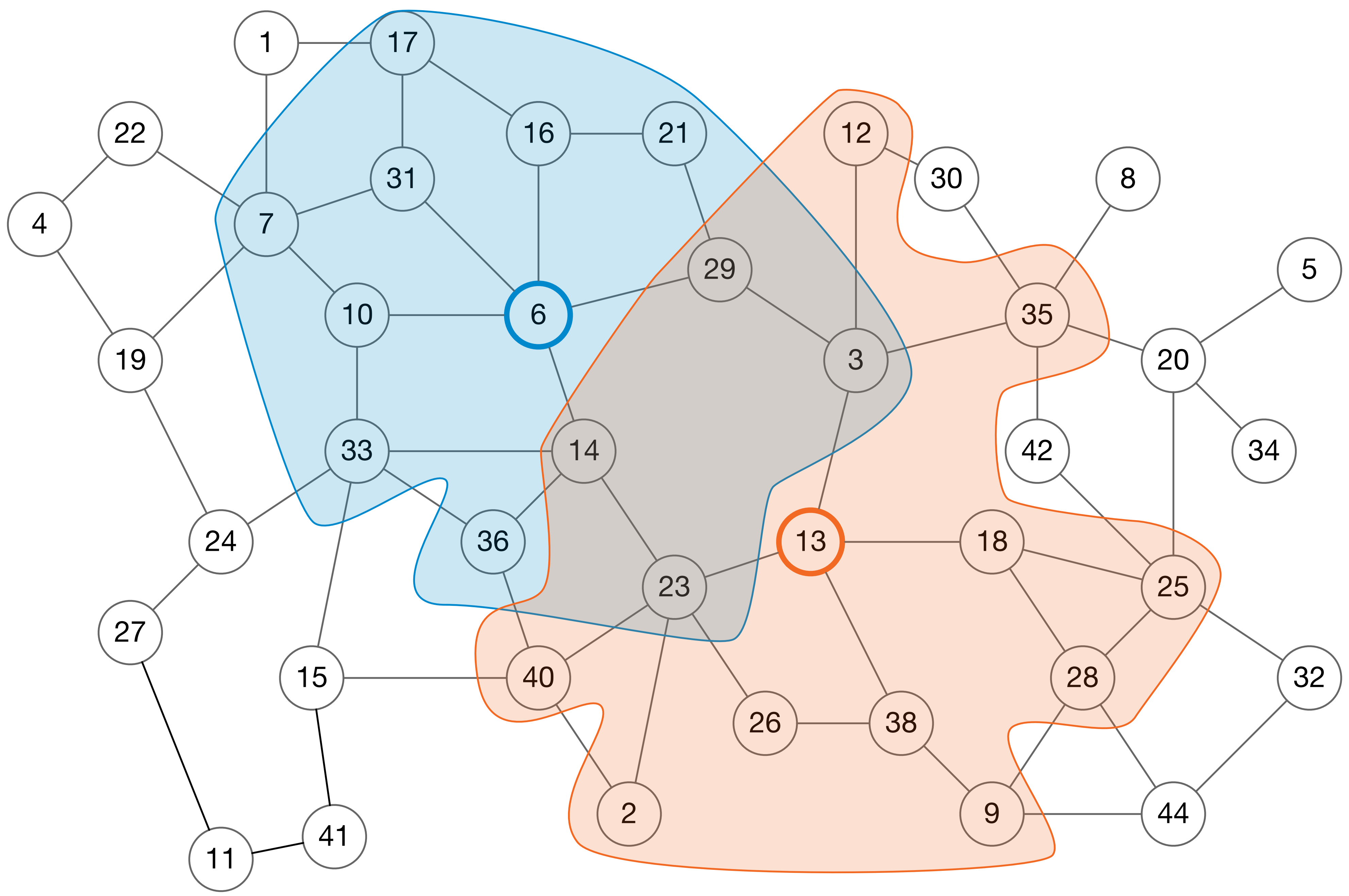
Distributed Edge Coloring in Time Quasi-Polylogarithmic in Delta

Alkida Balliu, Fabian Kuhn, **Dennis Olivetti**
University of Freiburg, Germany

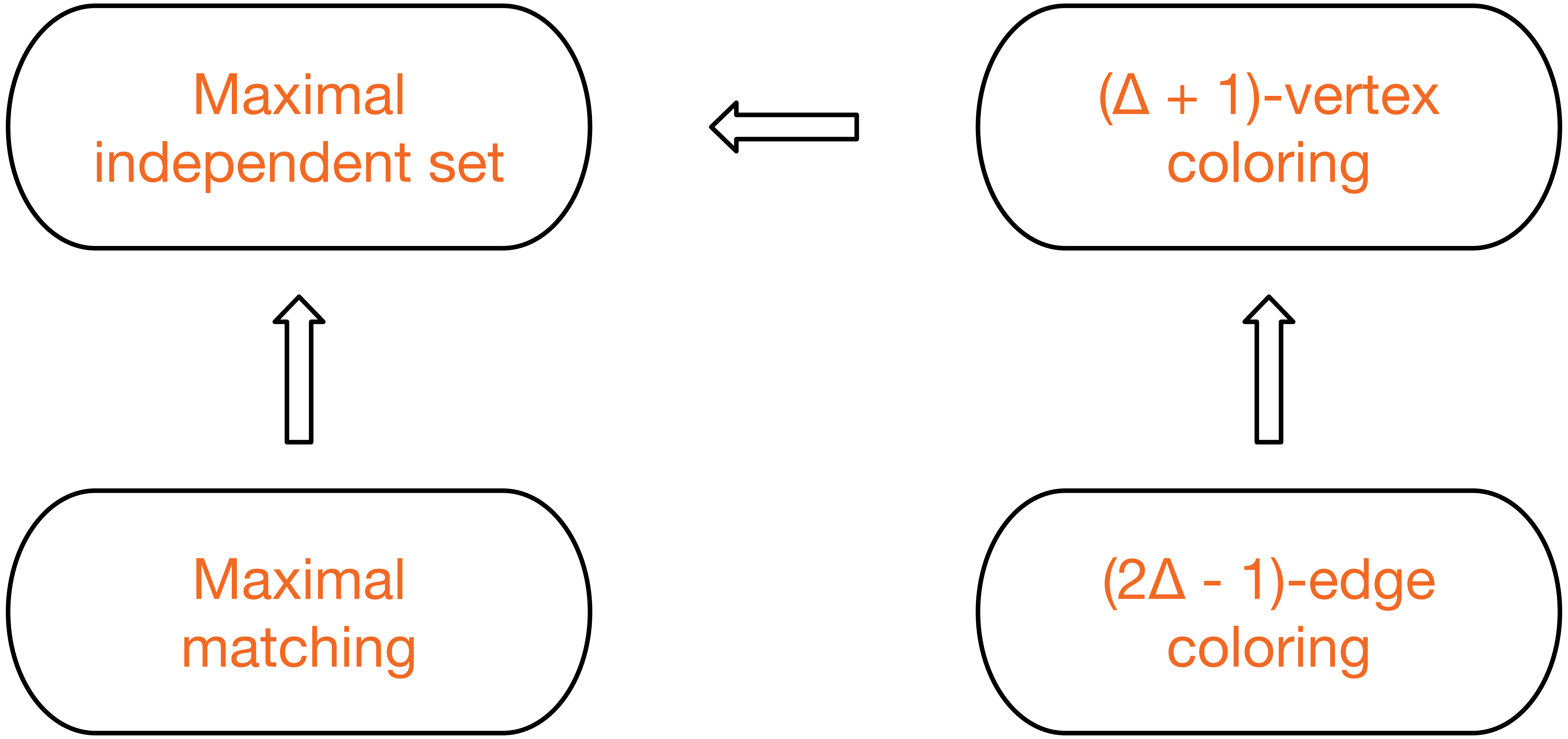
LOCAL model

- 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
- Focus on locality: time = number of rounds = distance



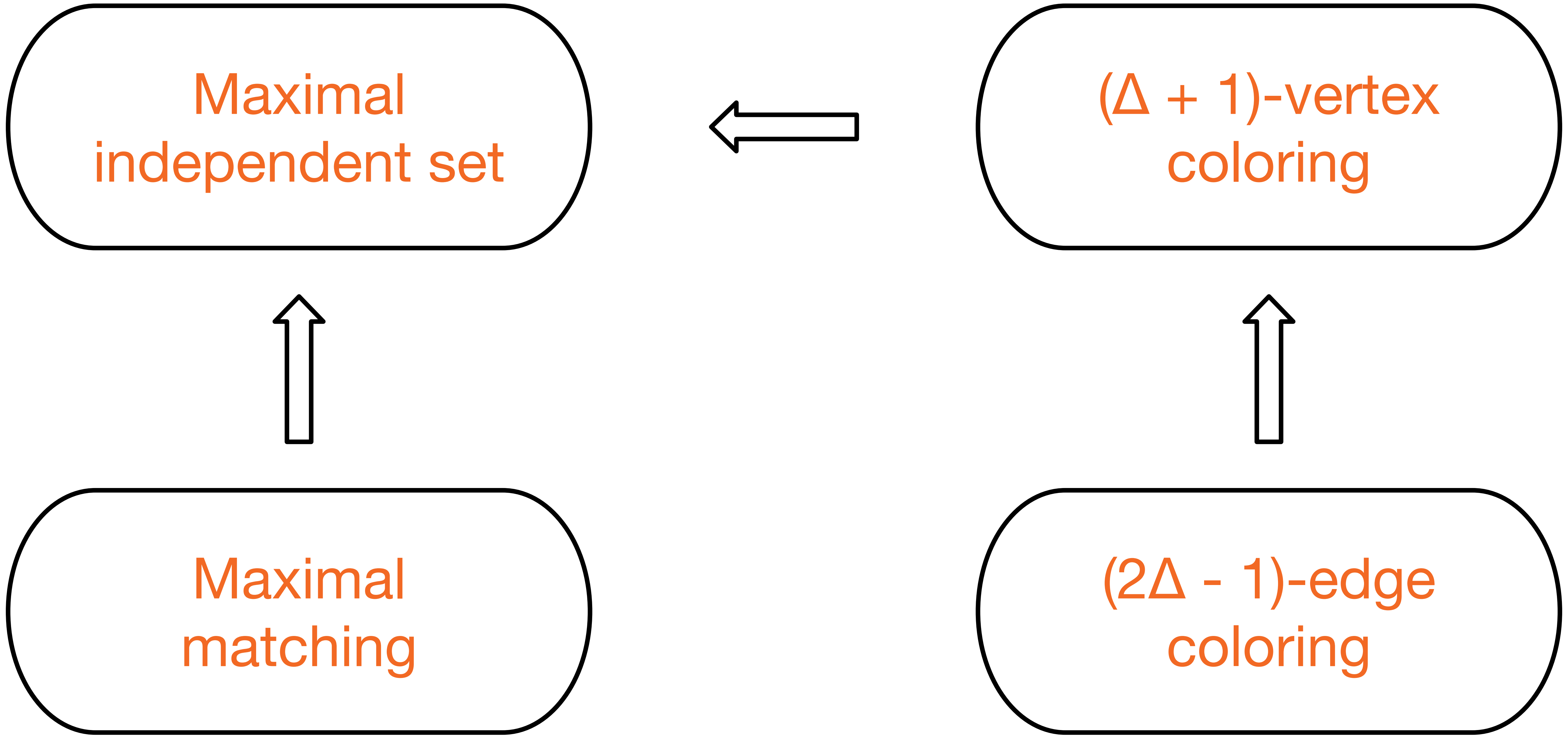
LOCAL model: symmetry breaking

Four classical problems



Four classical problems

These problems can be solved in poly(log n) rounds [Rozhon, Ghaffari '20]



But, how local are these problems?

Four classical problems: locality

Maximal independent set	$O(\Delta + \log^* n)$ [Barenboim, Elkin, Kuhn '09]	$\Omega(\min\{\Delta, \log n / \log\log n\})$ [Balliu et al. '19]	$\Omega(\log^* n)$ [Linial '87]

Four classical problems: locality

Maximal independent set	$O(\Delta + \log^* n)$ [Barenboim, Elkin, Kuhn '09]	$\Omega(\min\{\Delta, \log n / \log\log n\})$ [Balliu et al. '19]	$\Omega(\log^* n)$ [Linial '87]
Maximal matching	$O(\Delta + \log^* n)$ [Panconesi, Rizzi '01]	$\Omega(\min\{\Delta, \log n / \log\log n\})$ [Balliu et al. '19]	$\Omega(\log^* n)$ [Linial '87]

Four classical problems: locality

Maximal independent set	$O(\Delta + \log^* n)$ [Barenboim, Elkin, Kuhn '09]	$\Omega(\min\{\Delta, \log n / \log\log n\})$ [Balliu et al. '19]	$\Omega(\log^* n)$ [Linial '87]
Maximal matching	$O(\Delta + \log^* n)$ [Panconesi, Rizzi '01]	$\Omega(\min\{\Delta, \log n / \log\log n\})$ [Balliu et al. '19]	$\Omega(\log^* n)$ [Linial '87]
(Δ + 1)-vertex coloring	$\widetilde{O}(\sqrt{\Delta} + \log^* n)$ [FHK '16][BEG '18][MT '20]		$\Omega(\log^* n)$ [Linial '87]

Four classical problems: locality

Maximal independent set	$O(\Delta + \log^* n)$ [Barenboim, Elkin, Kuhn '09]	$\Omega(\min\{\Delta, \log n / \log\log n\})$ [Balliu et al. '19]	$\Omega(\log^* n)$ [Linial '87]
Maximal matching	$O(\Delta + \log^* n)$ [Panconesi, Rizzi '01]	$\Omega(\min\{\Delta, \log n / \log\log n\})$ [Balliu et al. '19]	$\Omega(\log^* n)$ [Linial '87]
(Δ + 1)-vertex coloring	$\widetilde{O}(\sqrt{\Delta} + \log^* n)$ [FHK '16][BEG '18][MT '20]		$\Omega(\log^* n)$ [Linial '87]
(2Δ - 1)-edge coloring	$2^{O(\sqrt{\log \Delta})} + O(\log^* n)$ [Kuhn '20]		$\Omega(\log^* n)$ [Linial '87]

Four classical problems: locality

Maximal independent set	$O(\Delta + \log^* n)$ [Barenboim, Elkin, Kuhn '09]	$\Omega(\min\{\Delta, \log n / \log\log n\})$ [Balliu et al. '19]	$\Omega(\log^* n)$ [Linial '87]
Maximal matching	$O(\Delta + \log^* n)$ [Panconesi, Rizzi '01]	$\Omega(\min\{\Delta, \log n / \log\log n\})$ [Balliu et al. '19]	$\Omega(\log^* n)$ [Linial '87]
(Δ + 1)-vertex coloring	$\widetilde{O}(\sqrt{\Delta} + \log^* n)$ [FHK '16][BEG '18][MT '20]		$\Omega(\log^* n)$ [Linial '87]
(2Δ - 1)-edge coloring	$2^{O(\sqrt{\log \Delta})} + O(\log^* n)$ [Kuhn '20]		$\Omega(\log^* n)$ [Linial '87]

Edge coloring: state of the art

- $(2\Delta - 1)$-edge coloring (achieved through $(\Delta + 1)$-vertex coloring):
  - $O(\Delta + \log^* n)$ [Barenboim, Elkin '09], [Kuhn '09]
  - $O(\Delta^{3/4} + \log^* n)$ [Barenboim '15]
  - $\tilde{O}(\sqrt{\Delta} + \log^* n)$ [Fraigniaud, Heinrich, Kosowski '16] [Barenboim, Elkin, Goldenberg '18] [Maus, Tonoyan '20]
- $O(\Delta)$-edge coloring: $O(\Delta^{\varepsilon} + \log^* n)$ [Barenboim, Elkin '10]
- $(2\Delta - 1)$-edge coloring in $2^{O(\sqrt{\log \Delta})} + O(\log^* n)$ [Kuhn '20]

Our result

(2Δ - 1)-edge coloring

$$2^{O(\log^2 \log \Delta)} + O(\log^* n)$$

[this paper]

$$\Omega(\log^* n)$$

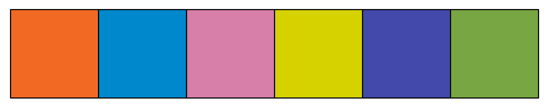
[Linial '87]

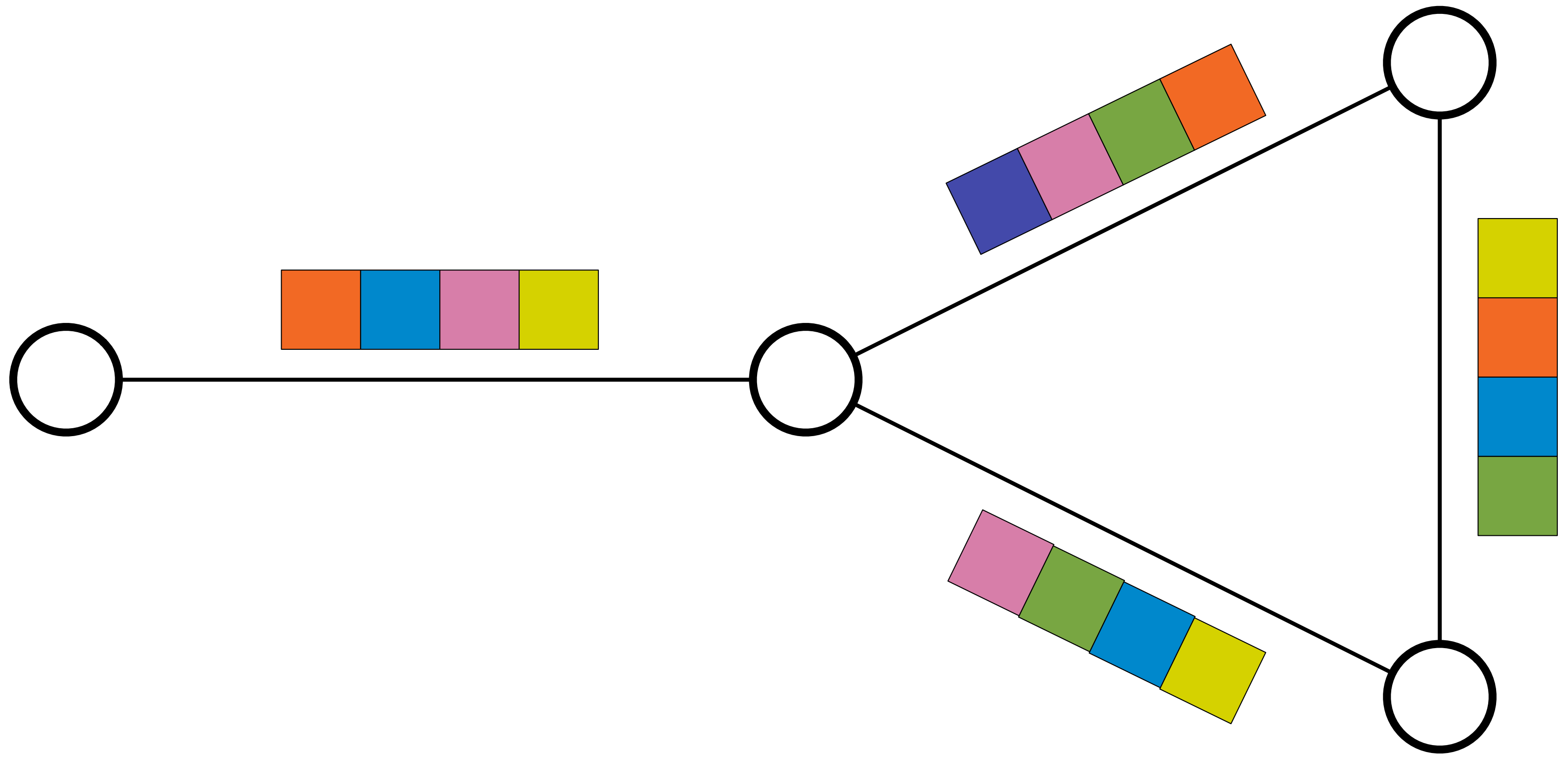
Our result

(deg(e) + 1)-list edge coloring can be solved in time quasi-polylogarithmic in Δ

(2Δ - 1)-edge coloring	$2^{O(\log^2 \log \Delta)} + O(\log^* n)$ [this paper]	$\Omega(\log^* n)$ [Linial '87]

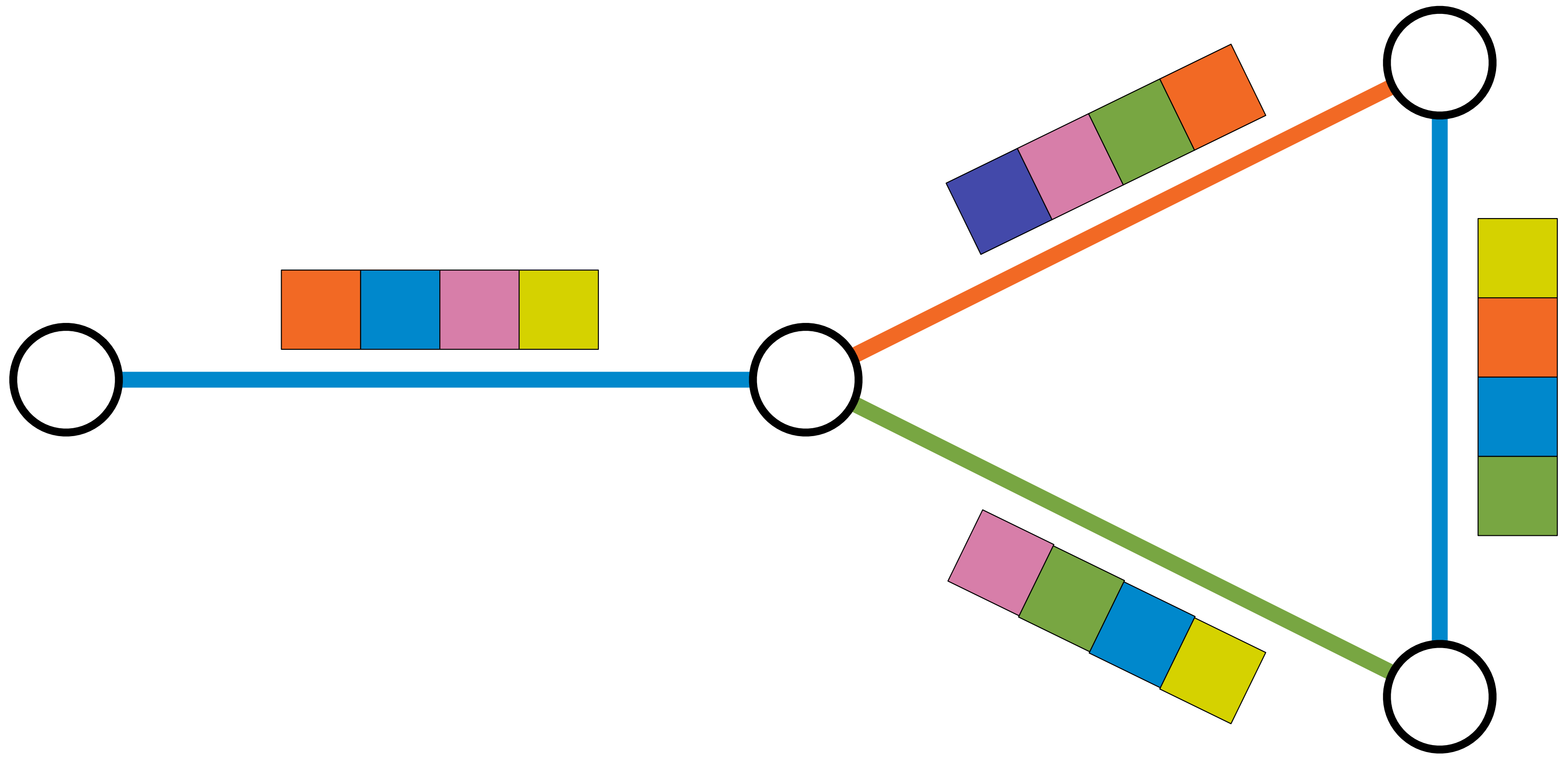
List edge coloring

Color palette: 



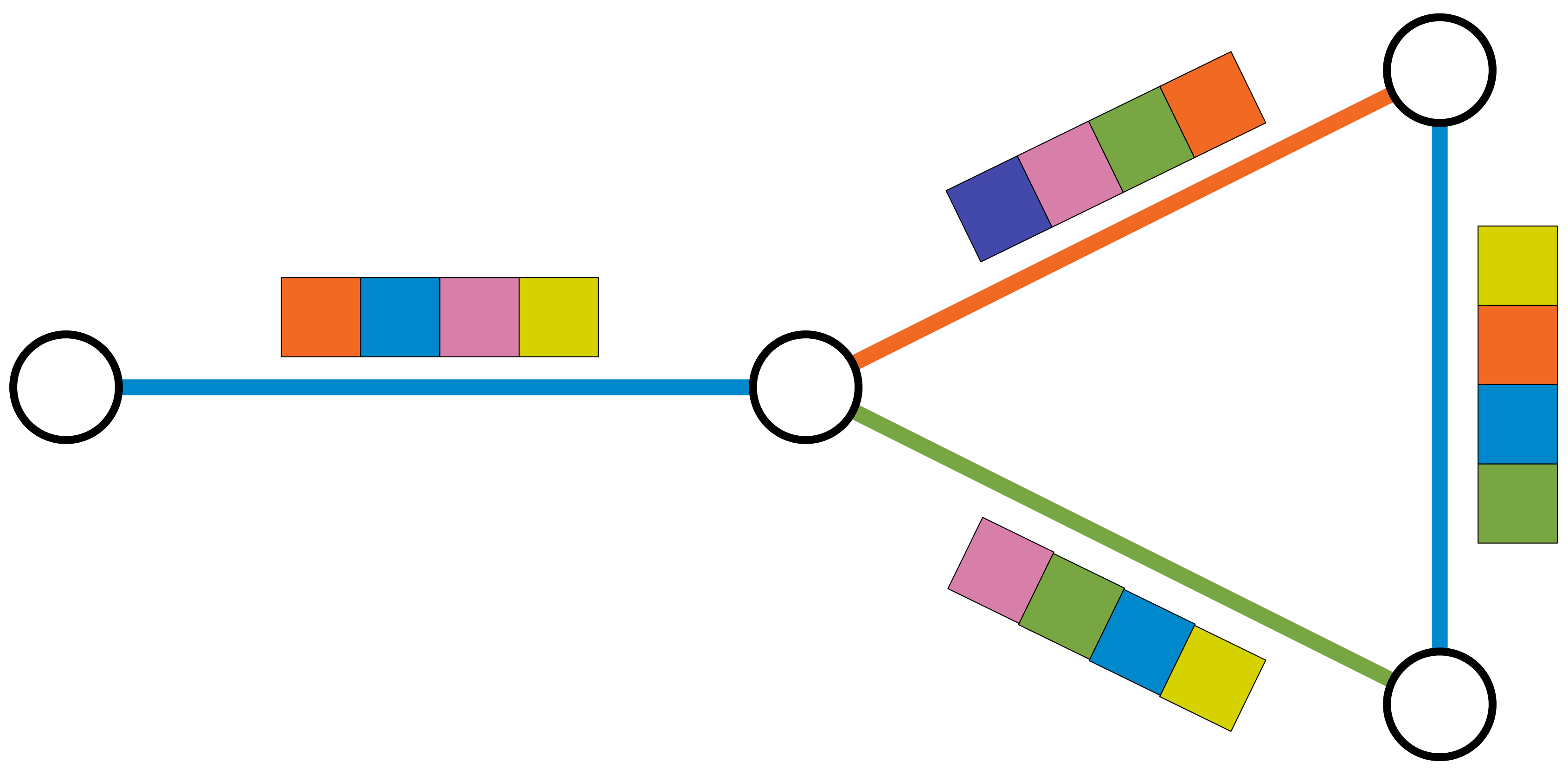
List edge coloring

Color palette:



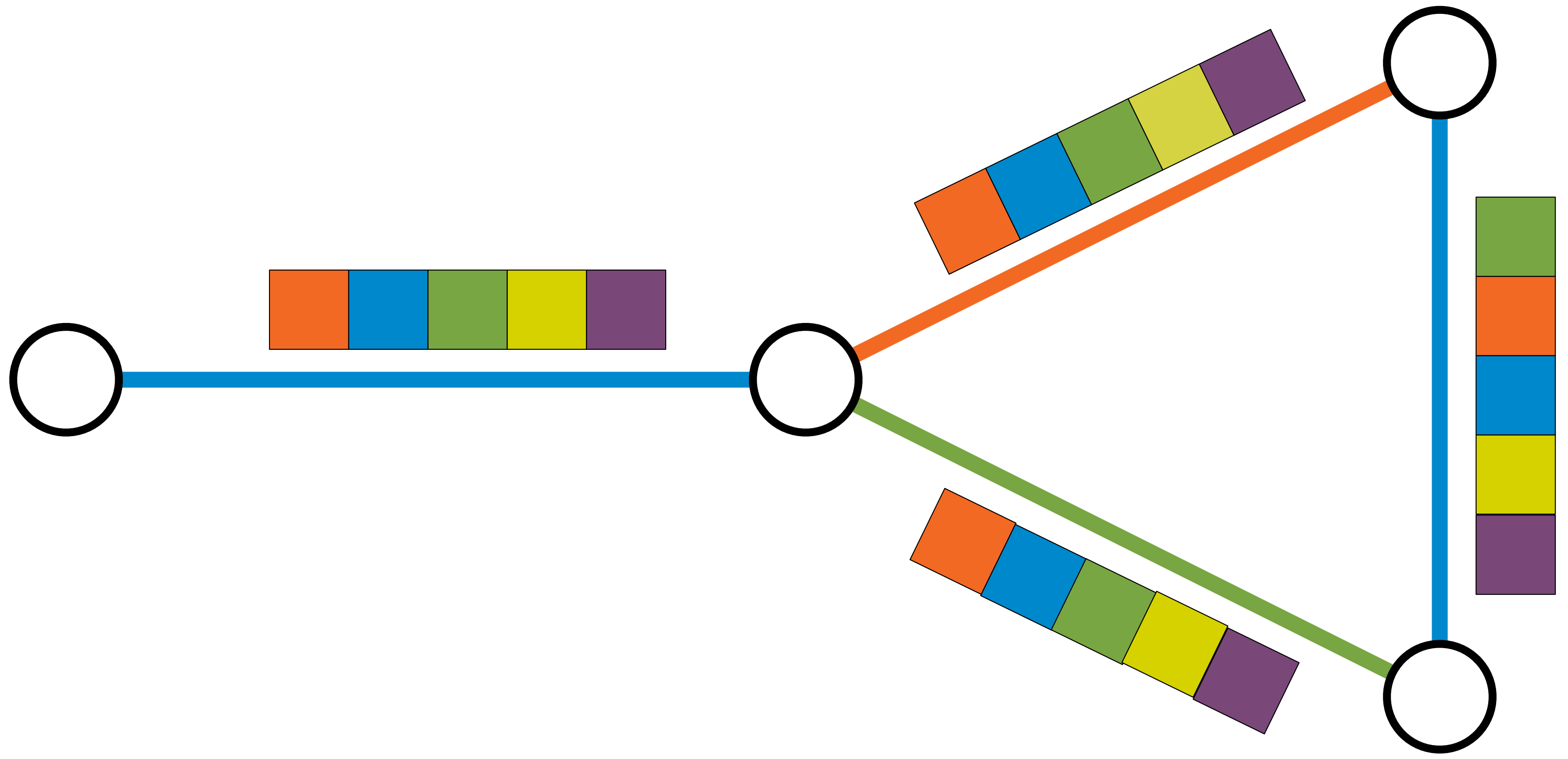
List edge coloring

(deg(e) + 1)-list edge coloring: lists of at least deg(e) + 1 colors



List edge coloring

- **(2Δ - 1)-edge coloring**:
  - lists of 2Δ - 1 colors
  - all lists the same

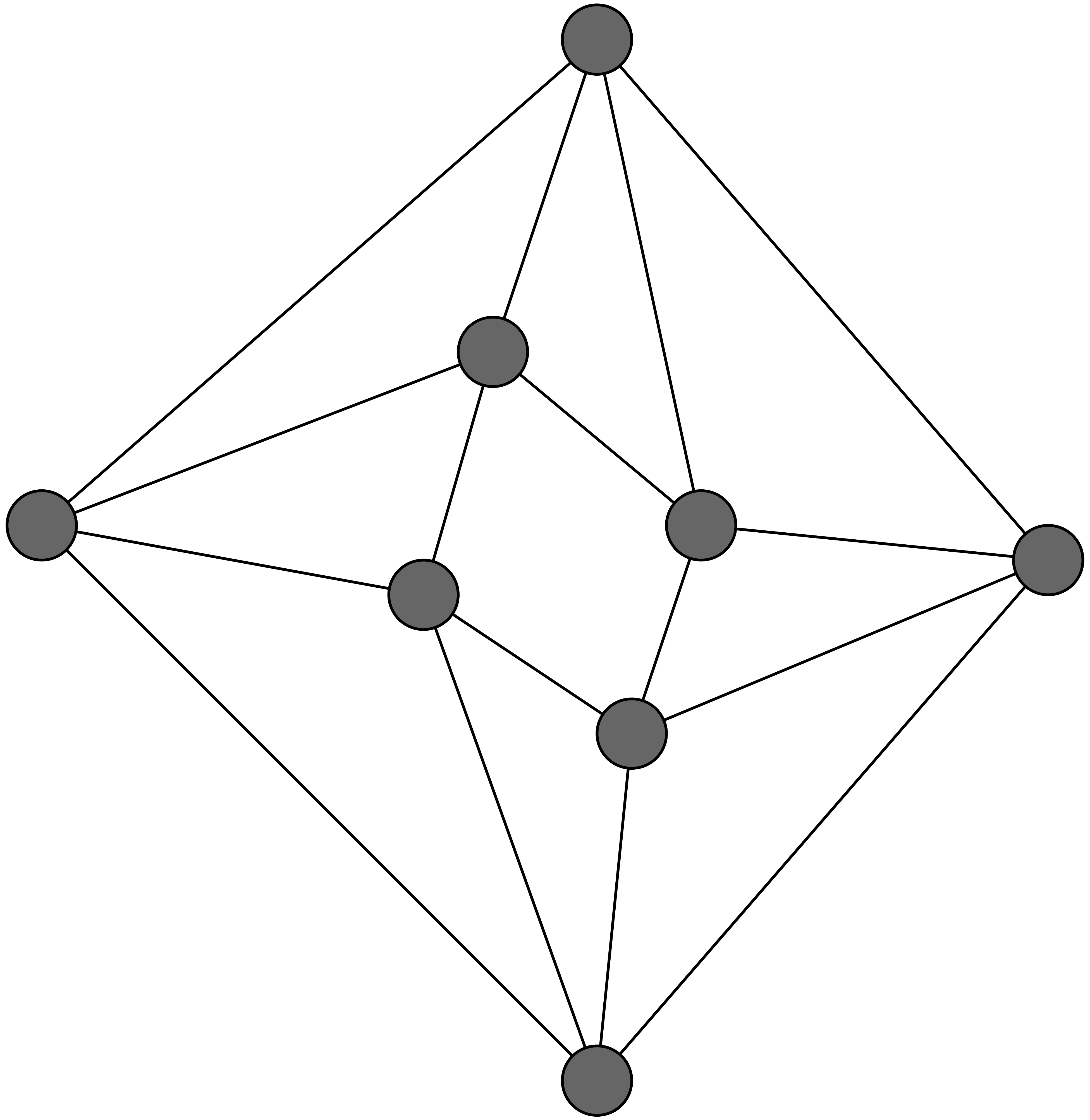


Why list coloring

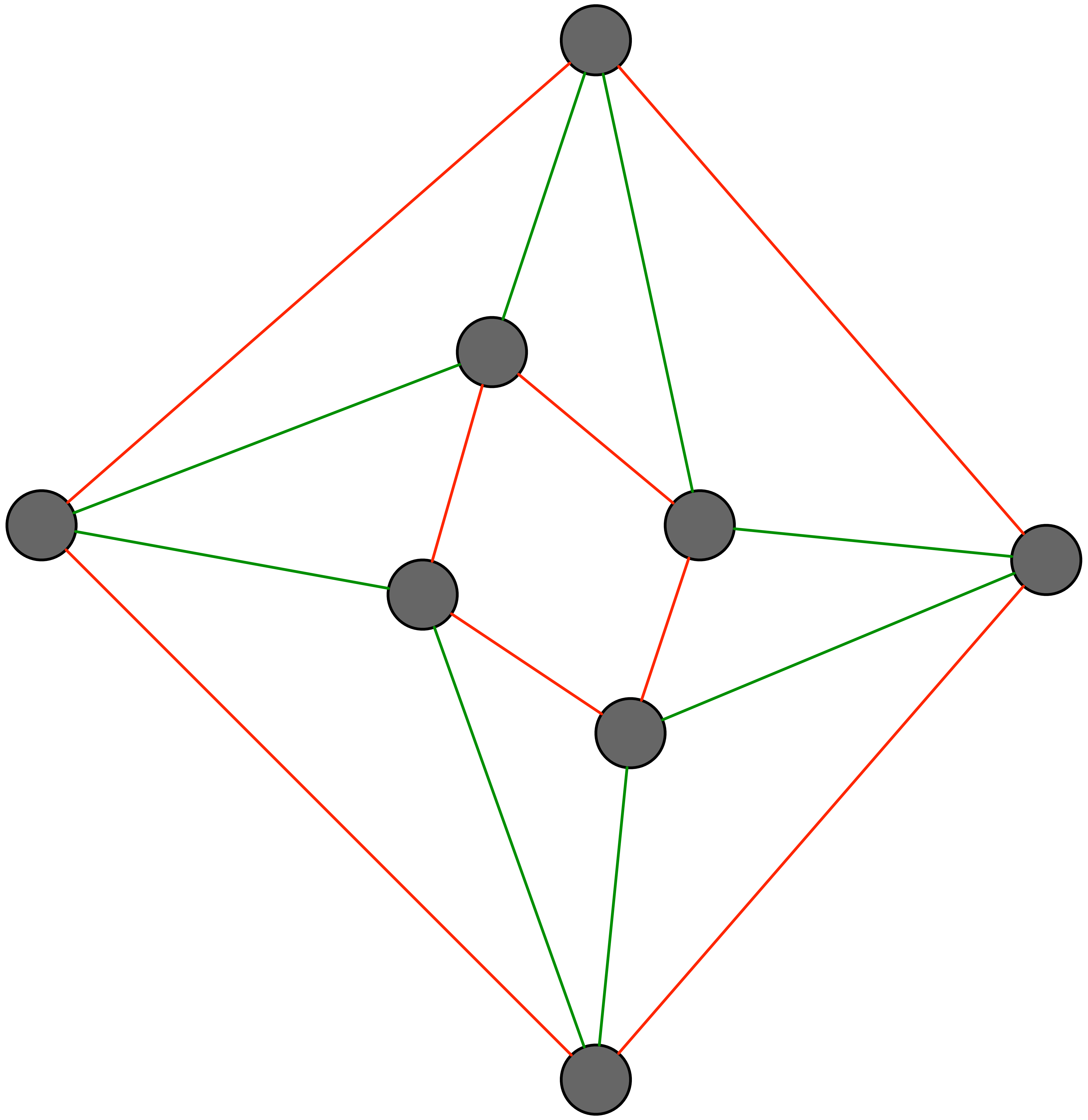
Let's try the following (non list coloring based) algorithm:

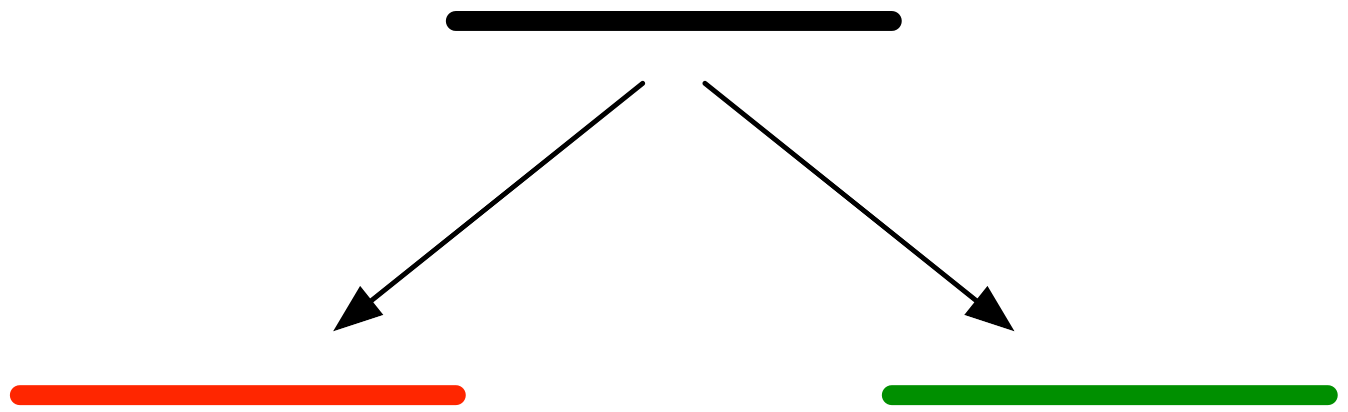
- Start from a graph of **maximum degree Δ**
- **2-color** the edges such that the graph induced by each color has **maximum degree Δ/2**
- **Recurse** on each subgraph

Why list coloring

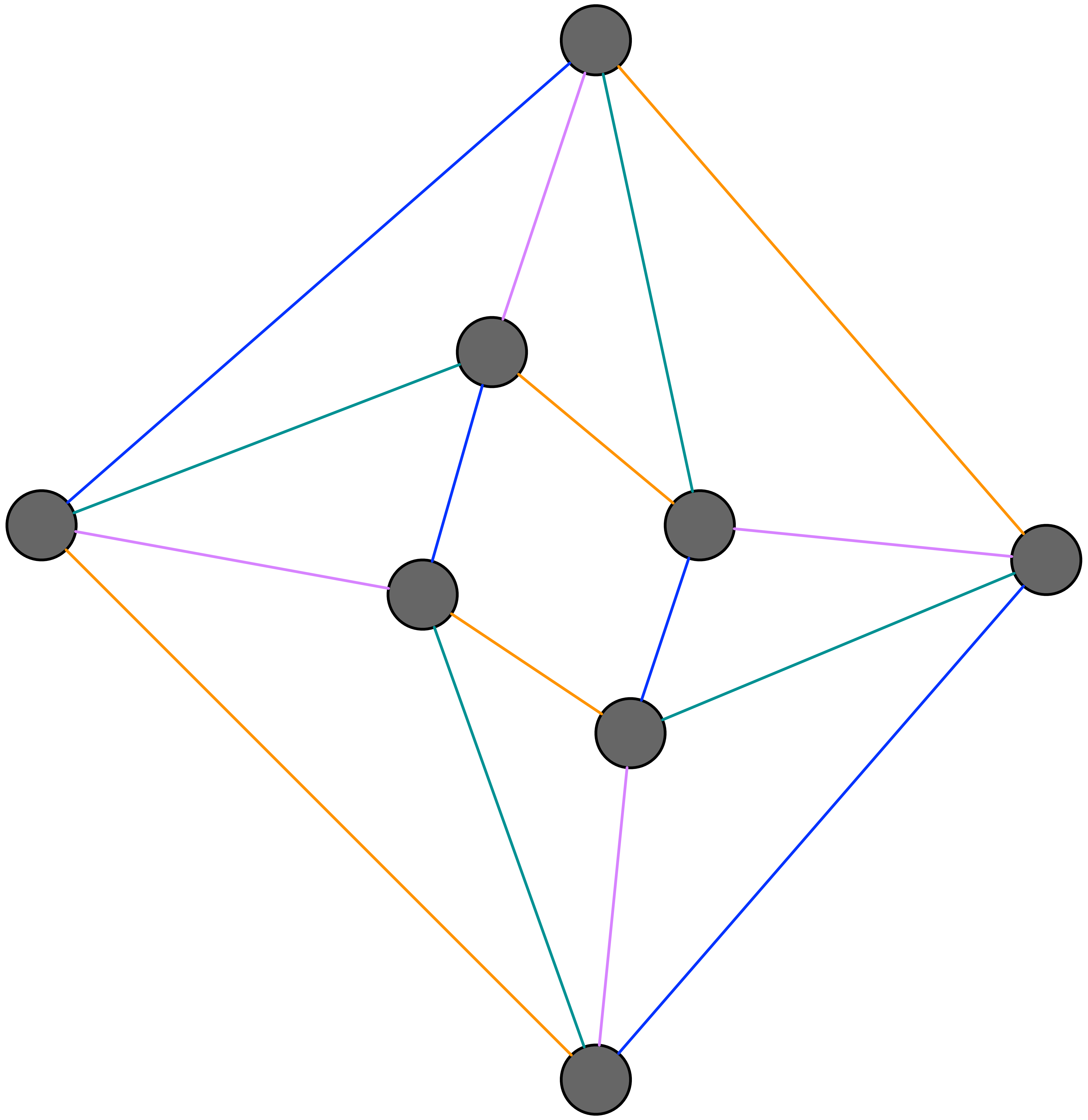


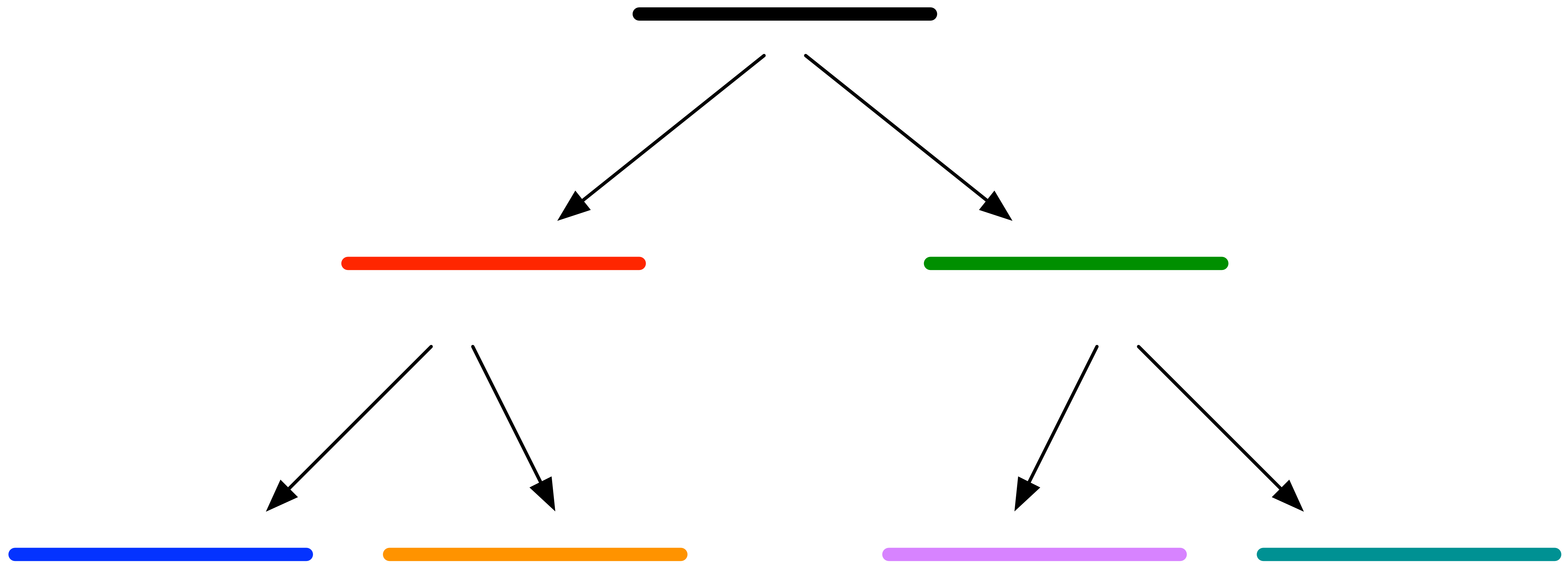
Why list coloring





Why list coloring





Why list coloring

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

Too hard

Why list coloring

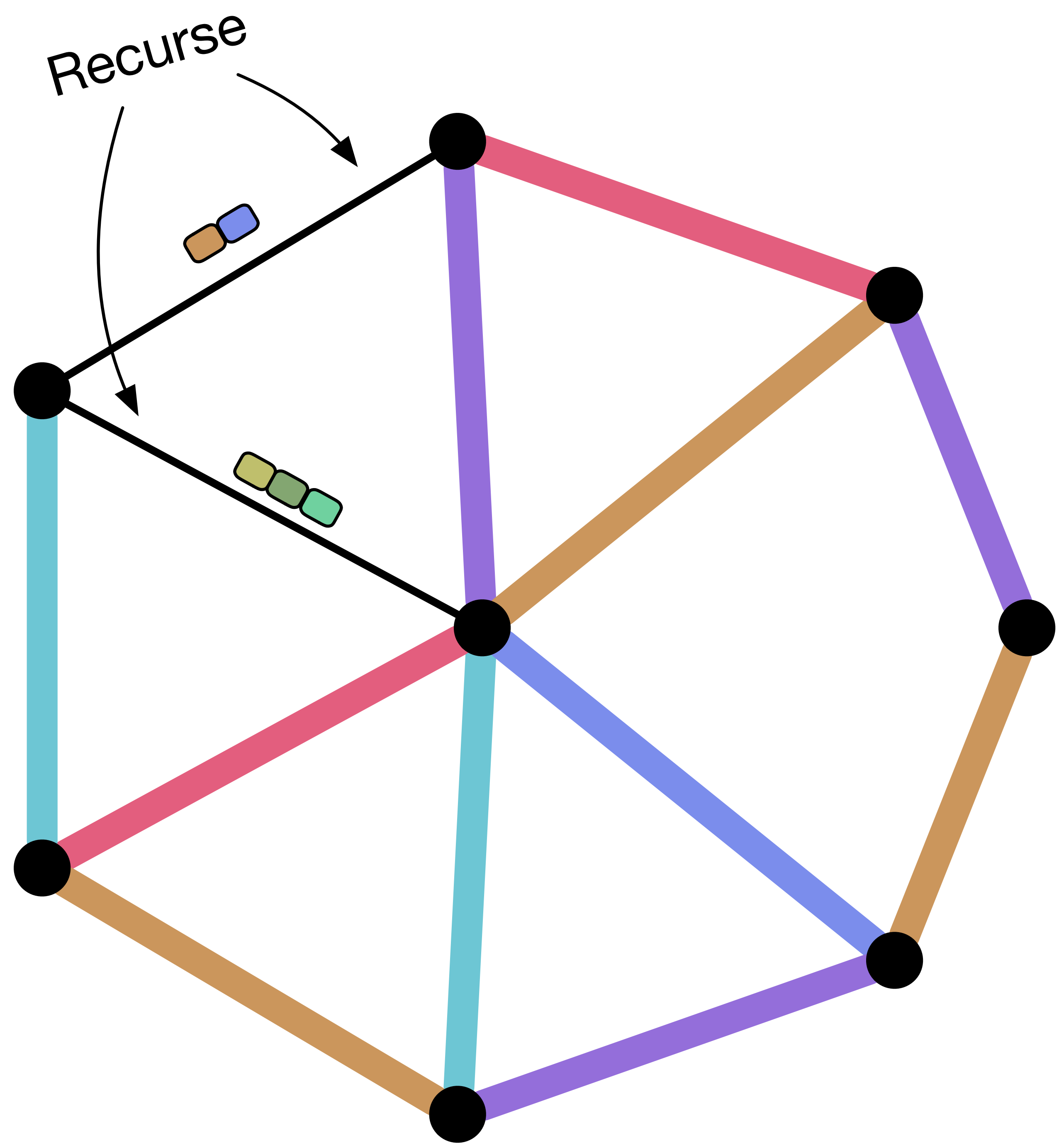
- Start from a graph of **maximum degree Δ**
- **c-color** the edges such that the graph induced by each color has **maximum degree O(Δ/c)**
- **Recurse** on each subgraph

We need too many colors

Why list coloring

- **Without lists**, by using a recursive algorithm, we have to **early commit** on color subspaces

- **With lists**, we can color a subgraph and then **recurse** on the remaining uncolored subgraph



$(\deg(e) + 1)$-list edge coloring in time $(\log \Delta)^{O(\log \log \Delta)} + O(\log^* n)$

High level ideas

Definitions

- **Goal**: $(\deg(e) + 1)$-list edge coloring
- **Relaxed version**: $(\beta \times \deg(e) + 1)$-list edge coloring

Definitions

slack 1

- **Goal**: $(\deg(e) + 1)$-list edge coloring
- **Relaxed version**: $(\beta \times \deg(e) + 1)$-list edge coloring

slack β

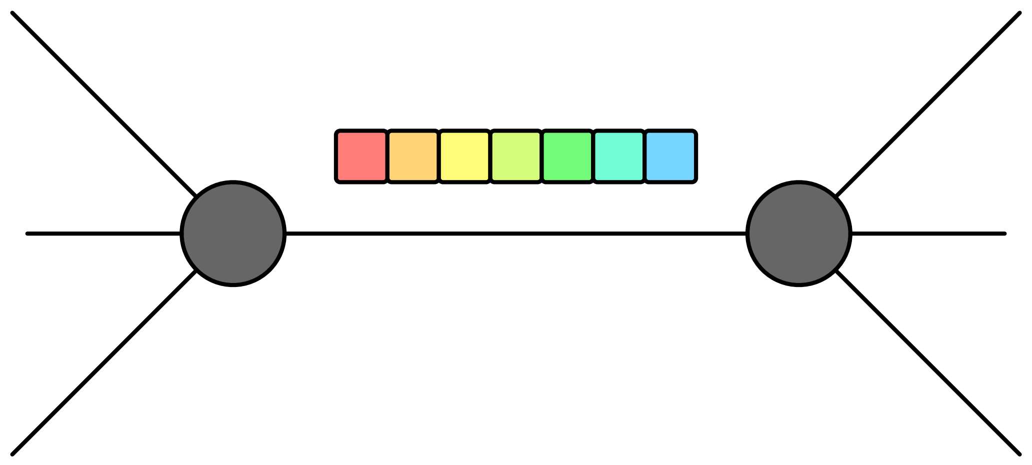
Definitions

slack 1

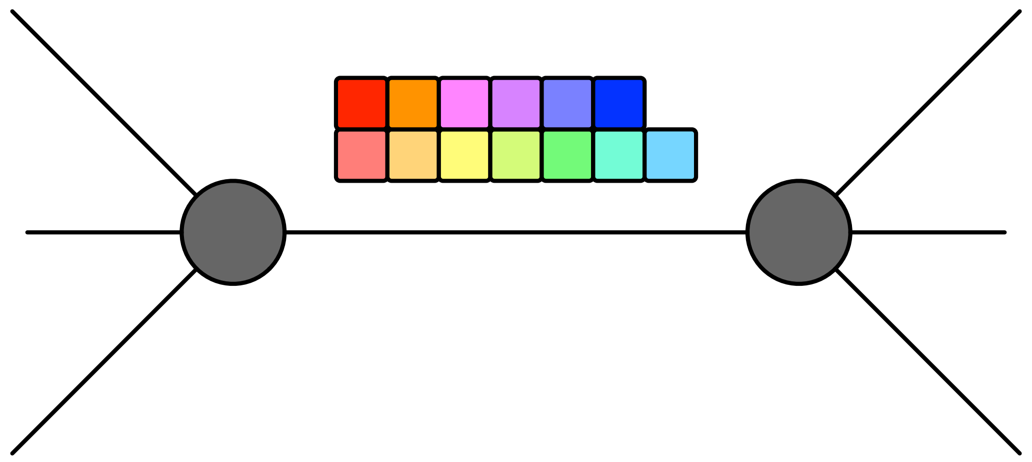
- **Goal**: (deg(e) + 1)-list edge coloring
- **Relaxed version**: (β × deg(e) + 1)-list edge coloring

slack β

slack 1:



slack 2:



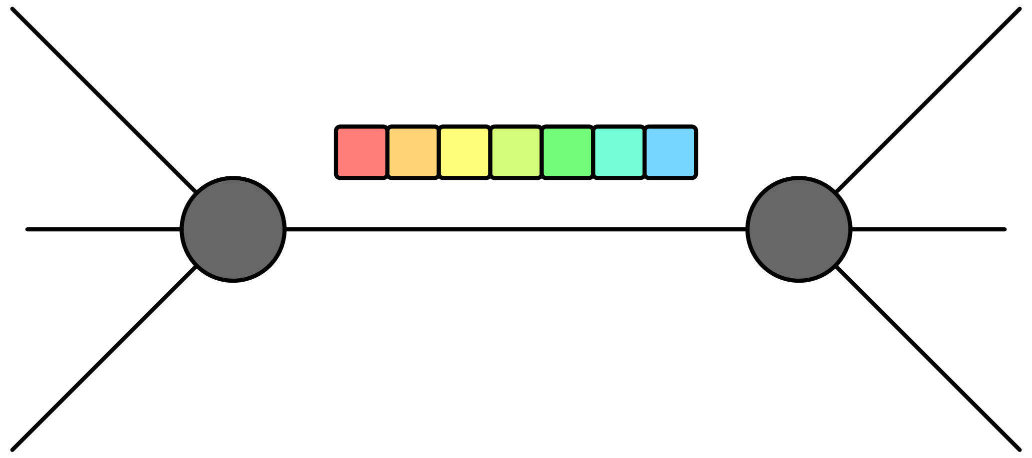
Definitions

slack 1

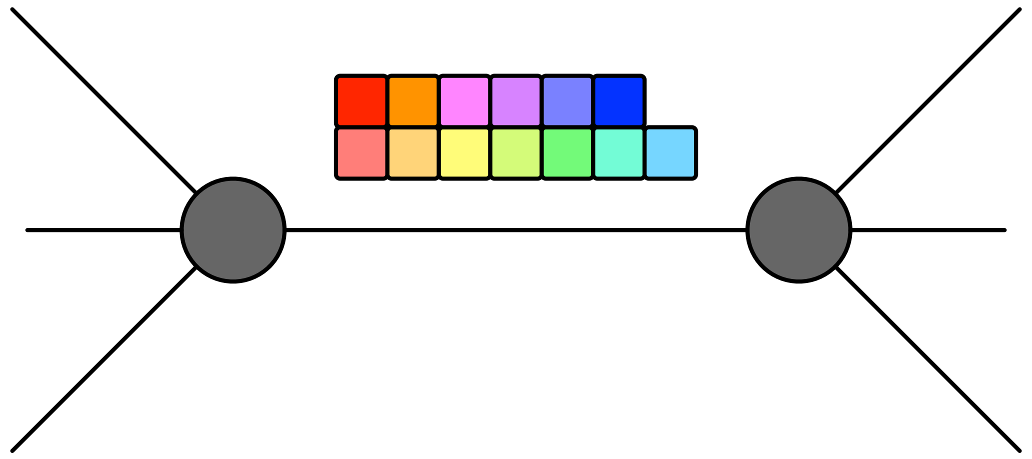
- **Goal**: (deg(e) + 1)-list edge coloring
- **Relaxed version**: (β × deg(e) + 1)-list edge coloring

slack β

slack 1:



slack 2:



T(β, **C**) = time required to solve a list coloring instance with a palette of size **C** and slack β

High level idea

$$T(1, \mathbf{C}) \leq \beta^2 \cdot \log \Delta \cdot T(\beta, \mathbf{C})$$

$$T(\beta, \mathbf{C}) \leq \log p \cdot T(1, \mathbf{p}) + T(\beta/\text{polylog } p, \mathbf{C/p})$$

High level idea

$$T(1, \mathbf{C}) \le \beta^2 \cdot \log \Delta \cdot T(\beta, \mathbf{C})$$

$$T(\beta, \mathbf{C}) \le \log p \cdot T(1, \mathbf{p}) + T(\beta/\text{polylog } p, \mathbf{C/p})$$

$$T(1, \mathbf{\Delta}) \le \text{polylog } \Delta \cdot T(1, \mathbf{\sqrt{\Delta}})$$

$$T(1, \mathbf{\Delta}) \le (\log \Delta)^{O(\log \log \Delta)} \cdot T(1, \mathbf{O(1)}) = (\log \Delta)^{O(\log \log \Delta)}$$

High level idea

$$T(1, \mathbf{C}) \le \beta^2 \cdot \log \Delta \cdot T(\beta, \mathbf{C})$$

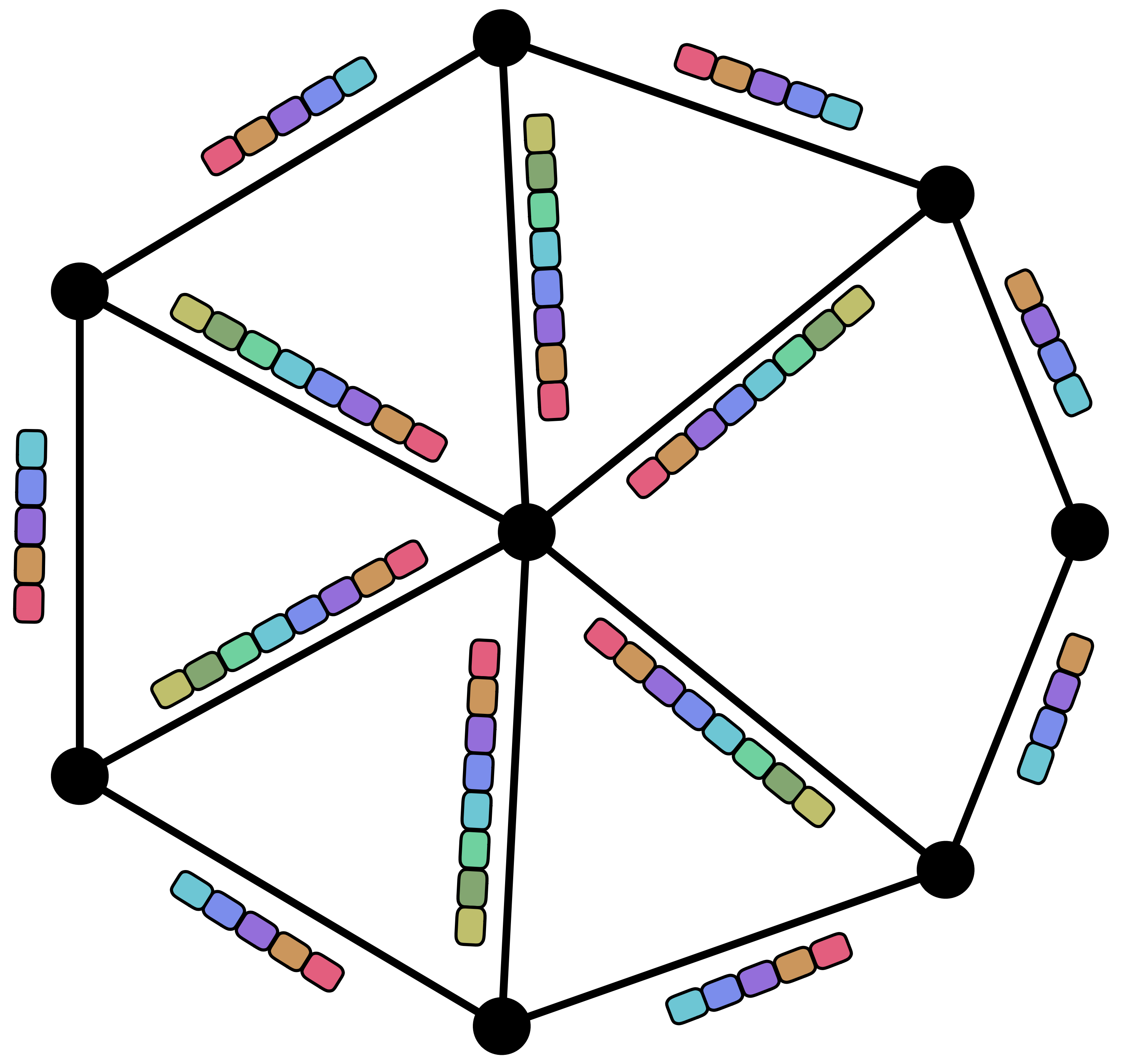
$$T(\beta, \mathbf{C}) \le \log p \cdot T(1, \mathbf{p}) + T(\beta/\mathrm{polylog}\, p, \mathbf{C/p})$$

$$T(1, \mathbf{\Delta}) \le \mathrm{polylog}\, \Delta \cdot T(1, \mathbf{\sqrt{\Delta}})$$

$$T(1, \mathbf{\Delta}) \le (\log \Delta)^{O(\log\log \Delta)} \cdot T(1, \mathbf{O(1)}) = (\log \Delta)^{O(\log\log \Delta)}$$

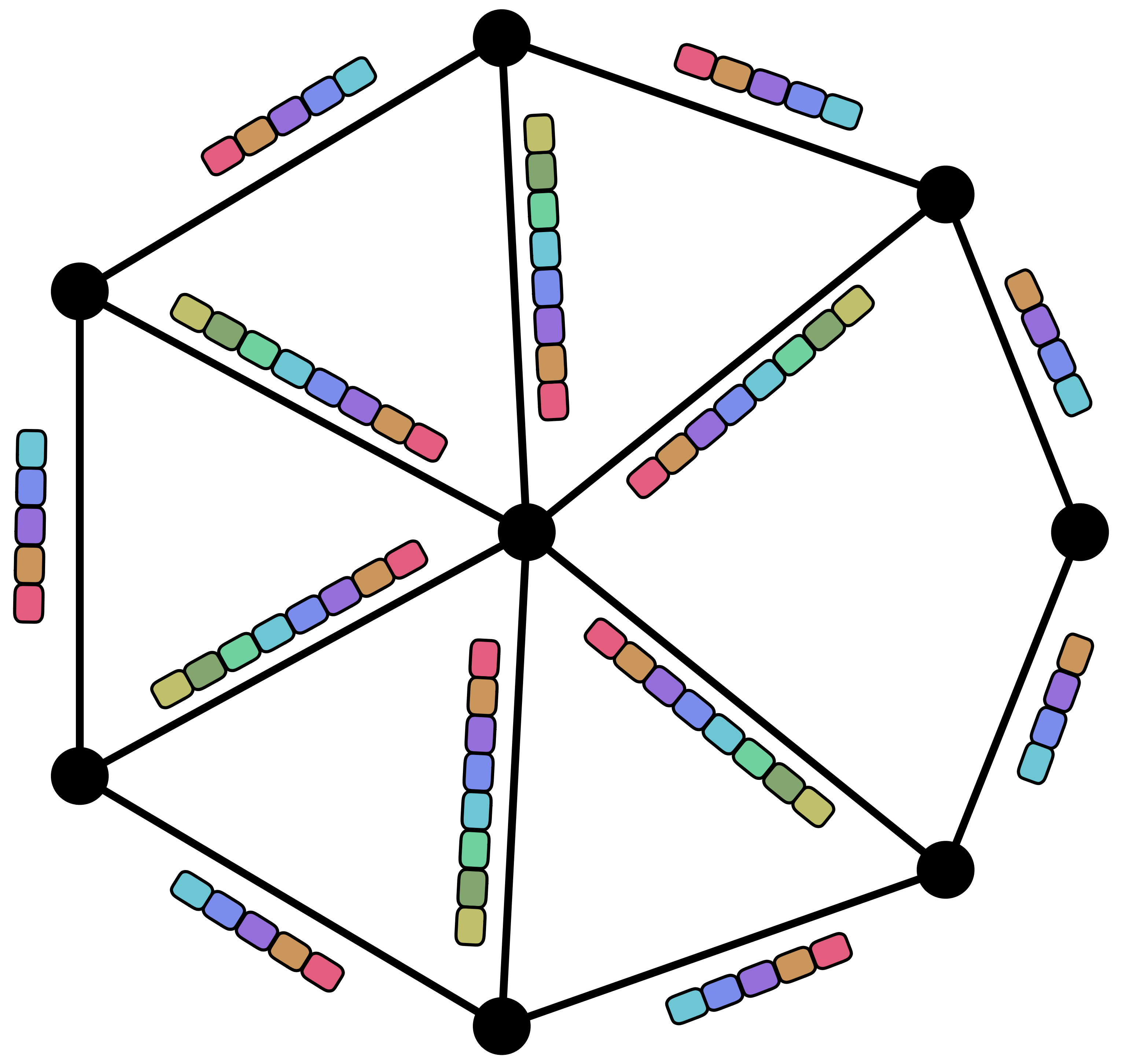
Increasing the slack

- Suppose we can solve “fast” a list edge coloring with slack β



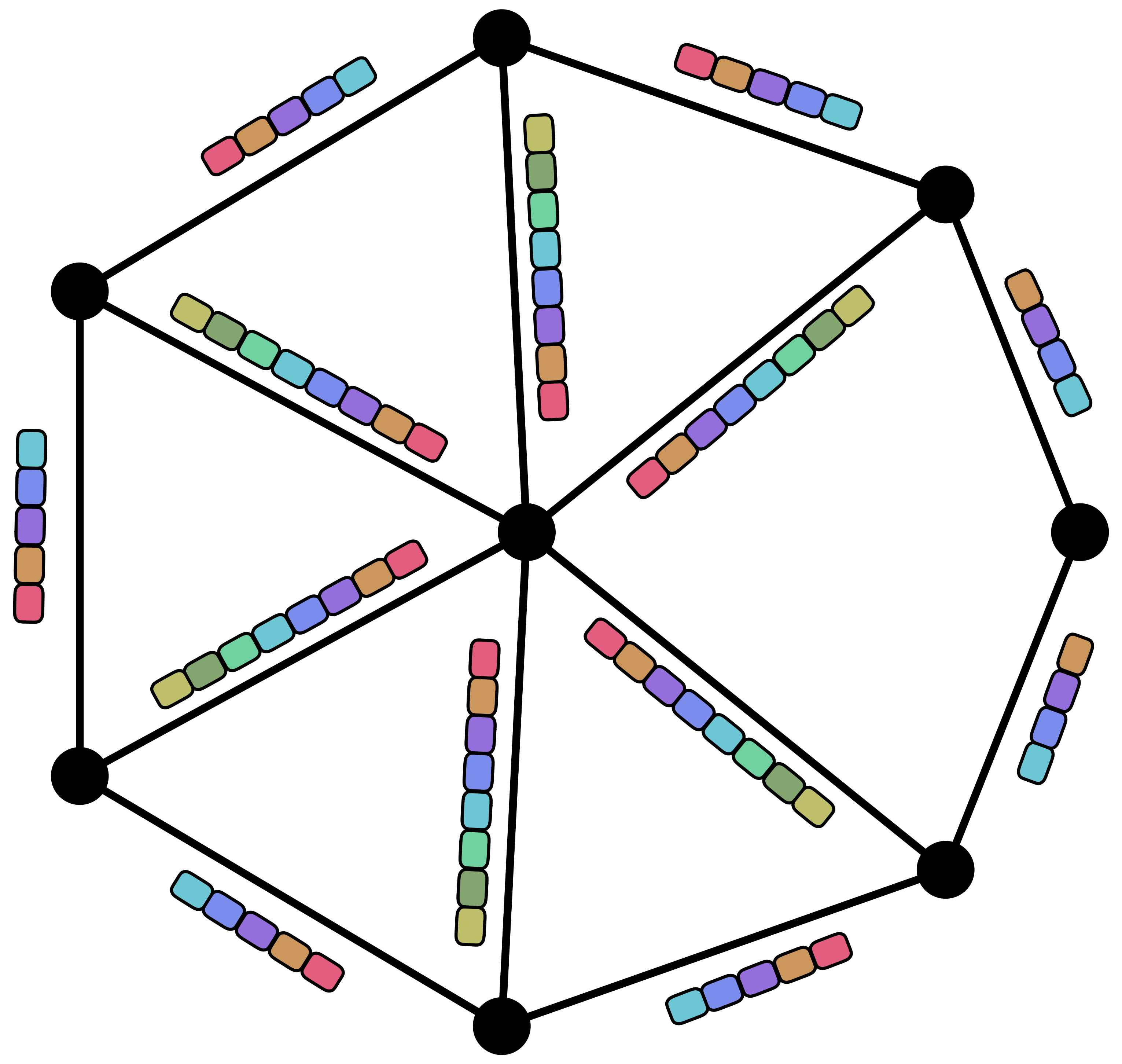
Increasing the slack

- Suppose we can solve “fast” a list edge coloring with slack β
- Reduce the degree by computing a defective edge coloring



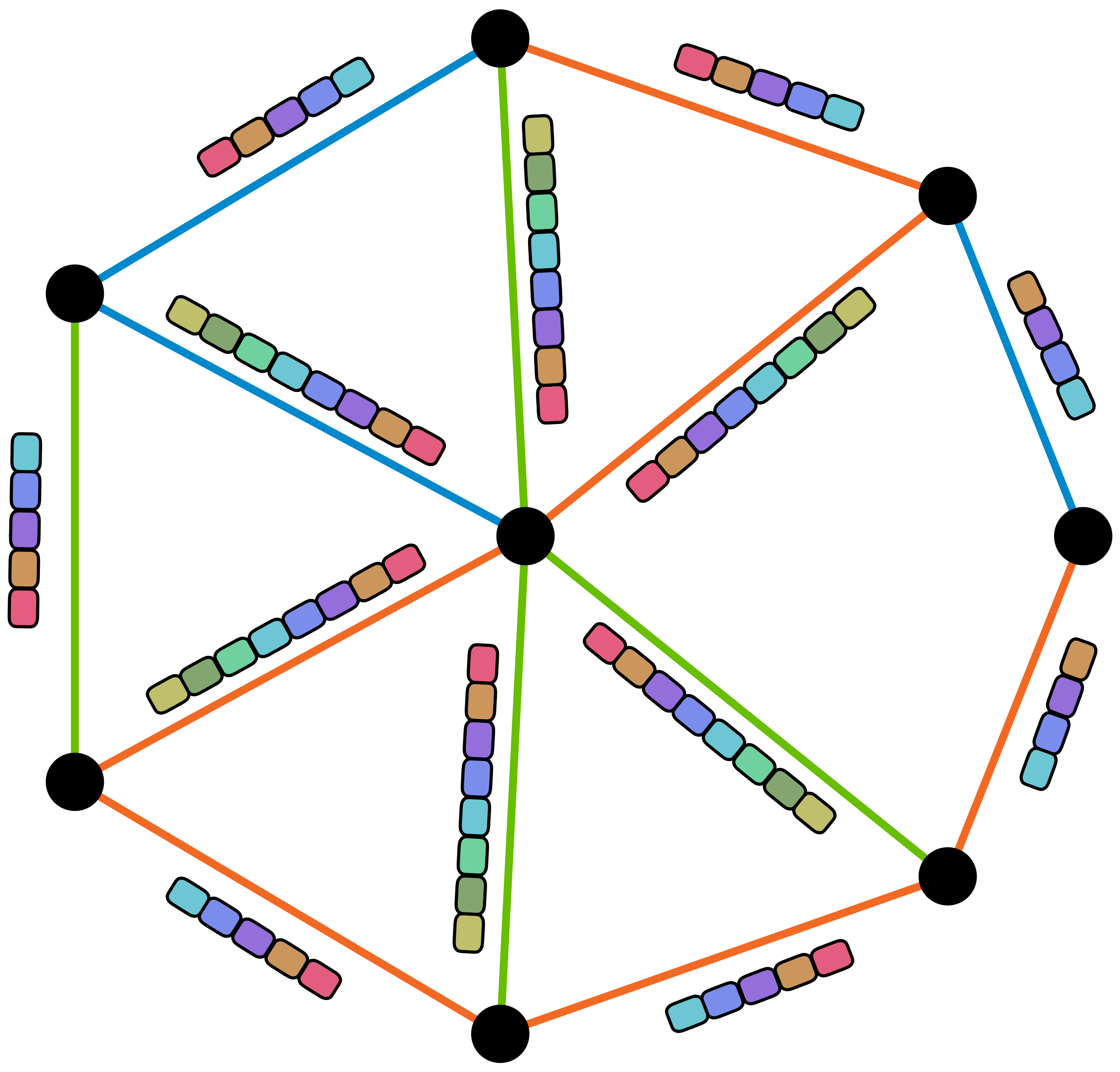
Increasing the slack

- Suppose we can solve "fast" a list edge coloring with slack β
- Reduce the degree by computing a defective edge coloring
- Solve many instances of relaxed list edge coloring sequentially (by going through color classes)



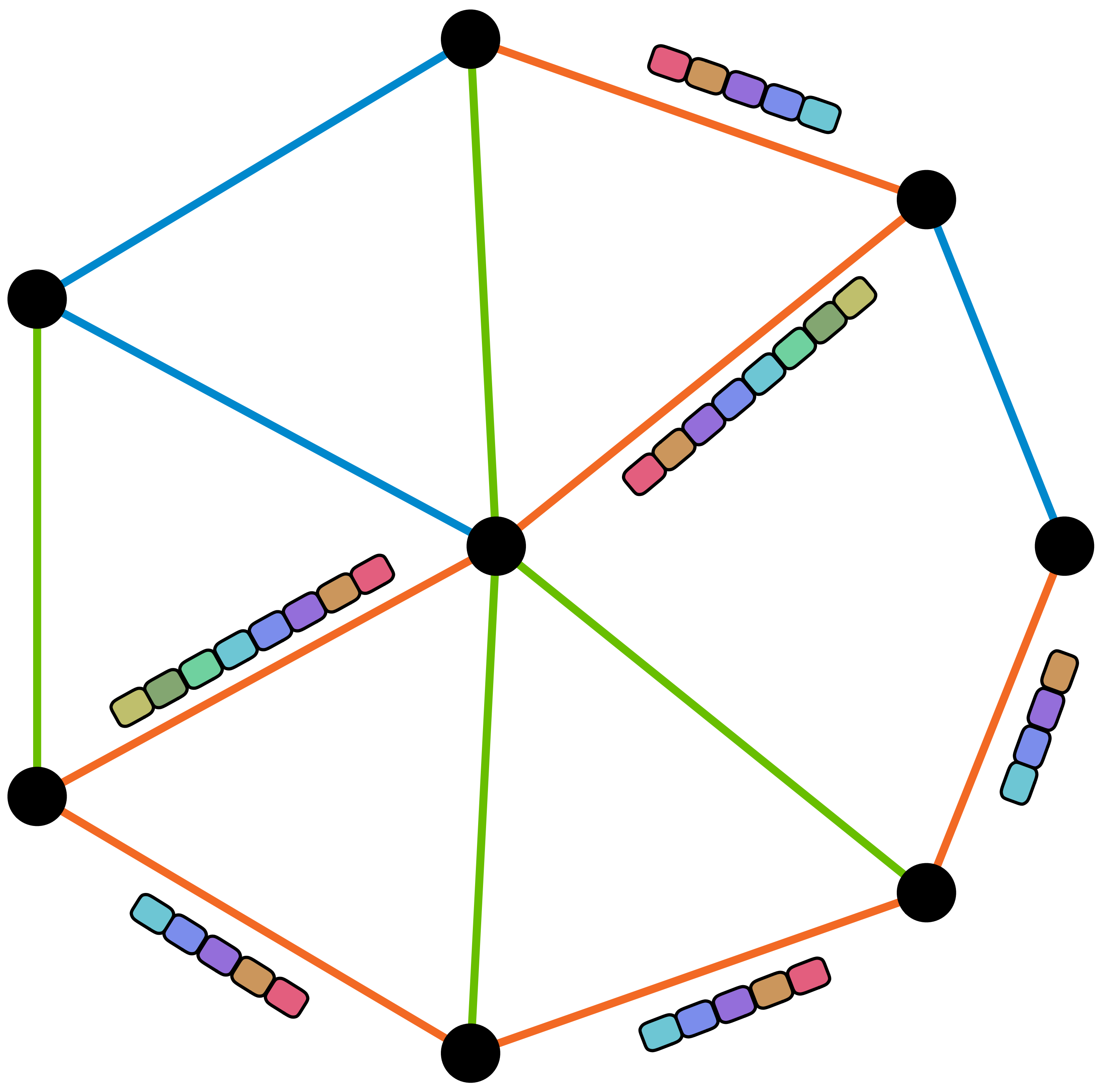
Increasing the slack

- Suppose we can solve "fast" a list edge coloring with slack β
- Reduce the degree by computing a defective edge coloring
- Solve many instances of relaxed list edge coloring sequentially (by going through color classes)



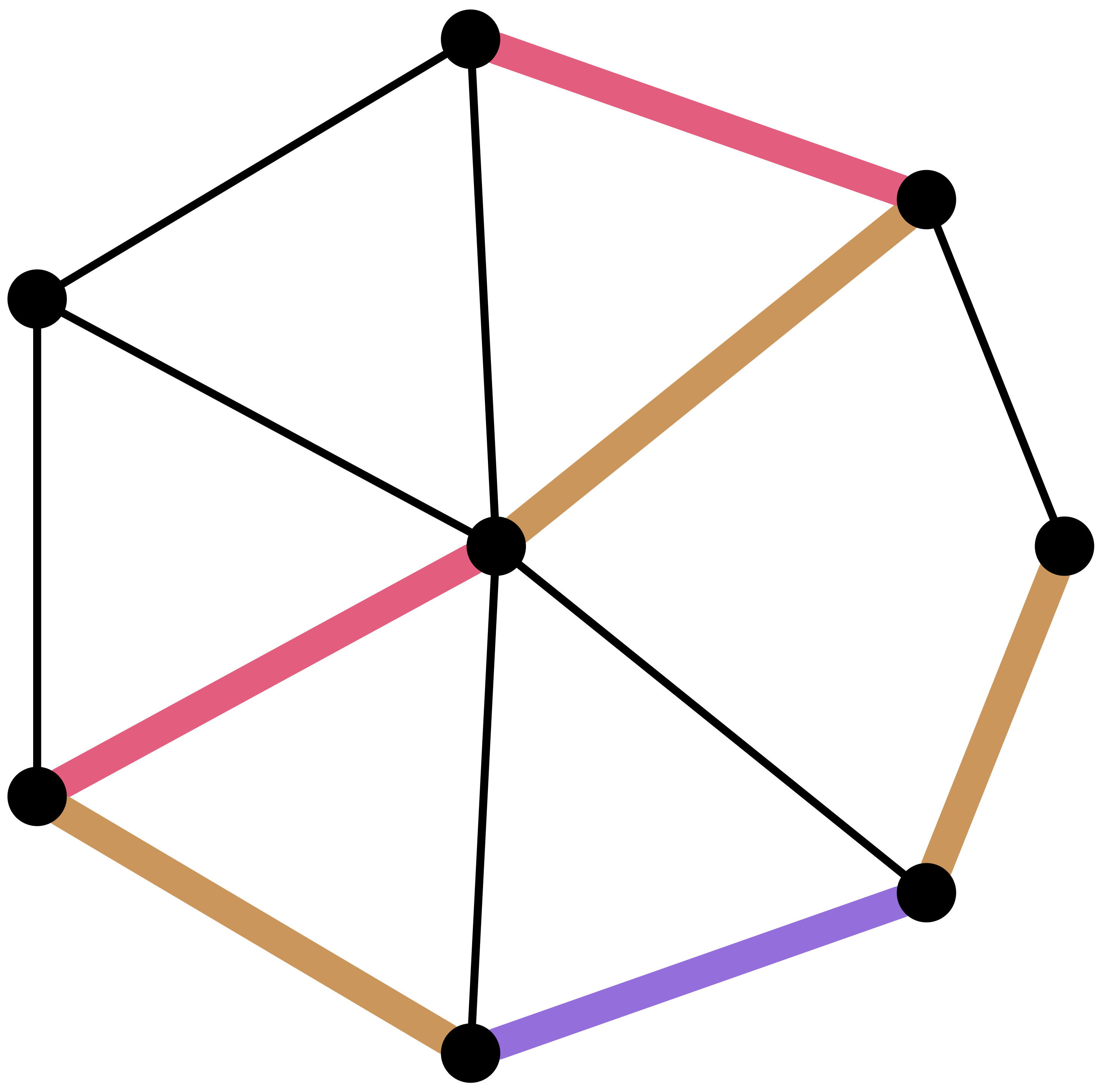
Increasing the slack

- Suppose we can solve “fast” a list edge coloring with slack β
- Reduce the degree by computing a defective edge coloring
- Solve many instances of relaxed list edge coloring sequentially (by going through color classes)



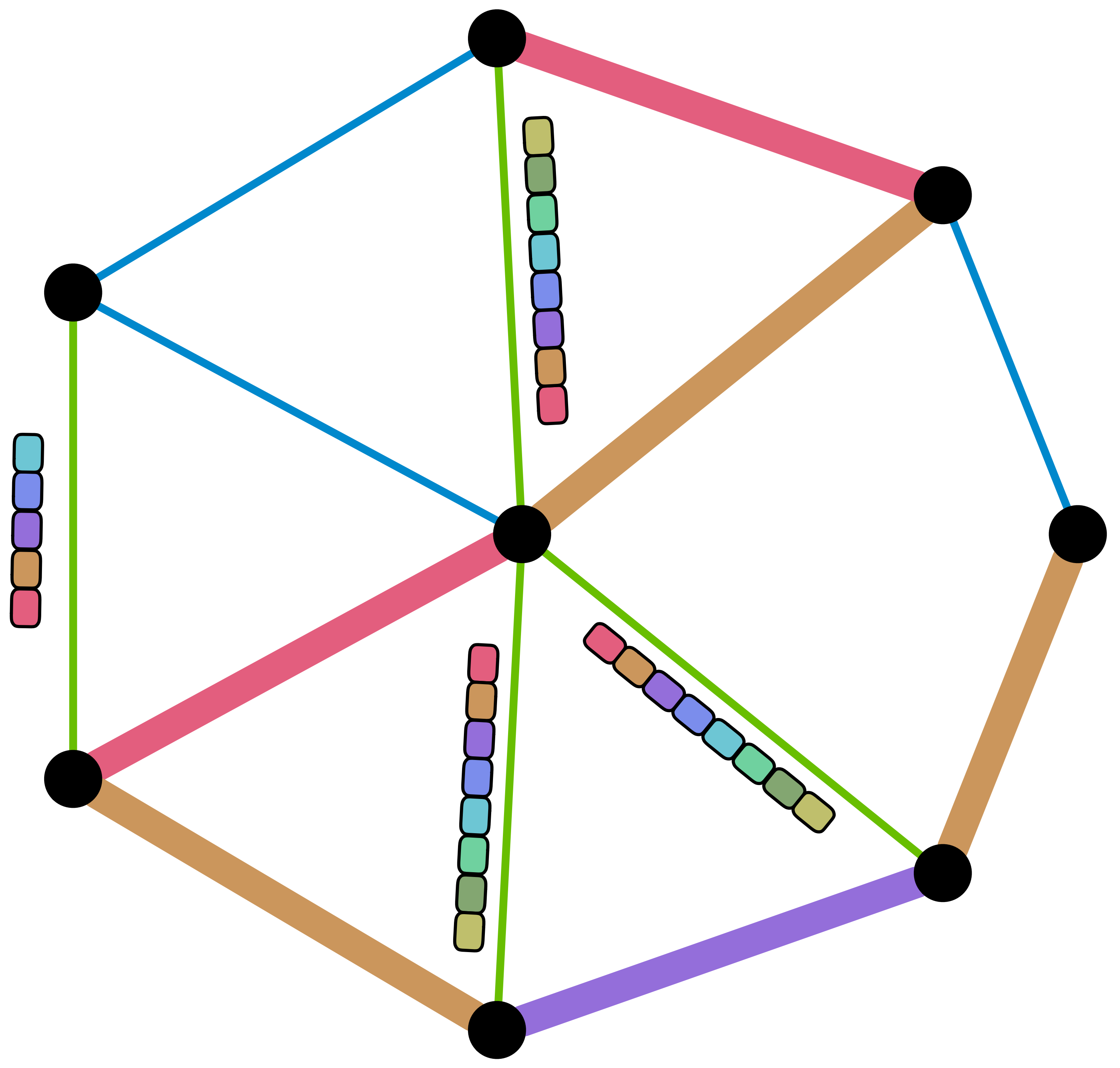
Increasing the slack

- Suppose we can solve "fast" a list edge coloring with slack β
- Reduce the degree by computing a defective edge coloring
- Solve many instances of relaxed list edge coloring sequentially (by going through color classes)



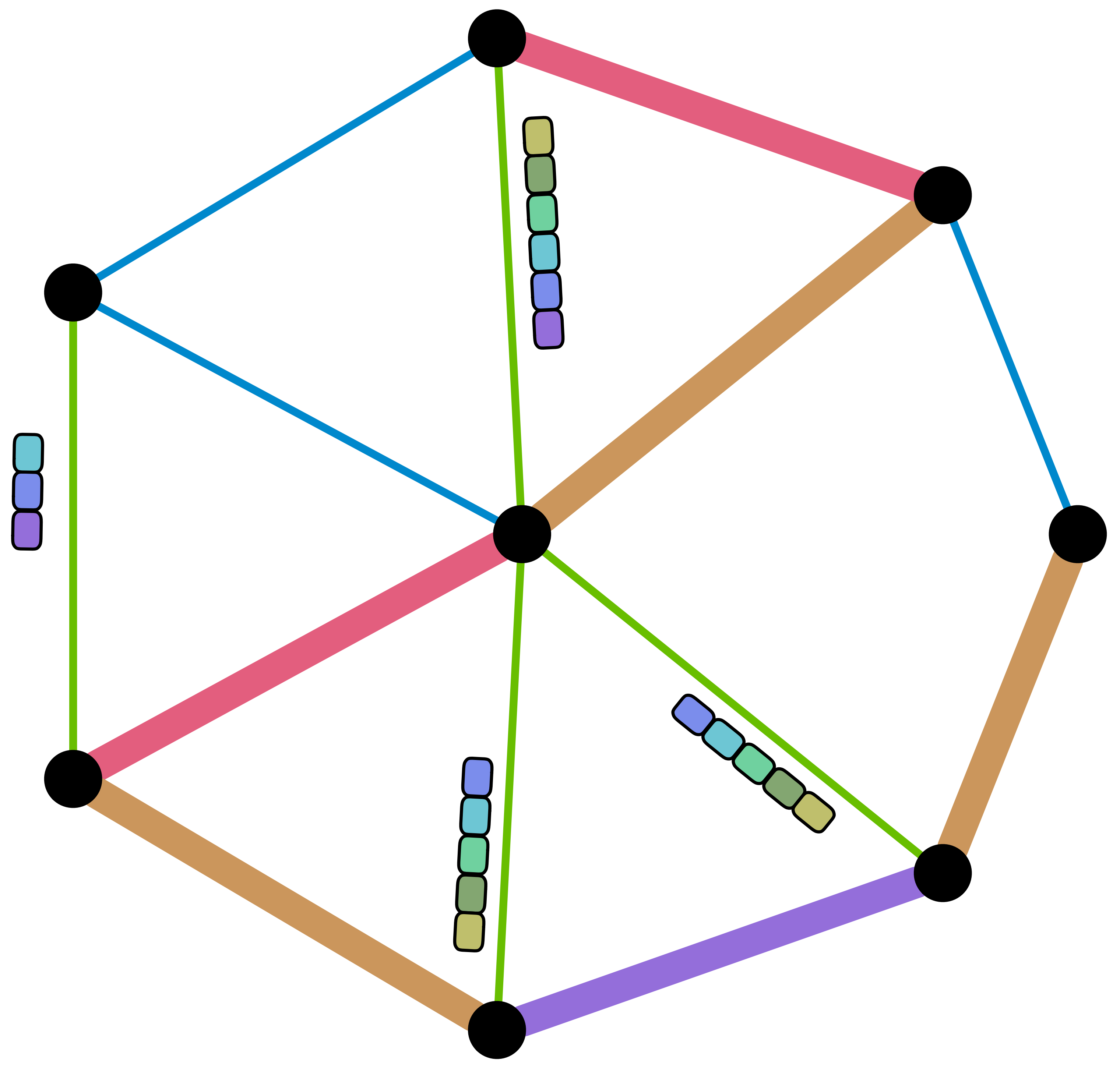
Increasing the slack

- Suppose we can solve “fast” a list edge coloring with slack β
- Reduce the degree by computing a defective edge coloring
- Solve many instances of relaxed list edge coloring sequentially (by going through color classes)



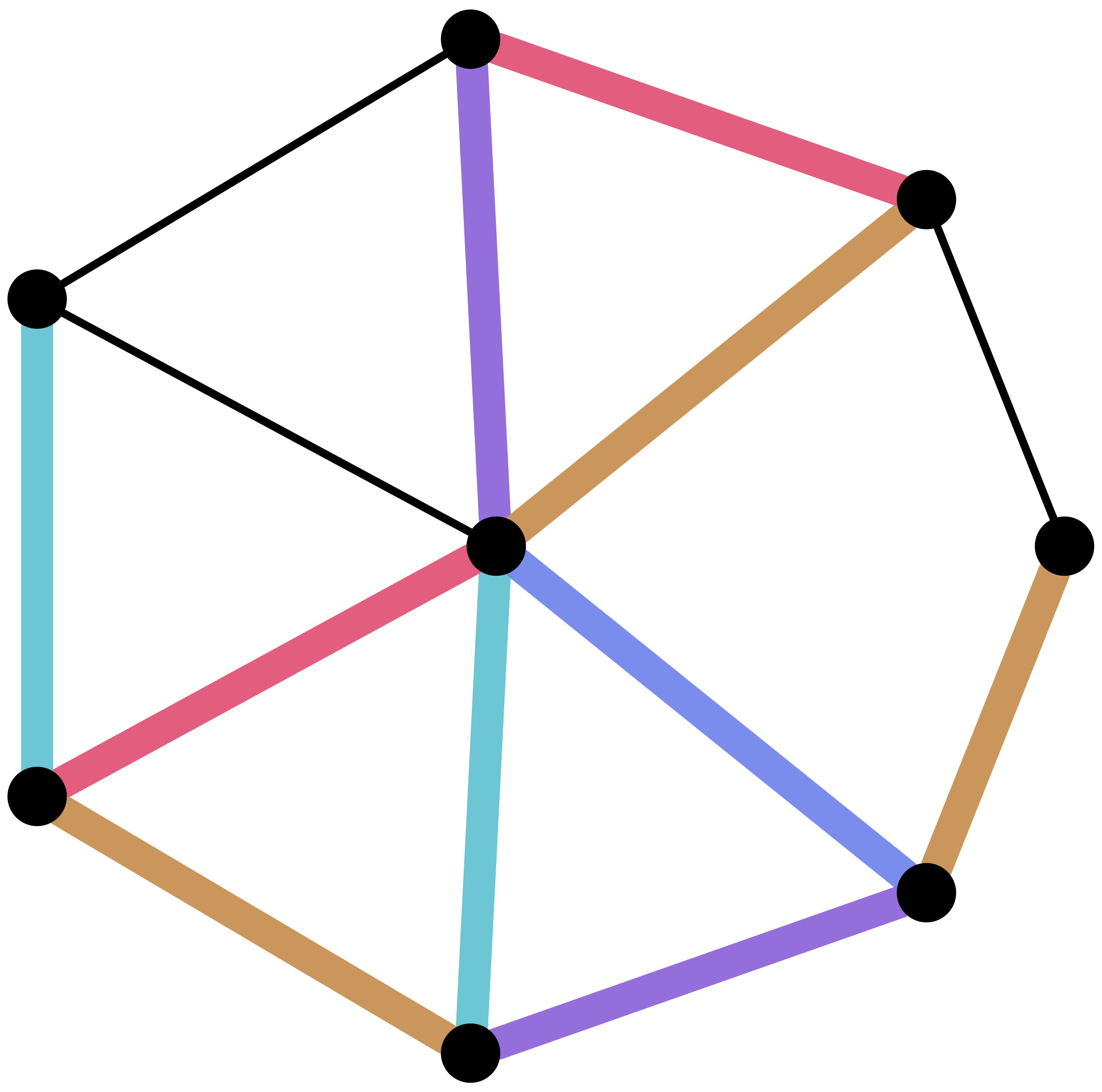
Increasing the slack

- Suppose we can solve "fast" a list edge coloring with slack β
- Reduce the degree by computing a defective edge coloring
- Solve many instances of relaxed list edge coloring sequentially (by going through color classes)



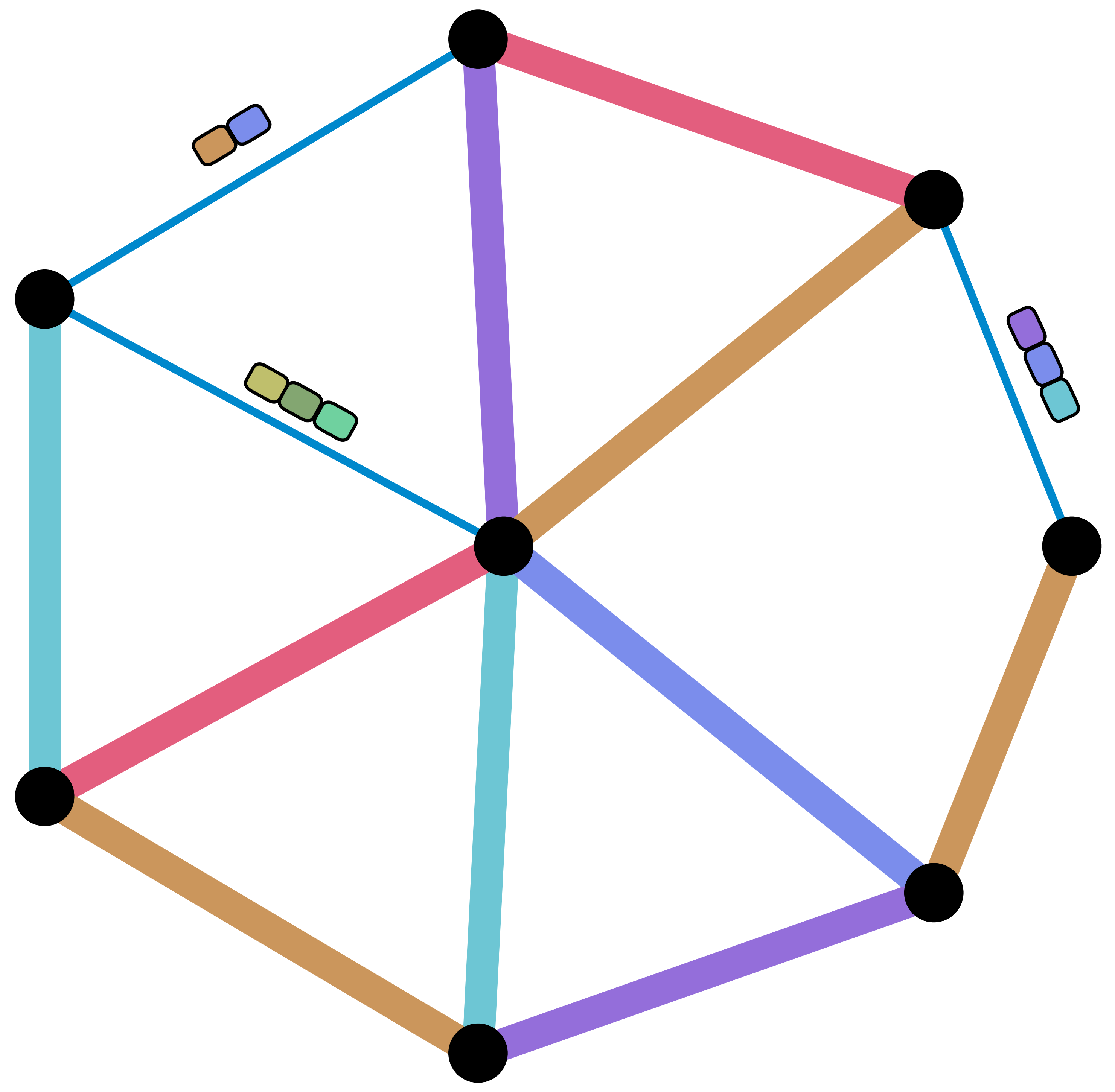
Increasing the slack

- Suppose we can solve "fast" a list edge coloring with slack β
- Reduce the degree by computing a defective edge coloring
- Solve many instances of relaxed list edge coloring sequentially (by going through color classes)



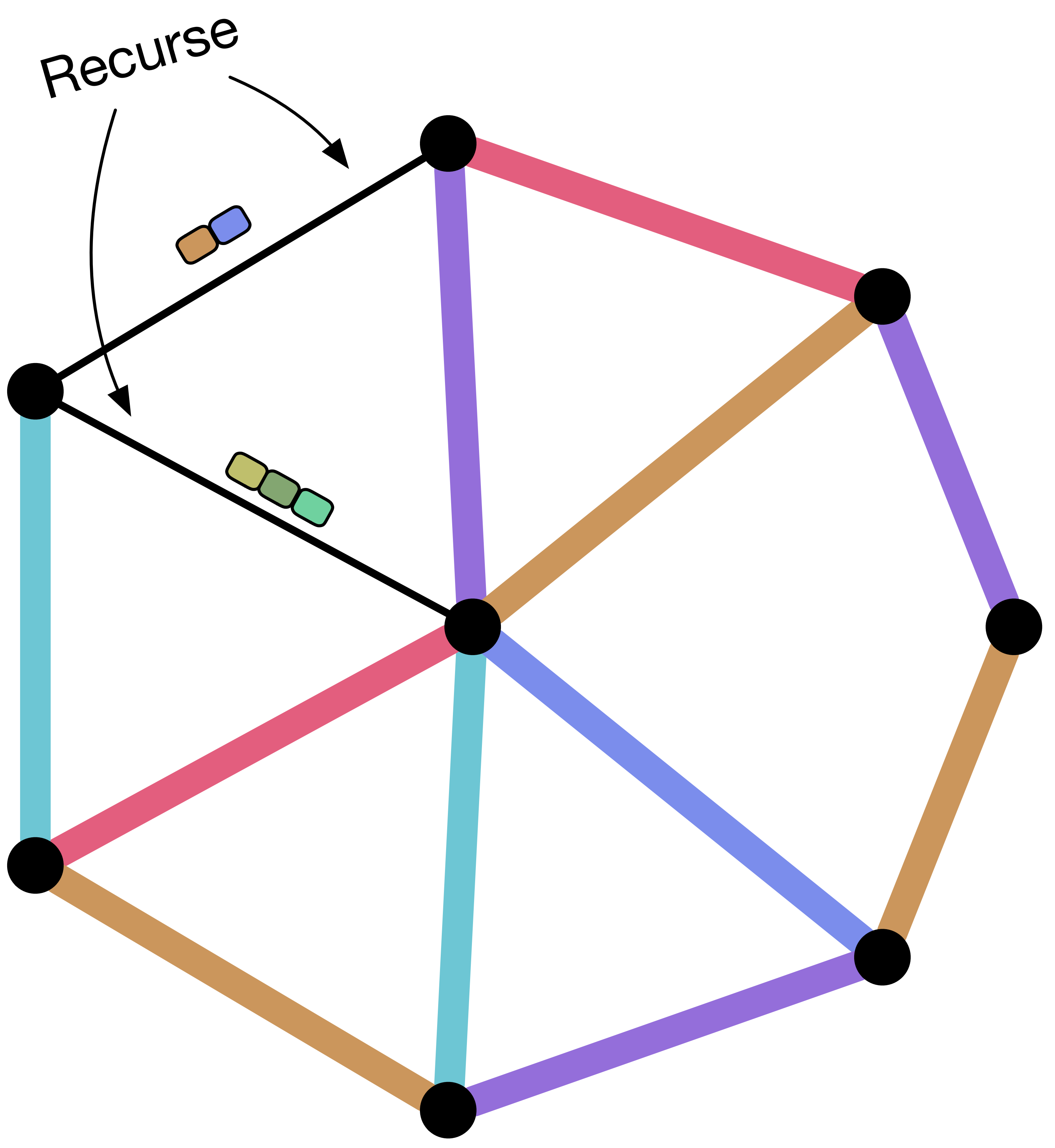
Increasing the slack

- Suppose we can solve "fast" a list edge coloring with slack β
- Reduce the degree by computing a defective edge coloring
- Solve many instances of relaxed list edge coloring sequentially (by going through color classes)



Increasing the slack

- Suppose we can solve "fast" a list edge coloring with slack β
- Reduce the degree by computing a defective edge coloring
- Solve many instances of relaxed list edge coloring sequentially (by going through color classes)



Increasing the slack

- Compute a **deg(e)/(2β)-defective** edge coloring **g(e)** with **$O(\beta^2)$** colors
- Iterate through each color class **i**. Edges of color **i** do the following:
  - Remove from the list the colors **c(e')** already used by the neighbors
  - If the list has size larger than **deg(e)/2** stay **active**
  - Apply the algorithm for slack **β** on active edges, obtain **c(e)**
- **Recurse** on uncolored edges

Increasing the slack

$$T(1, \mathbf{C}) \le T(\text{defective-coloring}) + \text{nr\_color\_classes} \cdot T(\text{large slack}) + T(\text{recursion})$$

$$T(1, \mathbf{C}) \le \beta^2 \cdot \log \Delta \cdot T(\beta, \mathbf{C})$$

High level idea

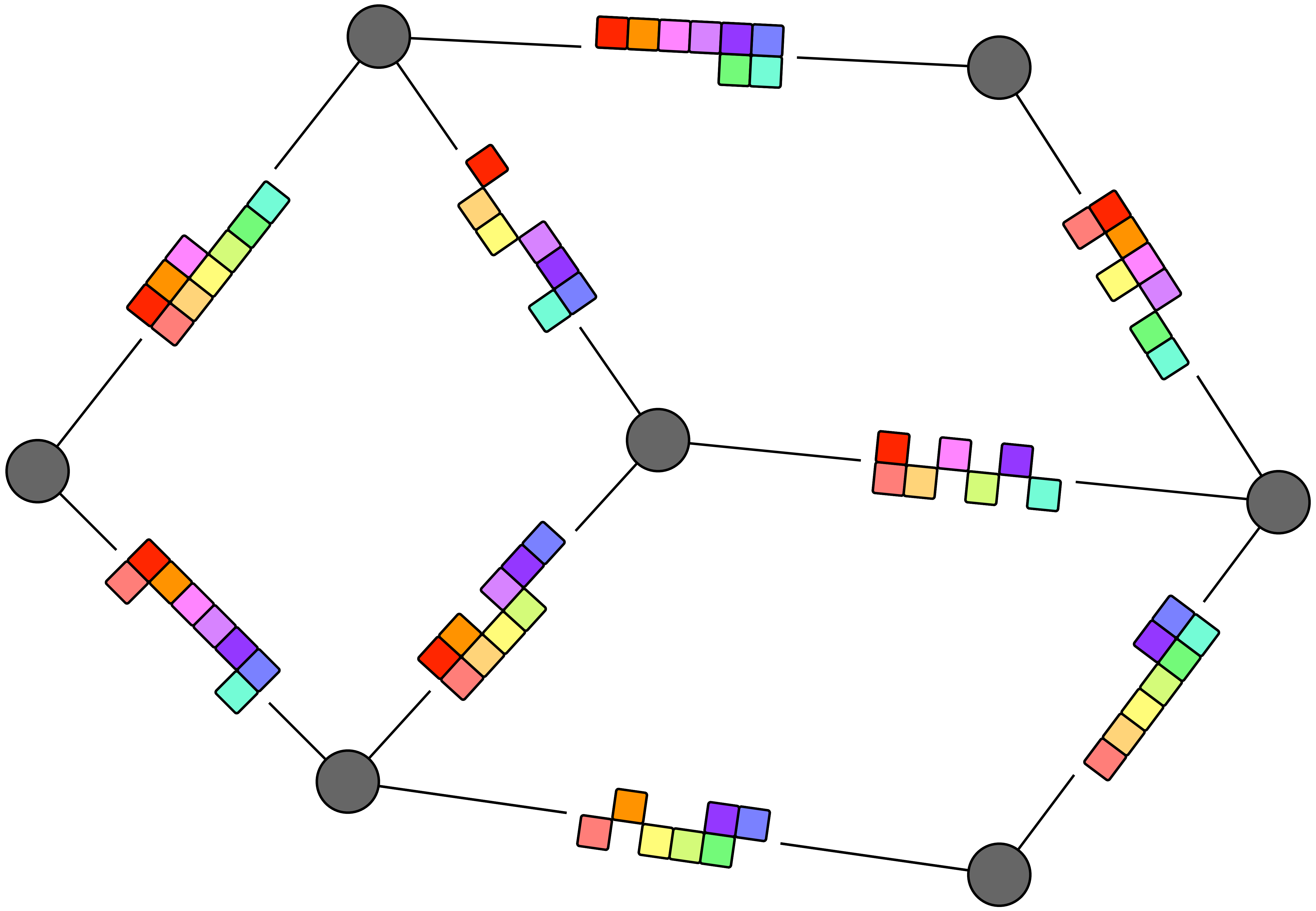
$$T(1, \mathbf{C}) \leq \beta^2 \cdot \log \Delta \cdot T(\beta, \mathbf{C})$$

$$T(\beta, \mathbf{C}) \leq \log p \cdot T(1, \mathbf{p}) + T(\beta/\text{polylog } p, \mathbf{C/p})$$

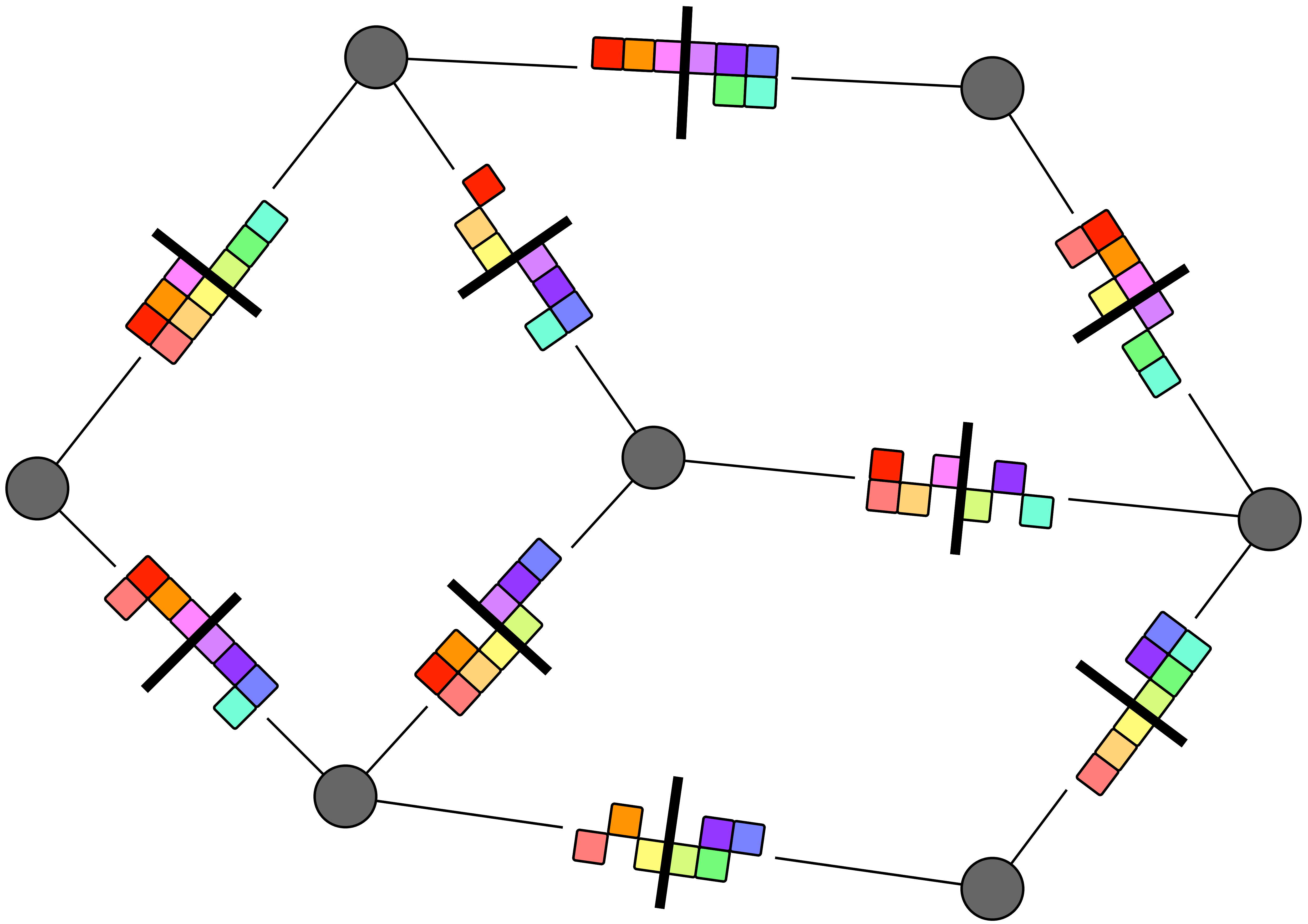
$$T(1, \mathbf{\Delta}) \leq \text{polylog } \Delta \cdot T(1, \mathbf{\sqrt{\Delta}})$$

$$T(1, \mathbf{\Delta}) \leq (\log \Delta)^{O(\log \log \Delta)} \cdot T(1, \mathbf{O(1)}) = (\log \Delta)^{O(\log \log \Delta)}$$

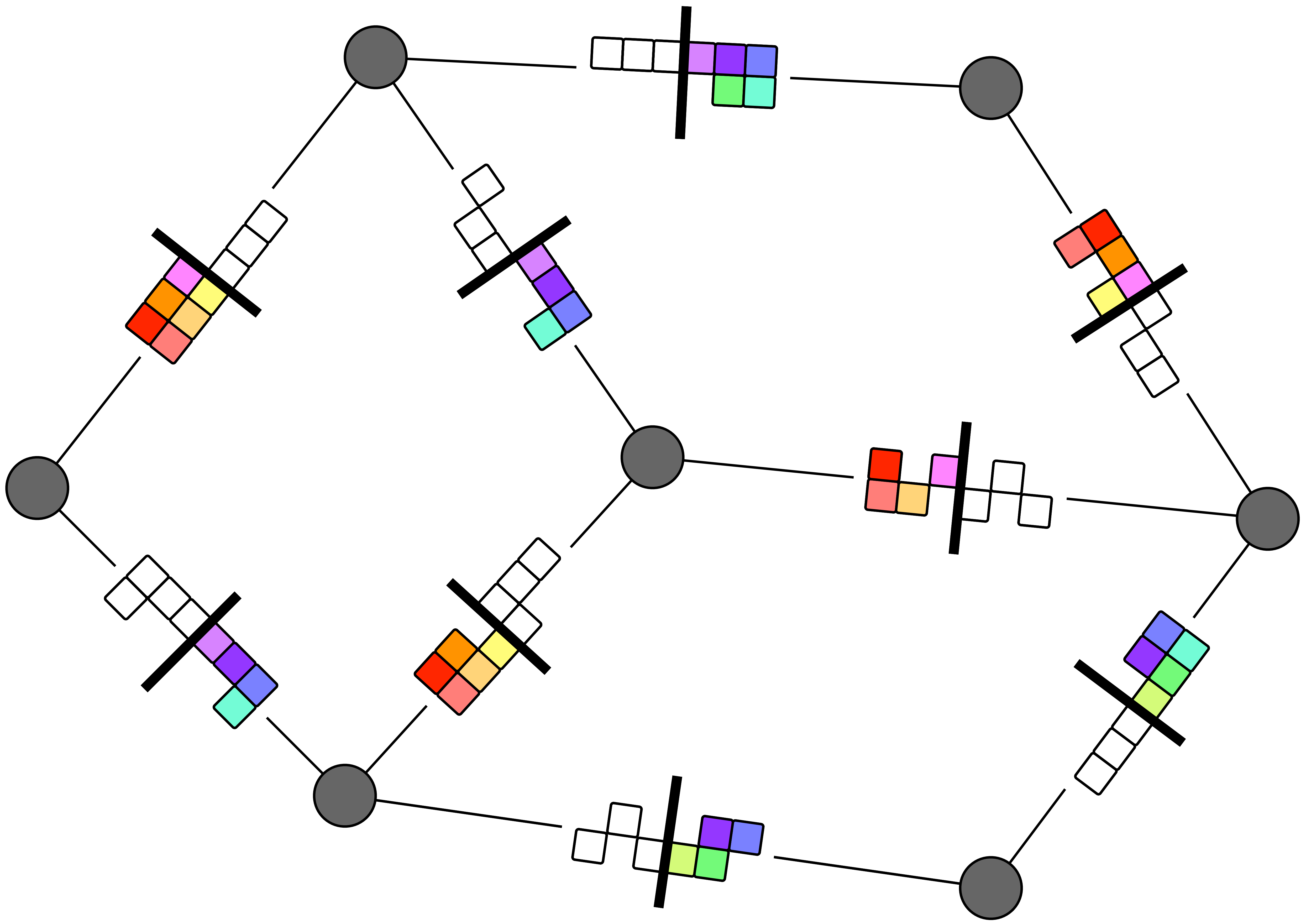
Relaxed list edge coloring



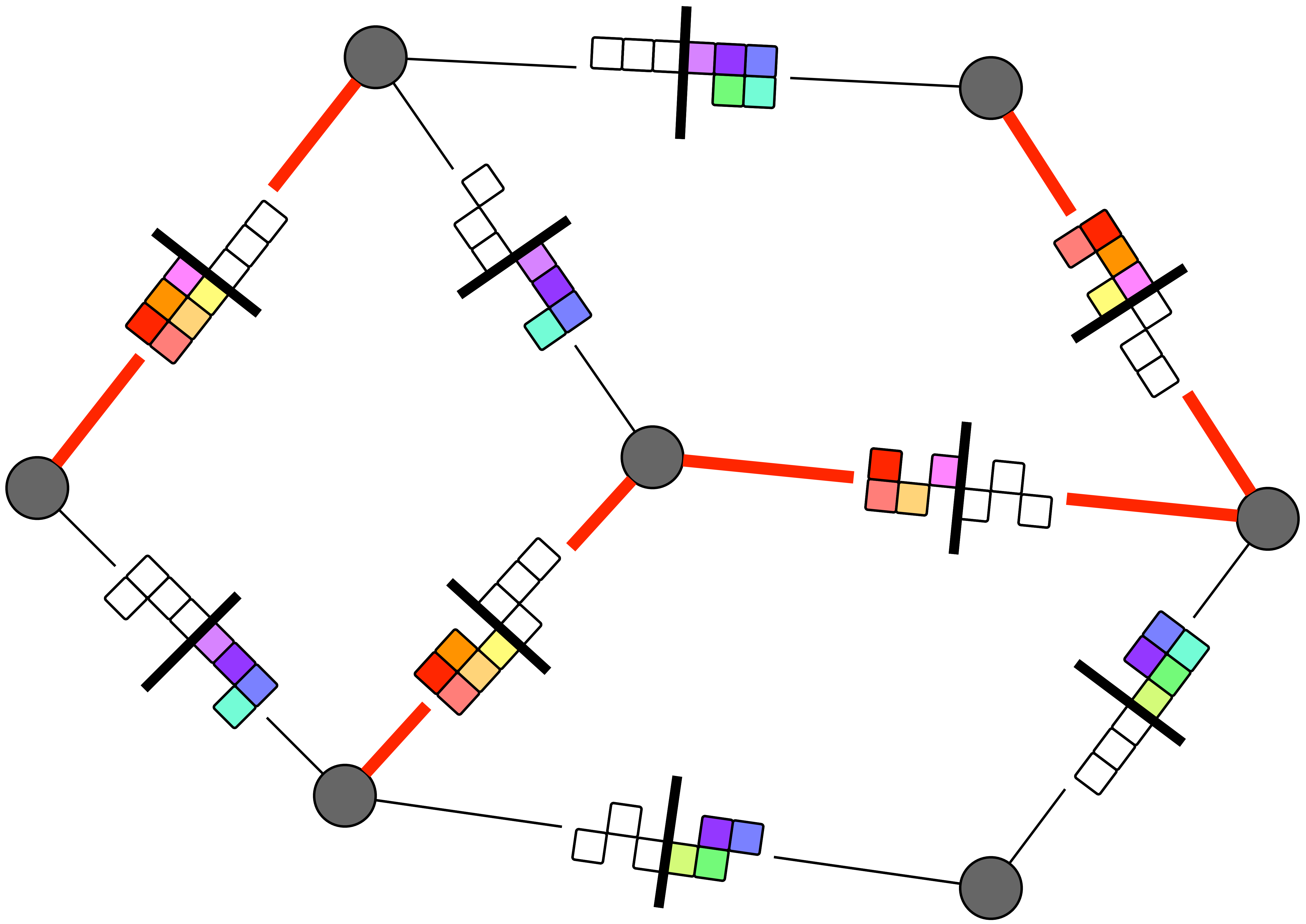
Relaxed list edge coloring



Relaxed list edge coloring



Relaxed list edge coloring



Relaxed list edge coloring

$$T(\beta, \mathbf{C}) \leq \log p \cdot T(1, \mathbf{p}) + T(\beta/\text{polylog } p, \mathbf{C/p})$$

Relaxed list edge coloring

$$T(\beta, \mathbf{C}) \leq \log p \cdot T(1, \mathbf{p}) + T(\beta/\text{polylog } p, \mathbf{C/p})$$

- Split the color space into many independent subspaces

Relaxed list edge coloring

$$T(\beta, \mathbf{C}) \leq \log p \cdot T(1, \mathbf{p}) + T(\beta/\text{polylog } p, \mathbf{C/p})$$

- Split the color space into many independent subspaces
- Assign a subspace to each edge

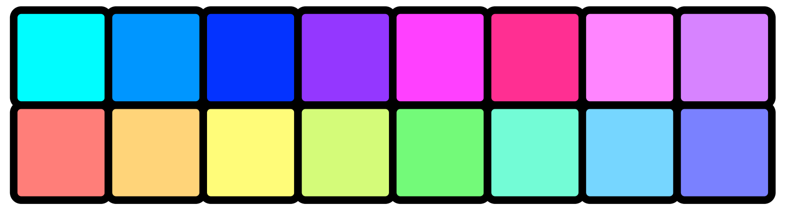
Relaxed list edge coloring

$$T(\beta, \mathbf{C}) \le \log p \cdot T(1, \mathbf{p}) + T(\beta/\text{polylog } p, \mathbf{C/p})$$

- Split the color space into many independent subspaces
- Assign a subspace to each edge
- Independently recurse on each graph induced by edges with the same assigned subspace

Relaxed list edge coloring

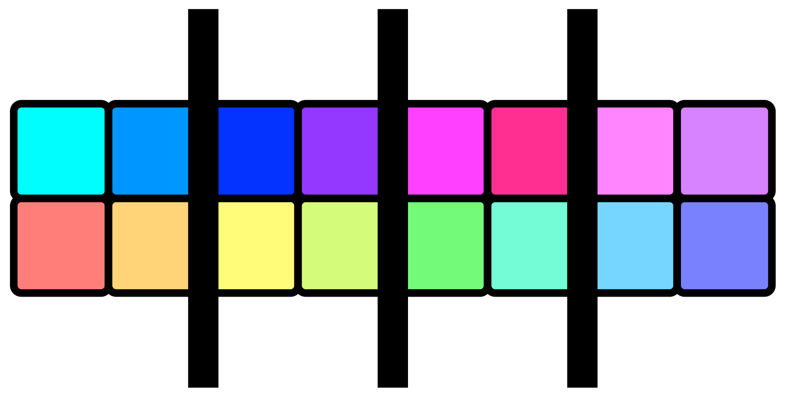
$$T(\beta, \mathbf{C}) \leq \log p \cdot T(1, \mathbf{p}) + T(\beta/\text{polylog } p, \mathbf{C/p})$$



- Split the color space into many independent subspaces
- Assign a subspace to each edge
- Independently recurse on each graph induced by edges with the same assigned subspace

Relaxed list edge coloring

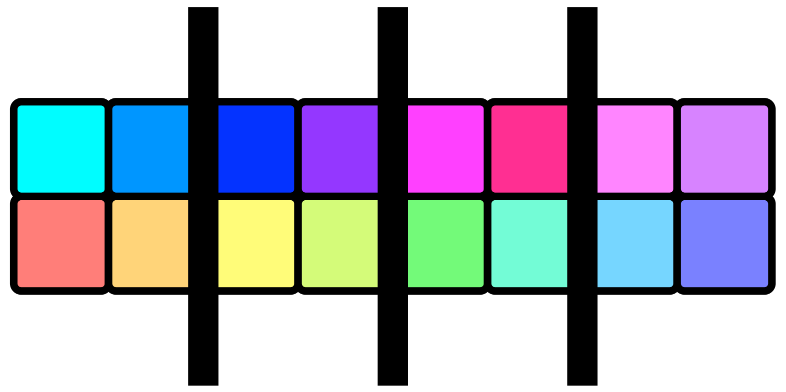
$$T(\beta, \mathbf{C}) \leq \log p \cdot T(1, \mathbf{p}) + T(\beta/\text{polylog } p, \mathbf{C/p})$$



- Split the color space into many independent subspaces
- Assign a subspace to each edge
- Independently recurse on each graph induced by edges with the same assigned subspace

Relaxed list edge coloring

$$T(\beta, \mathbf{C}) \le \log p \cdot T(1, \mathbf{p}) + T(\beta/\text{polylog } p, \mathbf{C/p})$$



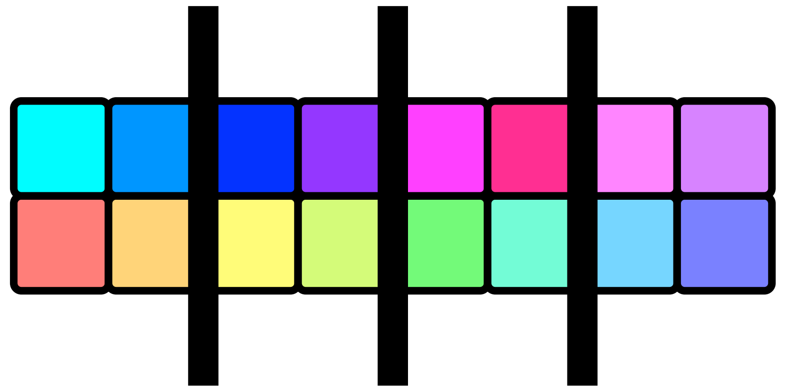
- Split the color space into many independent subspaces
- Assign a subspace to each edge

$\log p \cdot T(1, \mathbf{p})$

- Independently recurse on each graph induced by edges with the same assigned subspace

Relaxed list edge coloring

$$T(\beta, \mathbf{C}) \le \log p \cdot T(1, \mathbf{p}) + T(\beta/\mathrm{polylog}\ p, \mathbf{C/p})$$



- Split the color space into many independent subspaces
- Assign a subspace to each edge — $\log p \cdot T(1, \mathbf{p})$
- Independently recurse on each graph induced by edges with the same assigned subspace — $T(\beta/\mathrm{polylog}\ p, \mathbf{C/p})$

Relaxed list edge coloring

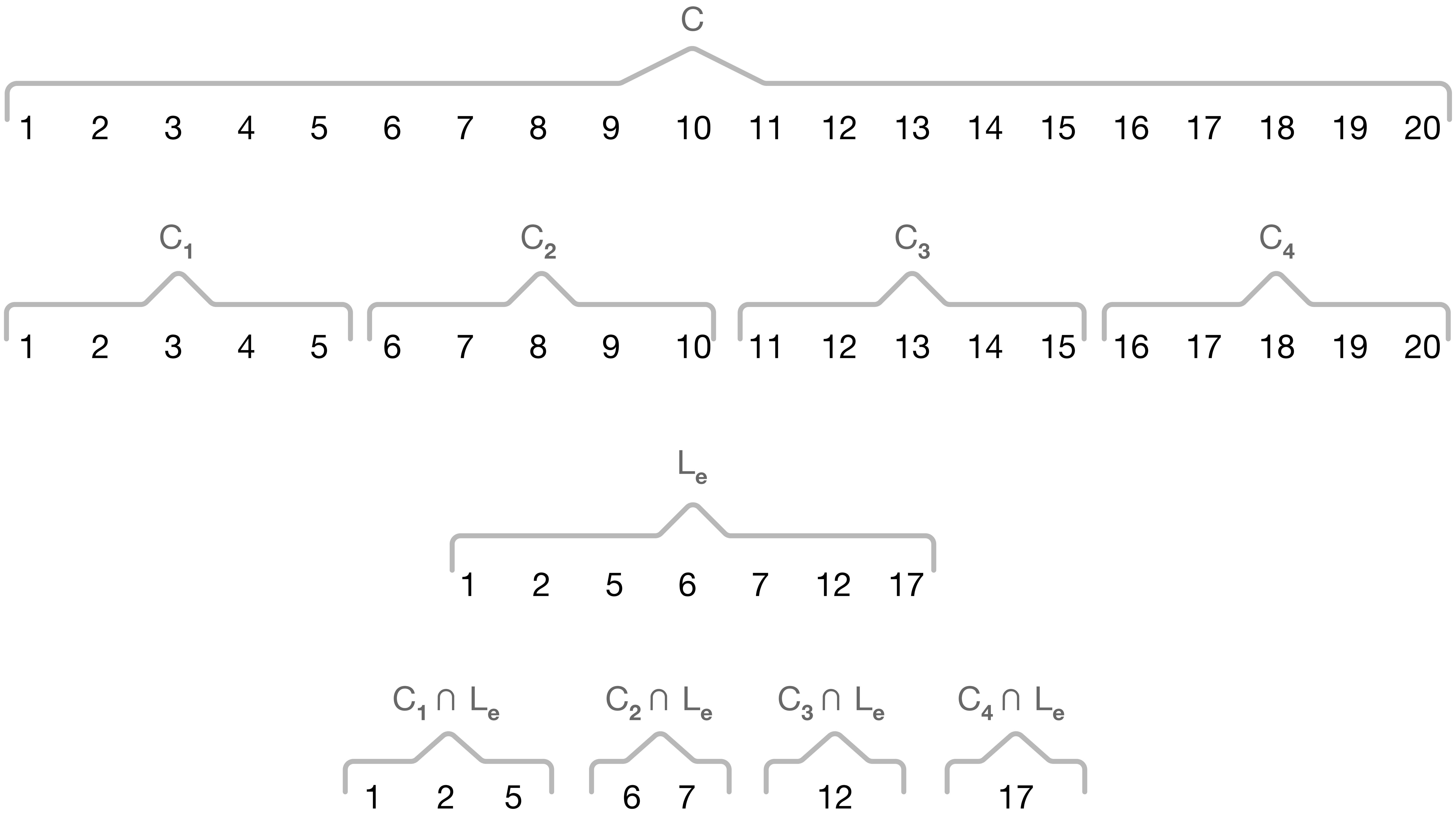
$$T(\beta, \mathbf{C}) \leq \log p \cdot T(1, \mathbf{p}) + T(\beta/\text{polylog } p, \mathbf{C/p})$$

- Split the color space into many independent subspaces
- Assign a subspace to each edge
- Independently recurse on each graph induced by edges with the same assigned subspace

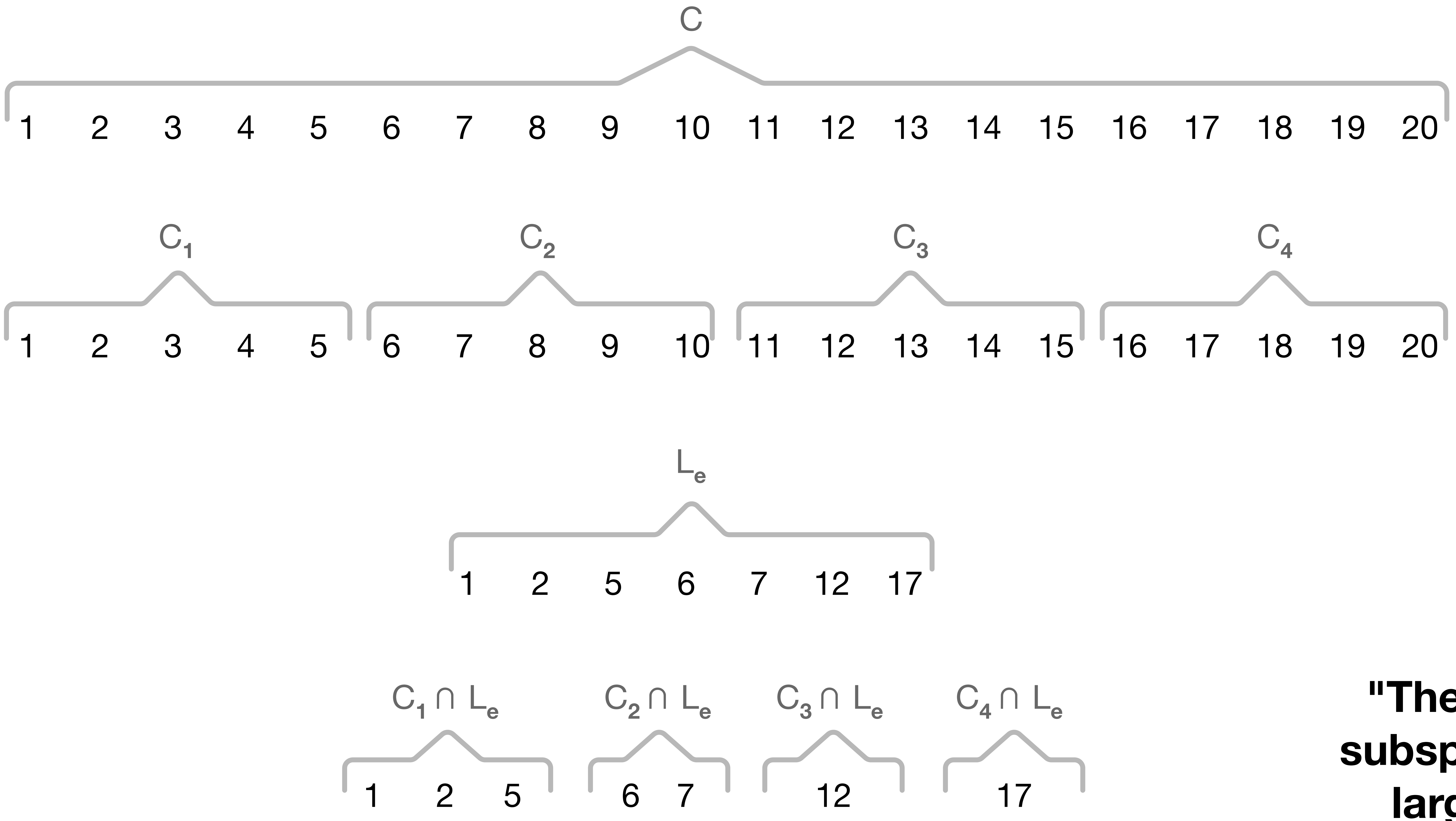
$\log p \cdot T(1, \mathbf{p})$

$T(\beta/\text{polylog } p, \mathbf{C/p})$

Subspace assignment

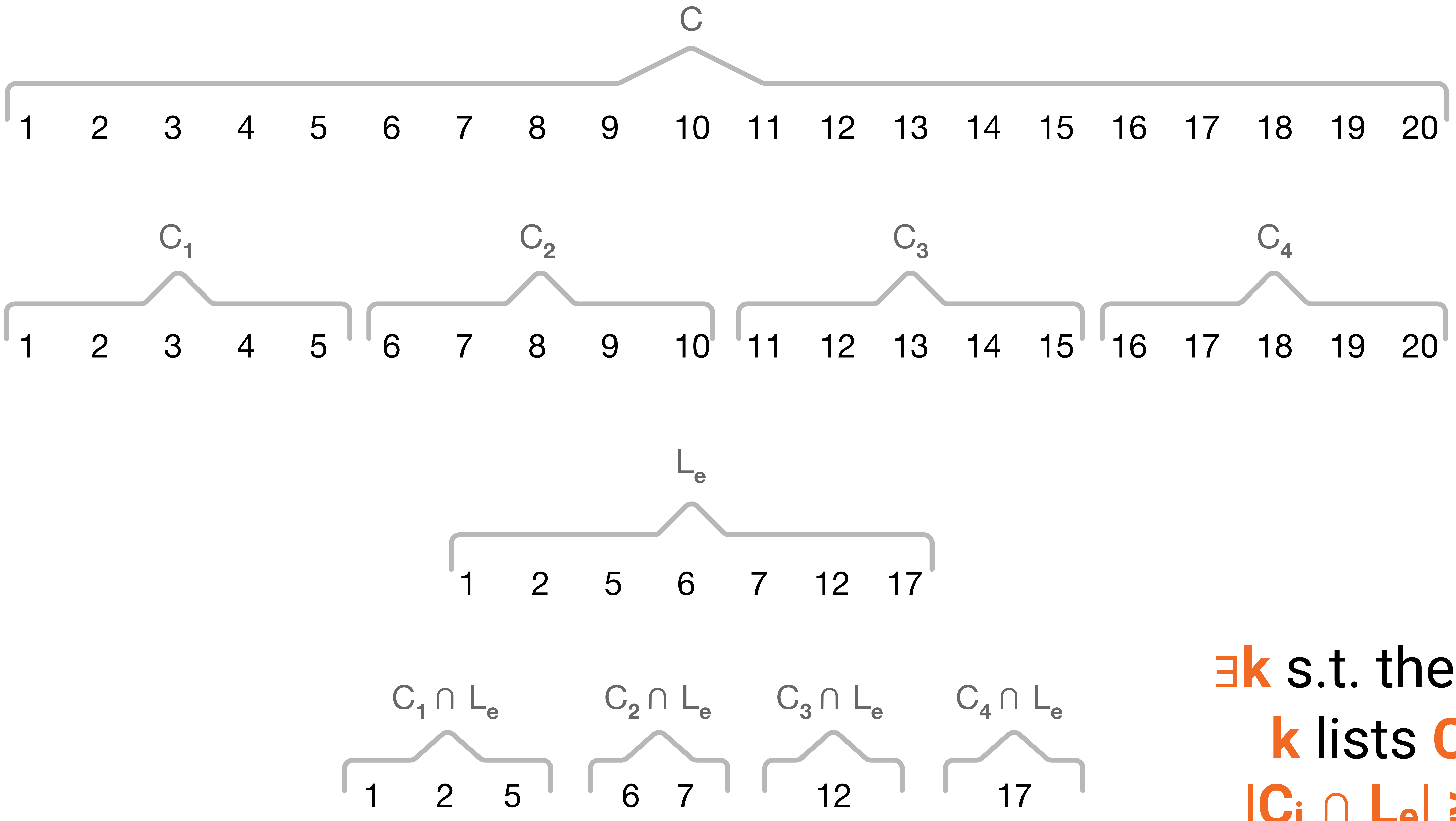


Subspace assignment



"There are many subspaces that are large enough"

Subspace assignment



$\exists k$ s.t. there are at least k lists C_i satisfying $|C_i \cap L_e| \geq |L_e| / (k\, H_p)$

Subspace assignment

Subspace assignment

$\exists k$ s.t. there are at least k lists C_i satisfying $|C_i \cap L_e| \geq |L_e| / (k H_p)$

Subspace assignment

$\exists$**k** s.t. there are at least **k** lists C_i satisfying $|C_i \cap L_e| \geq |L_e| / (k\, H_p)$

Subspace assignment

$\exists$**k** s.t. there are at least **k** lists C_i satisfying $|C_i \cap L_e| \geq |L_e| / (k H_p)$

Simple case:

Subspace assignment

$\exists$**k** s.t. there are at least **k** lists C_i satisfying $|C_i \cap L_e| \geq |L_e| / (k H_p)$

Simple case:

- **k** is the **same** for all edges

Subspace assignment

$\exists k$ s.t. there are at least k lists C_i satisfying $|C_i \cap L_e| \geq |L_e| / (k H_p)$

Simple case:

- k is the **same** for all edges

Goal:

Subspace assignment

$\exists$**k** s.t. there are at least **k** lists C_i satisfying $|C_i \cap L_e| \geq |L_e| / (k H_p)$

Simple case:

- **k** is the **same** for all edges

Goal:

- assign a list to each edge such that "few" neighboring edges have the same list

Subspace assignment

Subspace assignment

How:

Subspace assignment

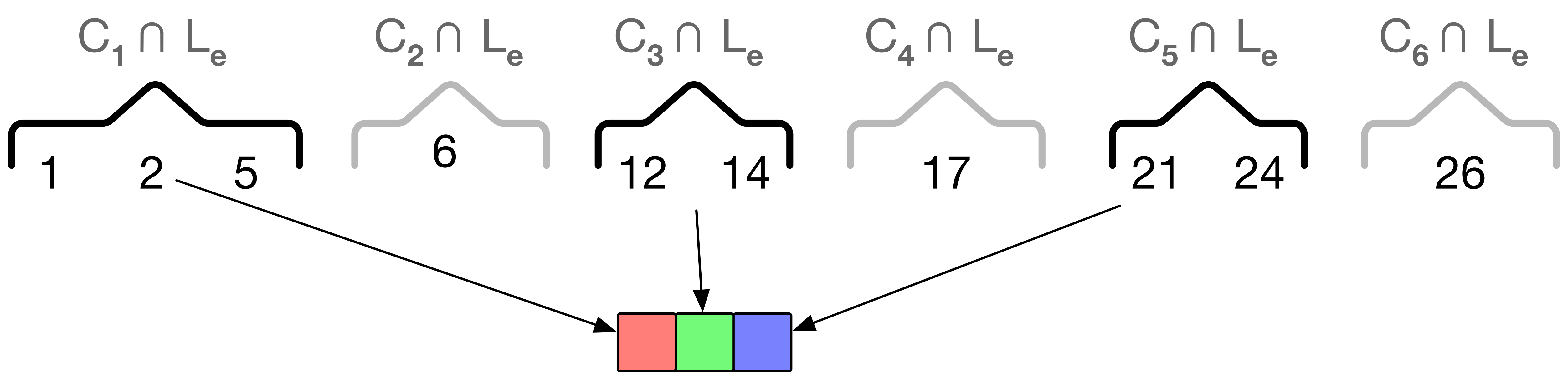
How:

- Transform this problem into a list coloring instance

Subspace assignment

How:

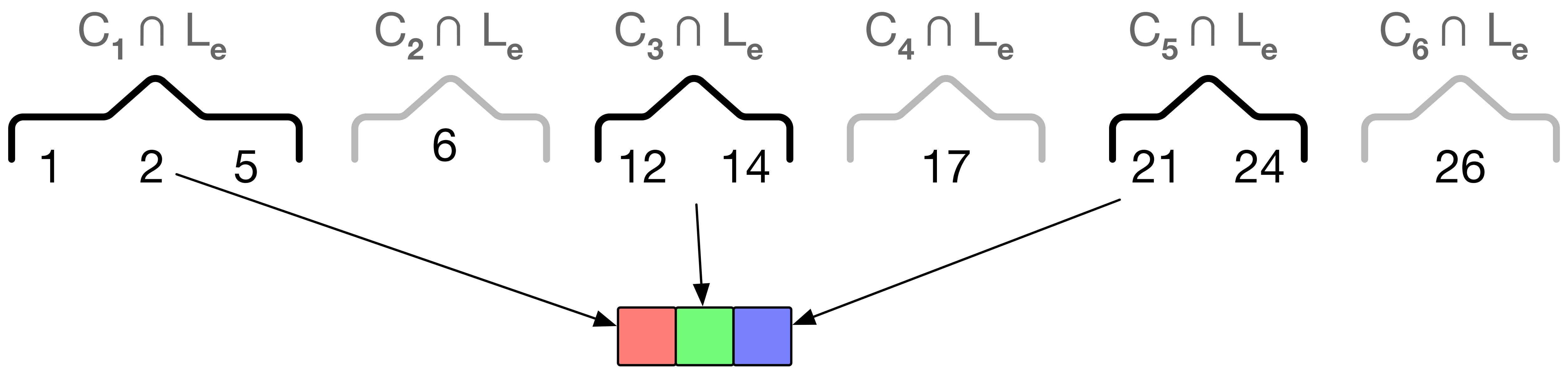
- Transform this problem into a list coloring instance



Subspace assignment

How:

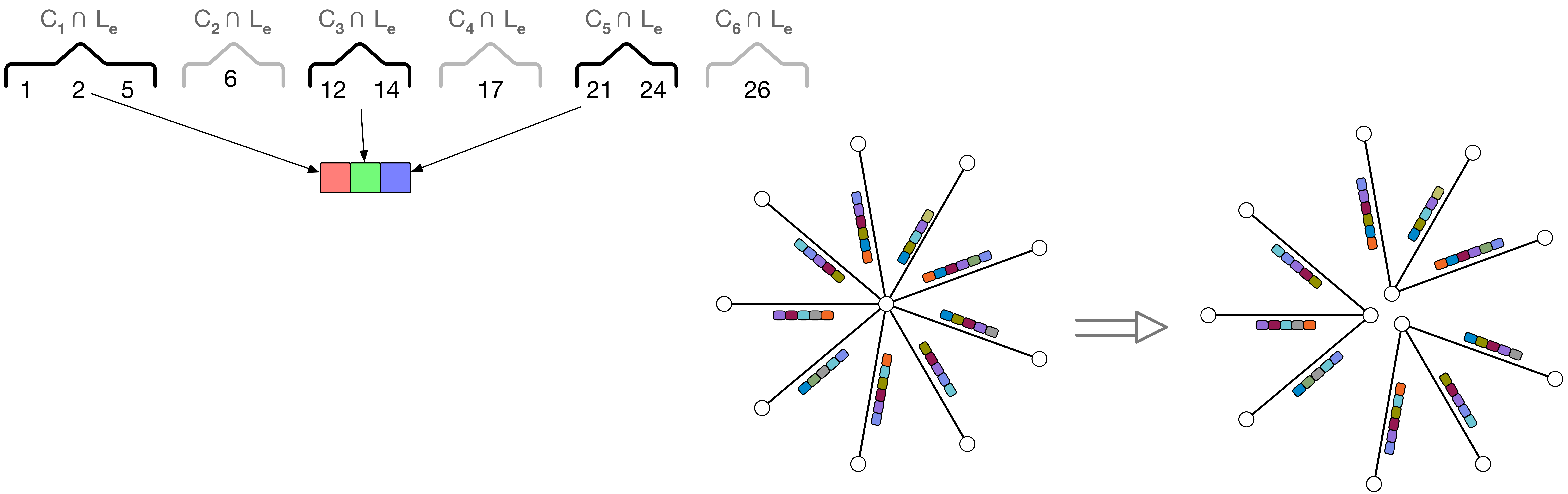
- Transform this problem into a list coloring instance
- Modify the graph such that the edge degree is at most **k-1**



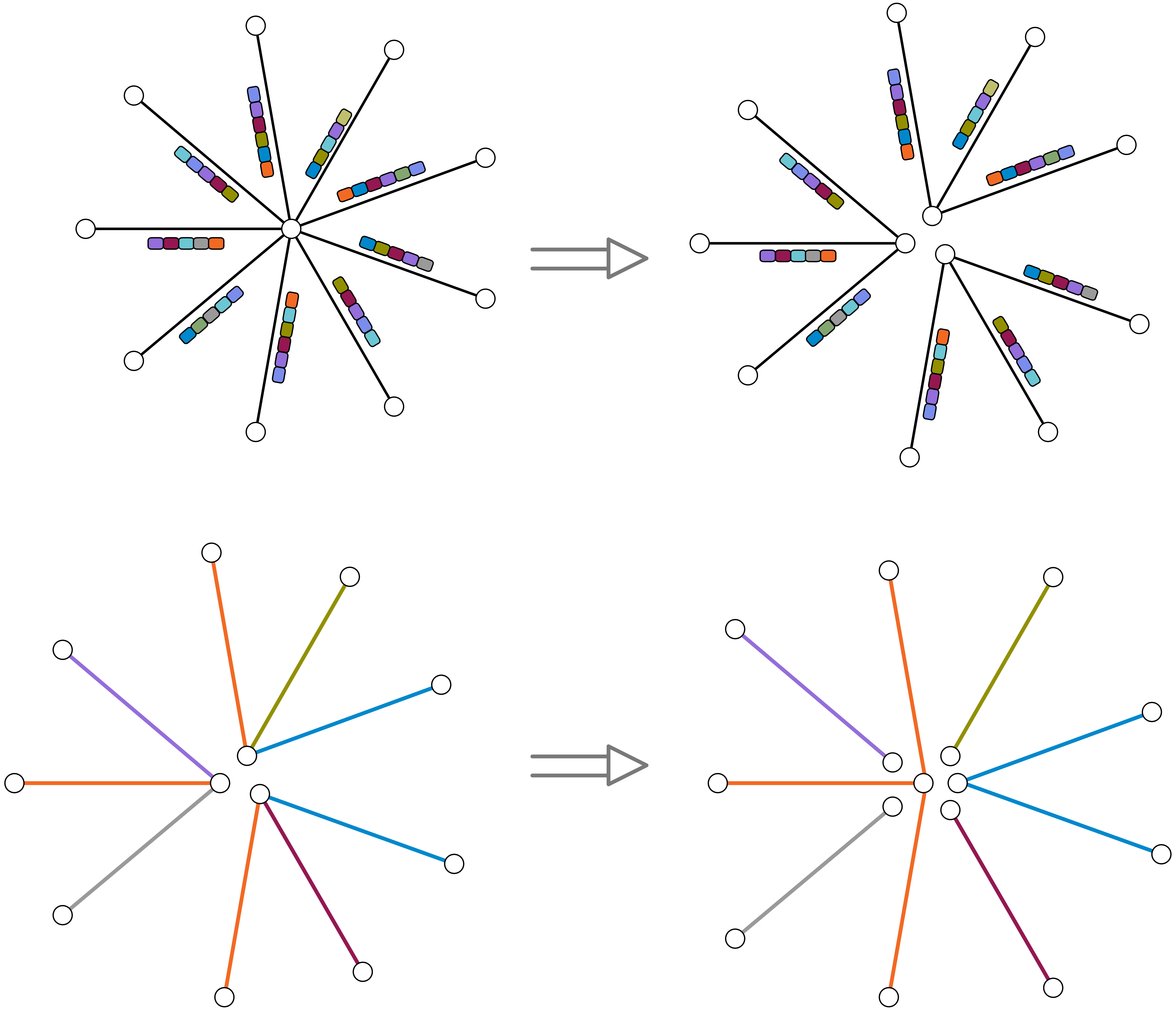
Subspace assignment

How:

- Transform this problem into a list coloring instance
- Modify the graph such that the edge degree is at most **k-1**



Subspace assignment



Subspace assignment

$\exists k$ s.t. there are at least k lists C_i satisfying $|C_i \cap L_e| \geq |L_e| / (k\, H_p)$

Different edges may have different k. Solution:

- Split edges in **buckets** with "similar" values of k
- Solve each bucket **sequentially** as in the simple case

k = 1 | 2 | 3 4 | 5 6 7 8 | 9 10 11 12 13 14 15 16 | 17 …

Subspace assignment

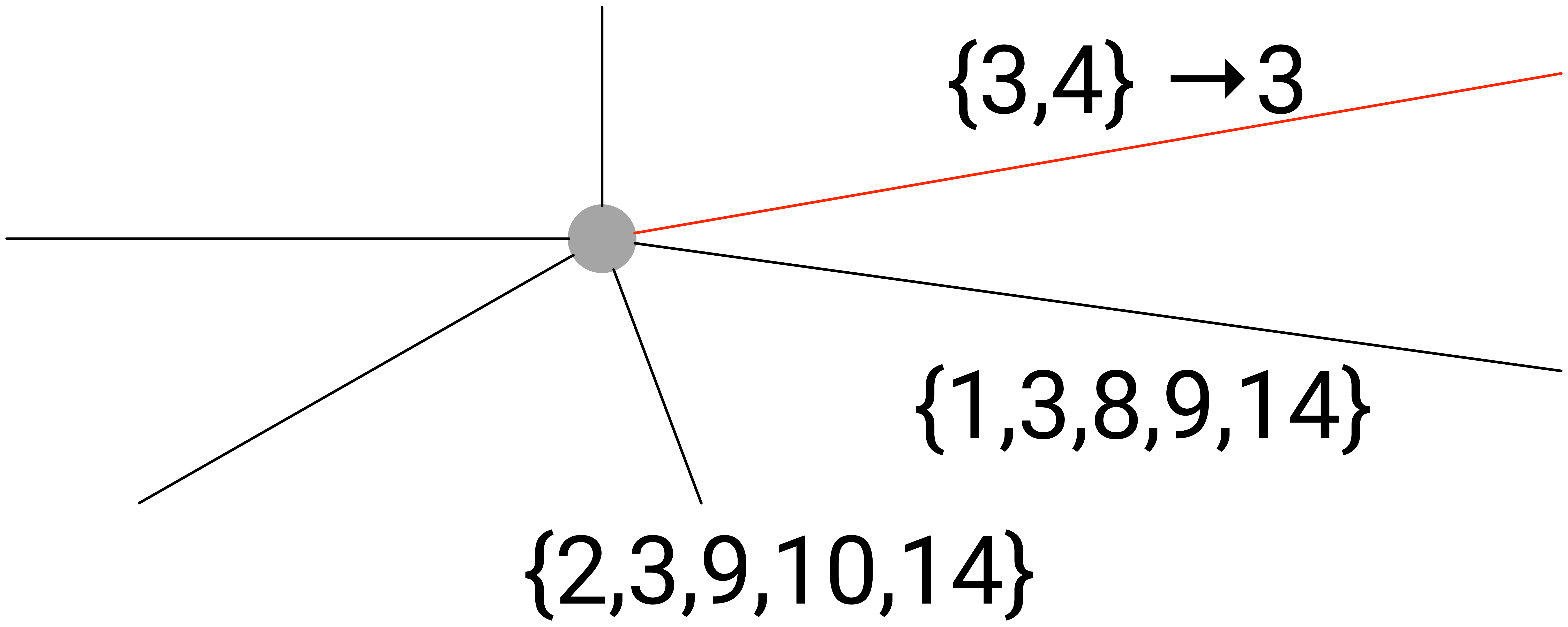
$\exists k$ s.t. there are at least k lists C_i satisfying $|C_i \cap L_e| \geq |L_e| / (k H_p)$

Different edges may have different k. Solution:

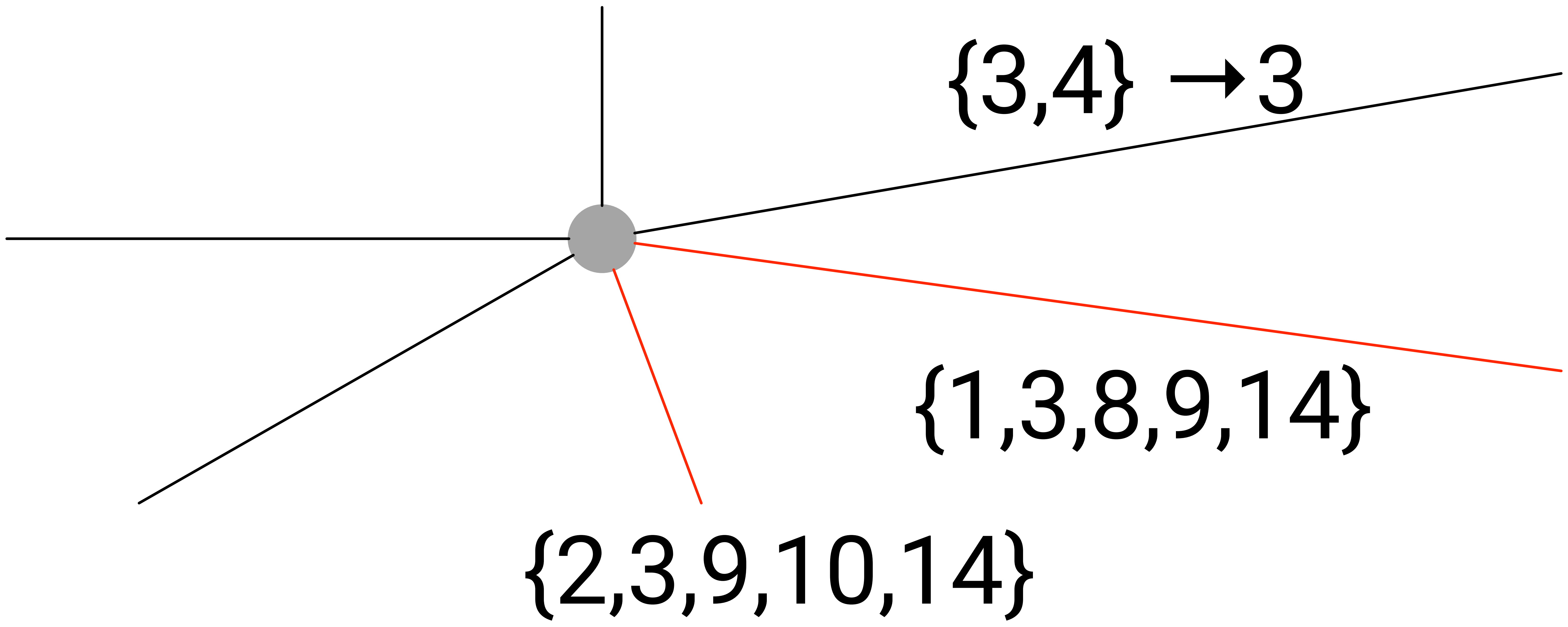
- Split edges in **buckets** with "similar" values of k
- Solve each bucket **sequentially** as in the simple case

k = 1 | 2 | 3 4 | 5 6 7 8 | 9 10 11 12 13 14 15 16 | 17 …

Subspace assignment



Subspace assignment



Subspace assignment

Solution:

- Make some edges inactive, based on their current defect
- More recursion!

Relaxed list edge coloring

$$T(\beta, \mathbf{C}) \leq \log p \cdot T(1, \mathbf{p}) + T(\beta/\text{polylog } p, \mathbf{C/p})$$

- Split the color space into many independent subspaces
- Assign a subspace to each edge

  $\log p \cdot T(1, \mathbf{p})$
- Independently recurse on each graph induced by edges with the same assigned subspace

  $T(\beta/\text{polylog } p, \mathbf{C/p})$

Putting things together

- Express (degree(e) + 1)-list edge coloring **as a function of** relaxed list edge coloring
  - Create many instances with smaller degree
  - Handle instances sequentially
  - Recurse
- Express relaxed list edge coloring **as a function of** smaller list edge colouring instances
  - Split the color space in many parts
  - Assign subspaces by solving many new list coloring instances
  - Recurse

Open questions: CONGEST model

- Can we adapt this algorithm to work in the CONGEST model?
  - If we assign colors to some edges, we have to remove those colors from the lists of their neighboring edges
  - Nodes incident on the same edge need to agree on the new list
  - Valid colors are the intersection of colors that are good for each side

Open questions: upper bounds

- We can solve $(2\Delta - 1)$-edge coloring in quasi-polylog(Δ) + $O(\log^* n)$
  - Can we solve it in polylog(Δ) + $O(\log^* n)$ rounds?
- Can we solve vertex coloring in subpoly(Δ)?
  - Can we solve $O(\Delta/c)$-defective c-coloring fast?

Open questions: lower bounds

- Can we prove a non-trivial lower bound for solving $(2\Delta - 1)$-edge coloring?
  - Can we show that it cannot be solved in $o(\log \Delta) + O(\log^* n)$?