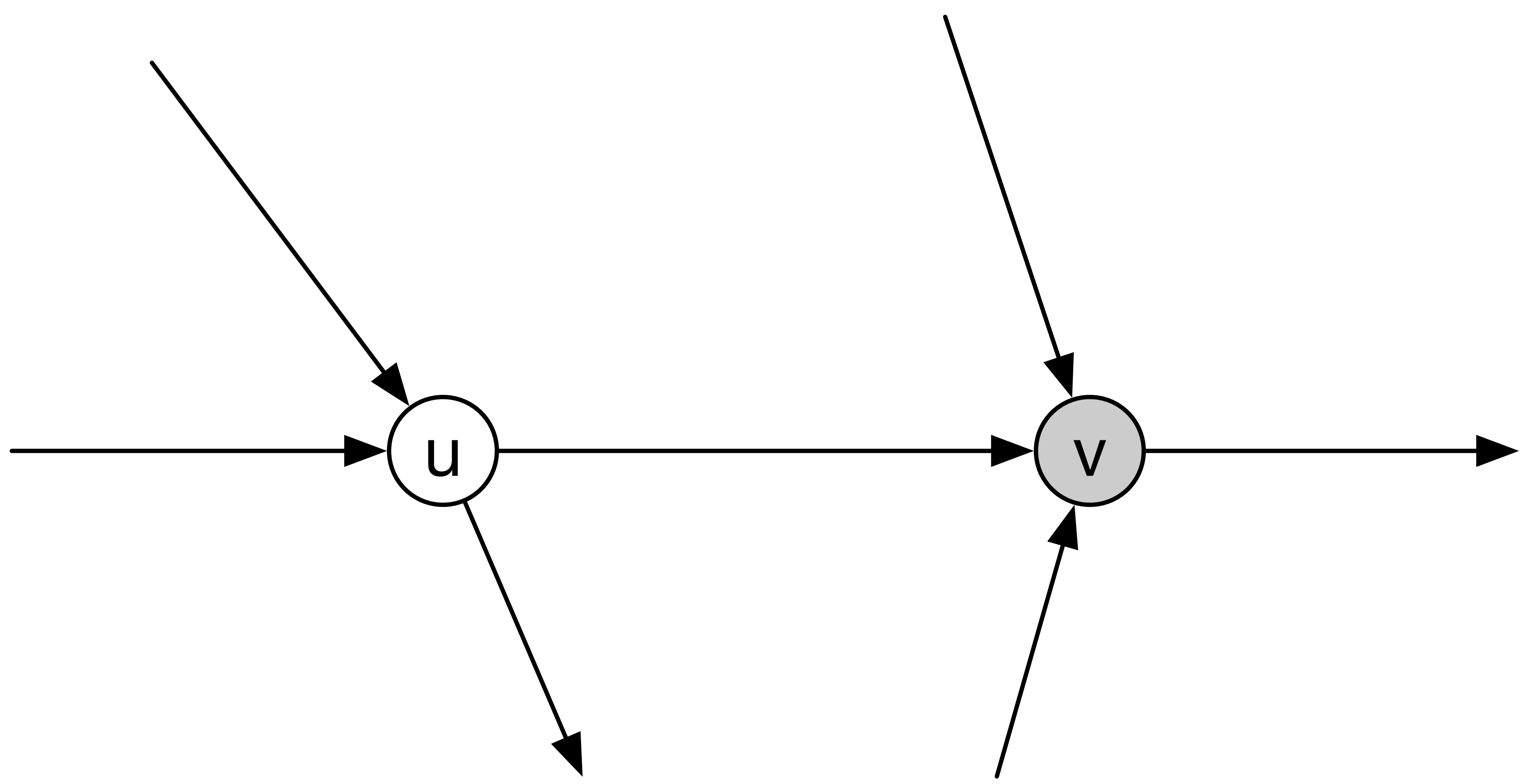


For $d \leq (1+\varepsilon)\Delta$, the problem can be solved in $O(\text{poly}(1/\epsilon, \log \Delta))$ time!

(for list coloring, we need a bit more)

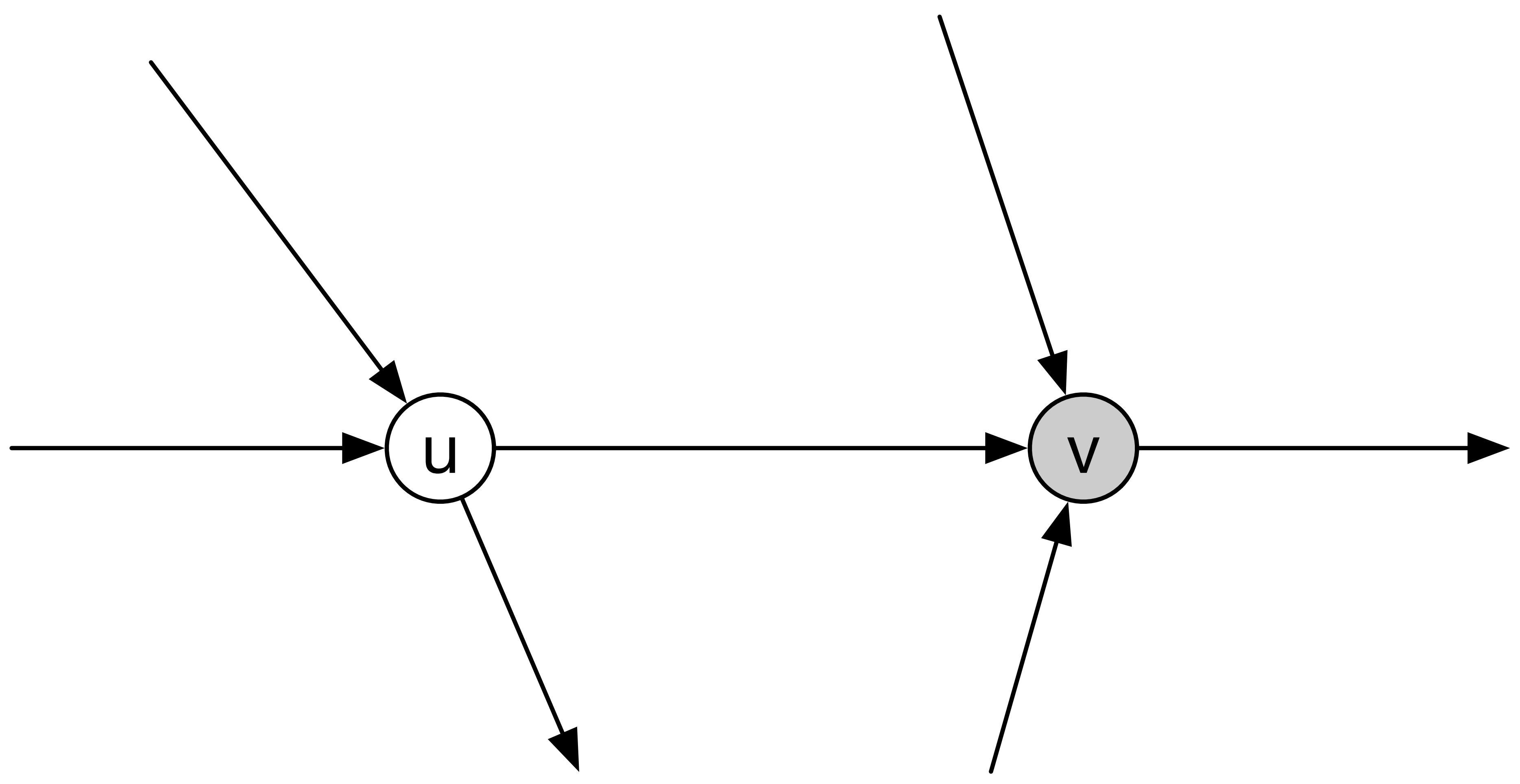
Stable Orientation

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + 1$



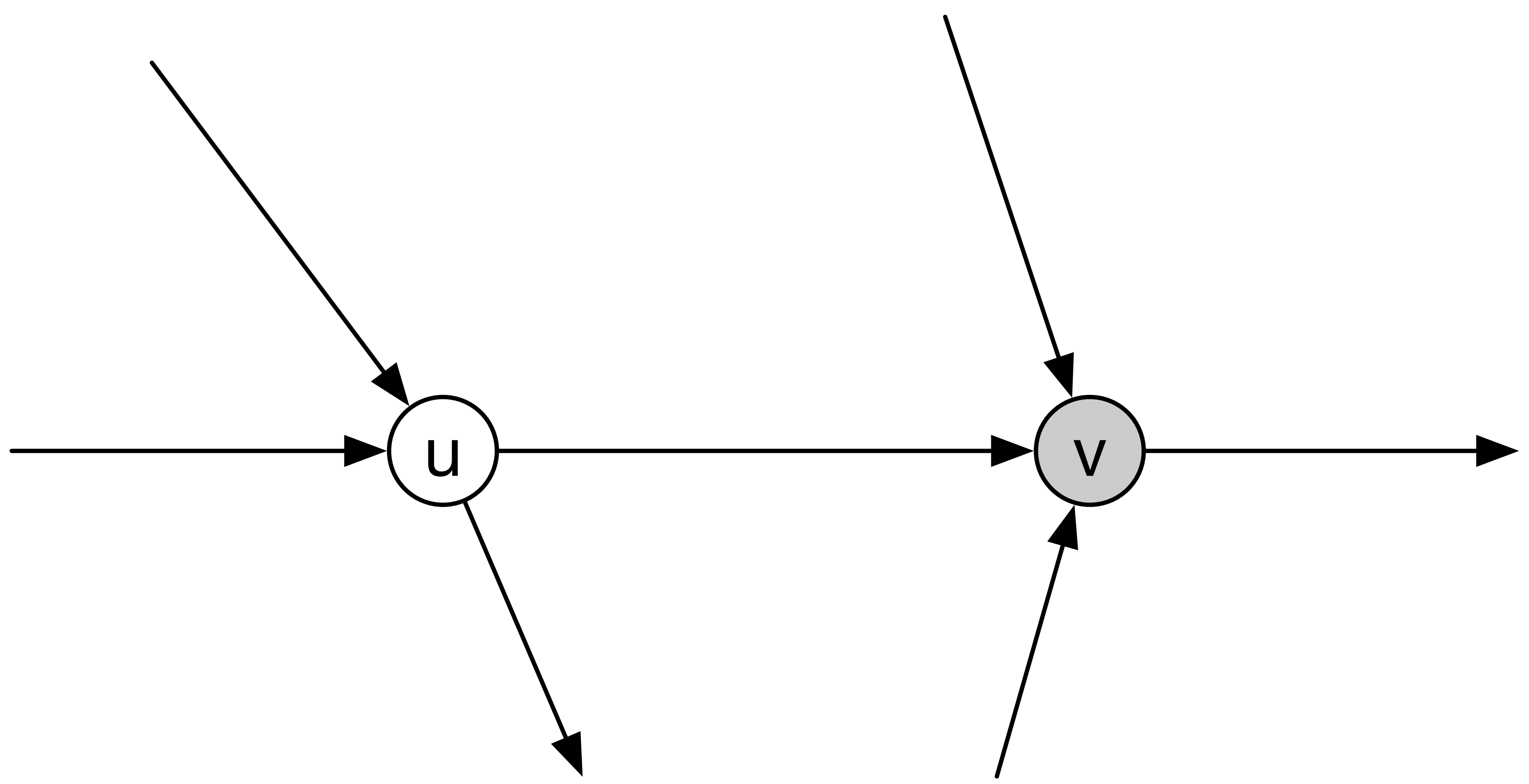
Stable Orientation

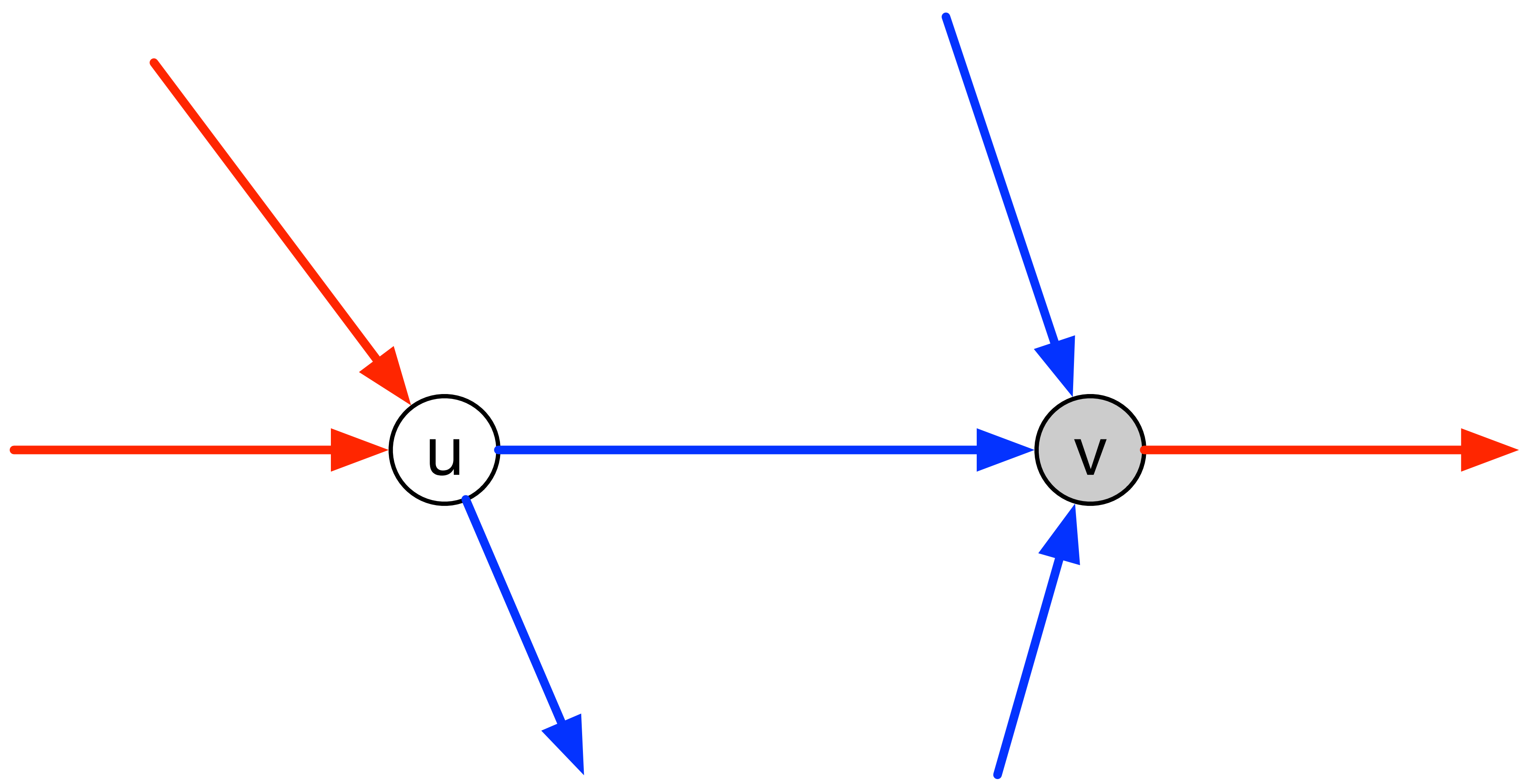
- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + 1$



Stable Orientation

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + 1$





Stable Orientation

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\mathrm{in}}(v) \leq \deg_{\mathrm{in}}(u) + 1$

Efficient Load-Balancing through Distributed Token Dropping

[Brandt, Keller, Rybicki, Suomela, Uitto 2021]

This problem can be solved in $O(\Delta^4)$ rounds!

Issues

- **Stable orientation** solves **"balanced" edge 2-coloring**, but:
  - The running time is $O(\Delta^4)$, we want $O(\log^c \Delta)$
  - We can turn a stable orientation into a edge 2-coloring **only if a 2-vertex coloring is given**, **we do not have that**
  - The conversion only works on **regular graphs, we do not have that**
  - The recursion schema solves $O(\Delta)$**-edge coloring,** not $(2\Delta - 1)$ **-edge coloring.** For a better result, we need to solve a harder variant **(list coloring)**

Relaxed Stable Orientation

Relaxed Stable Orientation

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + k$

Relaxed Stable Orientation

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + k$
- This problem can be solved in $O(\Delta^5/k^5)$ rounds!

Relaxed Stable Orientation

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + k$
- This problem can be solved in $O(\Delta^5/k^5)$ rounds!
- For $k = \dfrac{\Delta}{\log \Delta}$, this gives an $O(\log^5 \Delta)$ round algorithm!

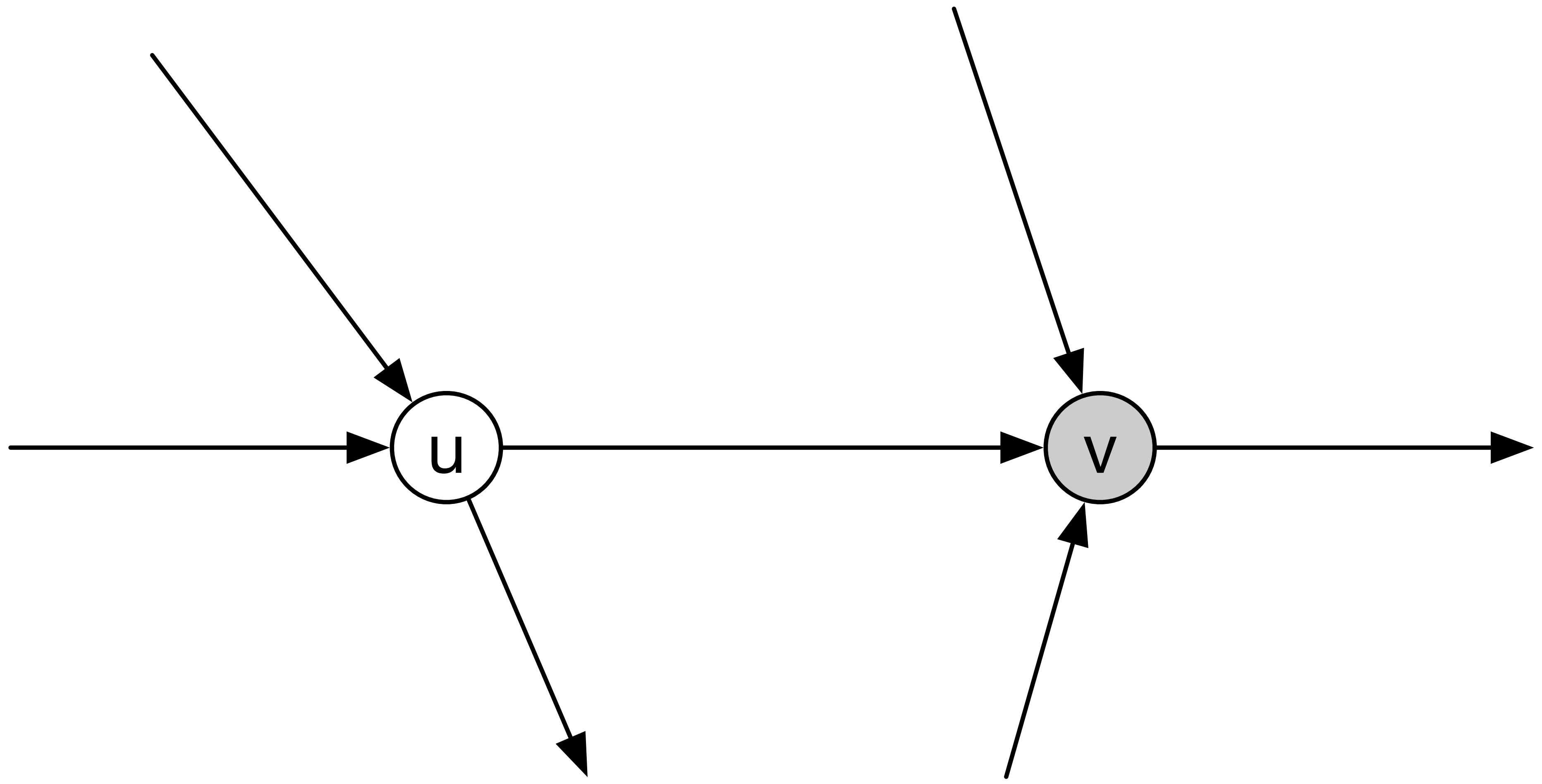
Open questions: edge coloring

- We can solve $(2\Delta - 1)$-edge coloring in $O(\log^{12} \Delta + \log^* n)$ rounds
  - Can we improve the exponent? We know a faster algorithm, but only for $O(\Delta)$-edge coloring
- Can we solve vertex coloring in subpoly(Δ)?
- Can we prove a non-trivial lower bound for solving (2Δ - 1)-edge coloring?
  - Can we show that it cannot be solved in o(log Δ) + O(log* n)?

Open questions

Stable Orientation: **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + 1$

- This problem can be solved in $O(\Delta^4)$ rounds
  [Brandt, Keller, Rybicki, Suomela, Uitto 2021]
- Can we do better?



Thank you!