

CS 5003: Parameterized Algorithms

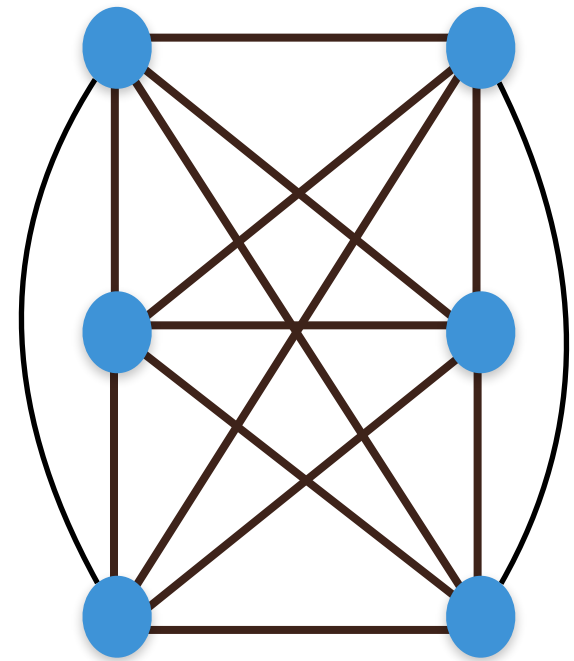
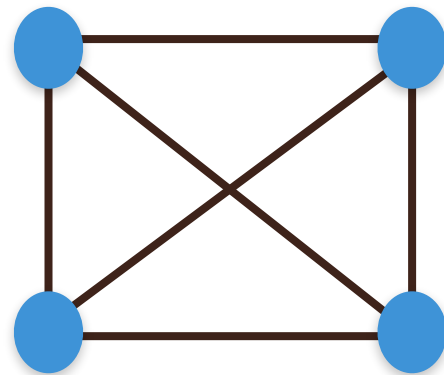
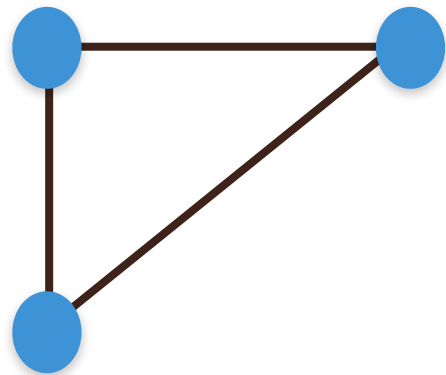
Lectures 24-25

Krithika Ramaswamy

IIT Palakkad

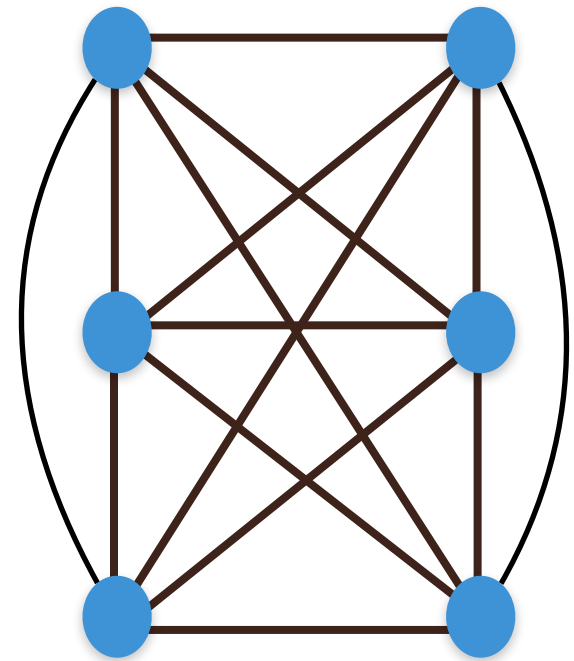
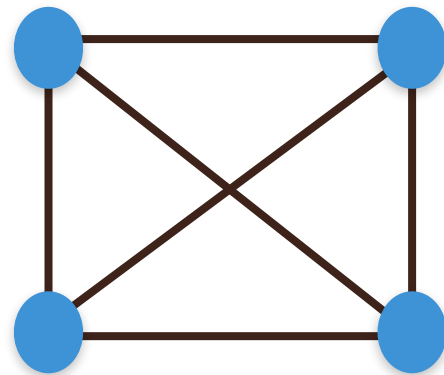
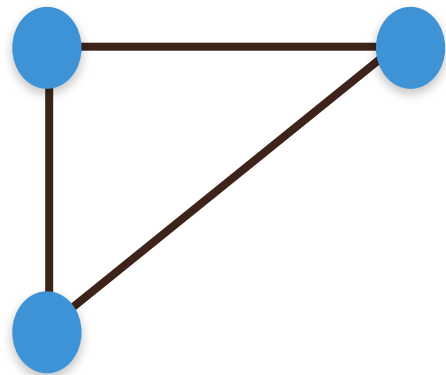
References: Parameterized Algorithms by Cygan et al. and Kernelization by Fomin et al.

Cluster Graphs



A graph in which each component is a complete graph

d-Cluster Graphs



A graph with d components each of which is a complete graph

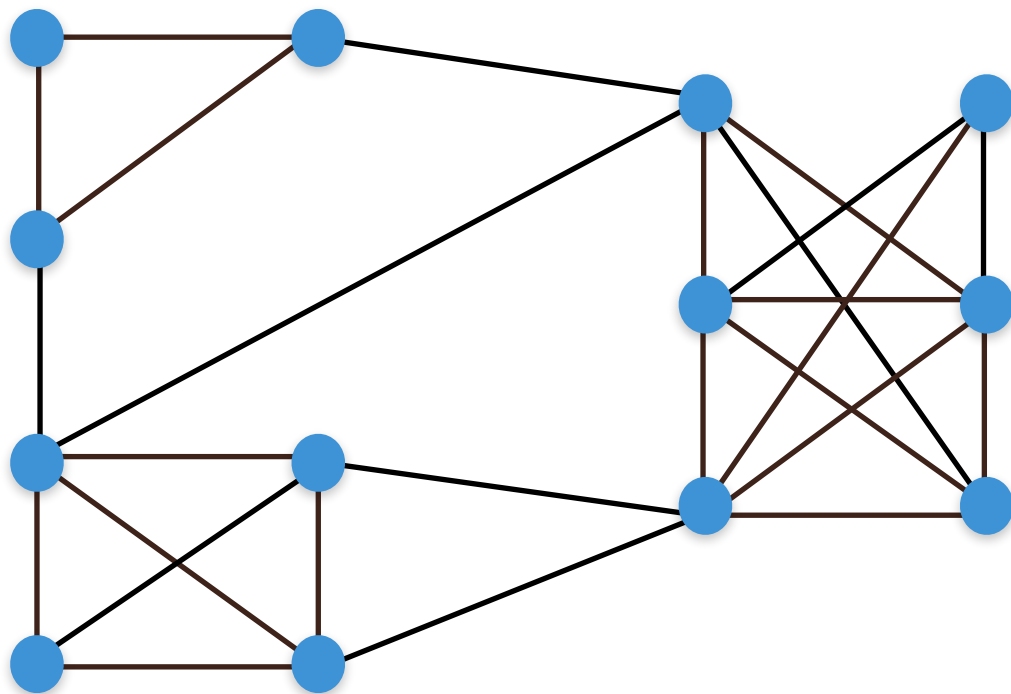
Clustering

Cluster Editing

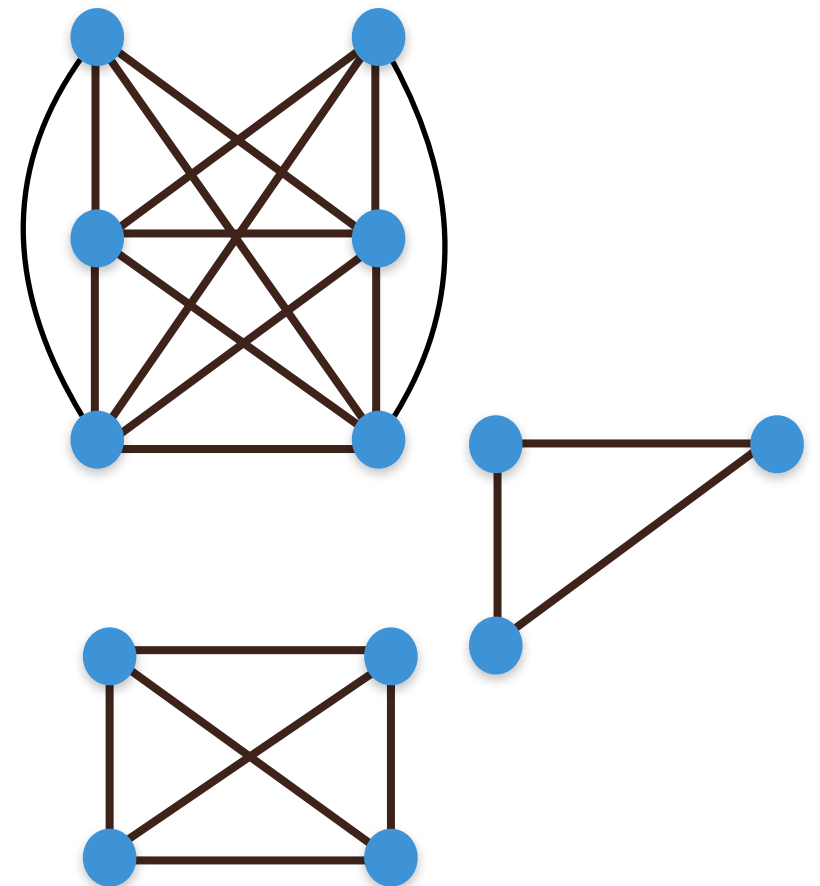
Instance: An undirected graph G and an integer k

Question: Does there exist a set of at most k edge additions and deletions to G so that the resulting graph is a cluster graph?

Parameter: k



$\leq k$ edge modifications



cluster graph

Cluster Editing

Lemma: A graph is a cluster graph iff it has no P_3 as an induced subgraph

Algorithm_Cluster_Editing(G, k)

- * If G is a cluster graph declare that (G, k) is yes-instance
- * If $k \leq 0$ return, otherwise, find a $P_3 = (w, u, v)$
- * **Branch 1:** Delete $\{u, v\}$, decrease k by 1, set $\{u, v\}$ as forbidden. Recurse on resultant instance.
- * **Branch 2:** Delete $\{u, w\}$, decrease k by 1, set $\{u, w\}$ as forbidden, $\{v, w\}$ as forbidden, $\{u, v\}$ as permanent. Recurse on resultant instance.
- * **Branch 3:** Add $\{w, v\}$, decrease k by 1, set $\{u, v\}$, $\{u, w\}$ and $\{v, w\}$ as permanent. Recurse on resultant instance.

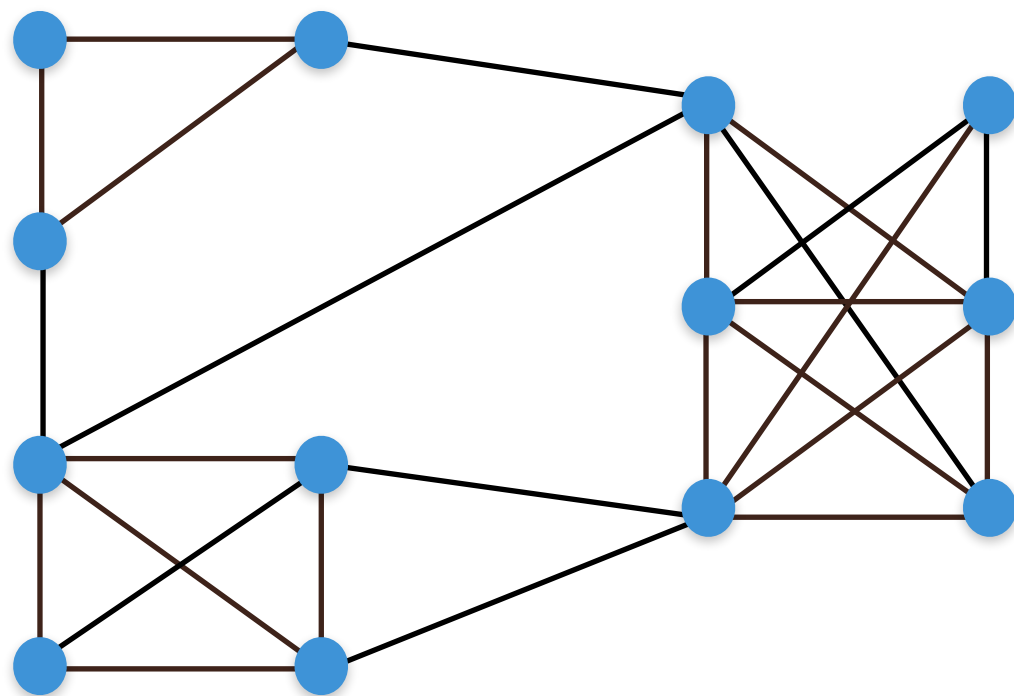
$3^k n^{O(1)}$ algorithm

d-Clustering

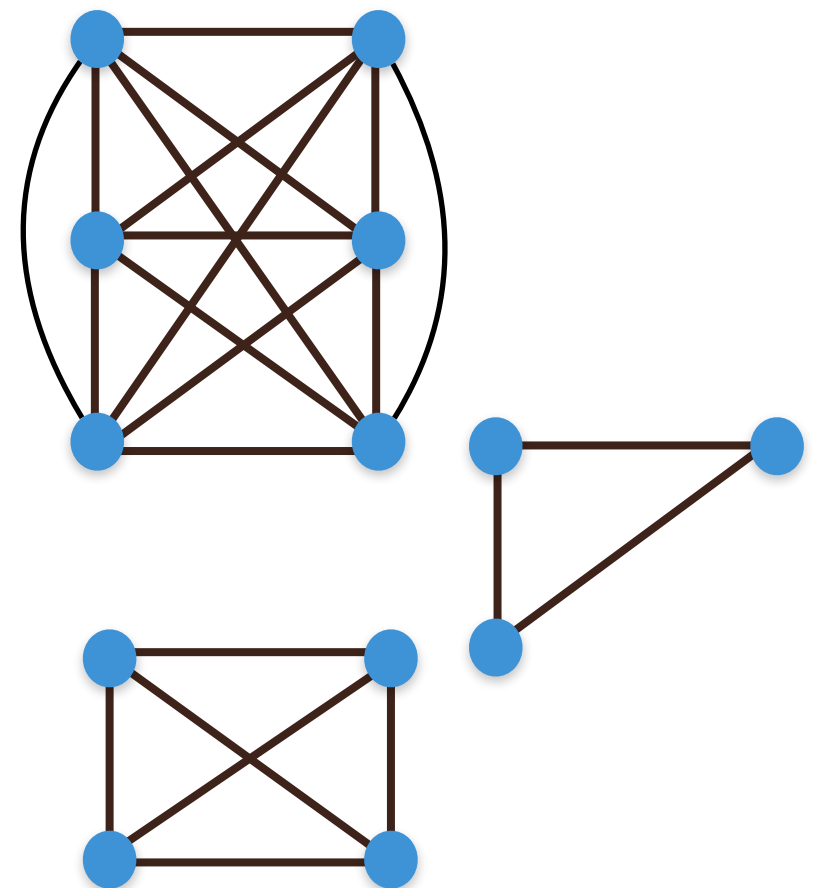
Instance: An undirected graph G and an integer k

Question: Does there exist a set of at most k edge additions and deletions to G so that the resulting graph is a d -cluster graph?

Parameter: k



$\leq k$ edge modifications



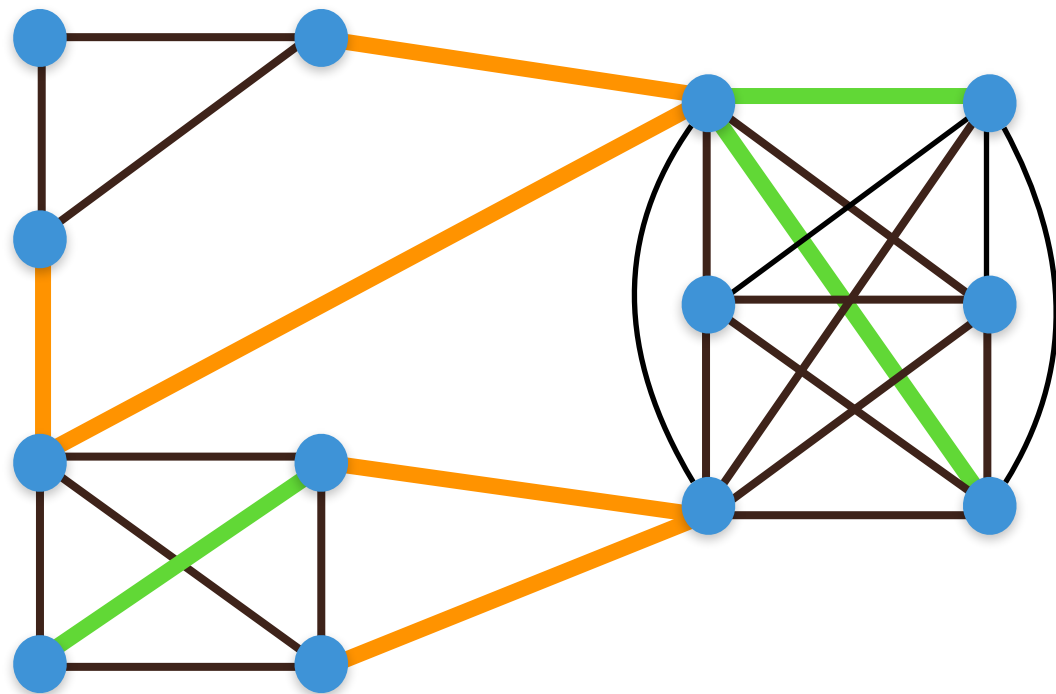
d-cluster graph

3-Clustering

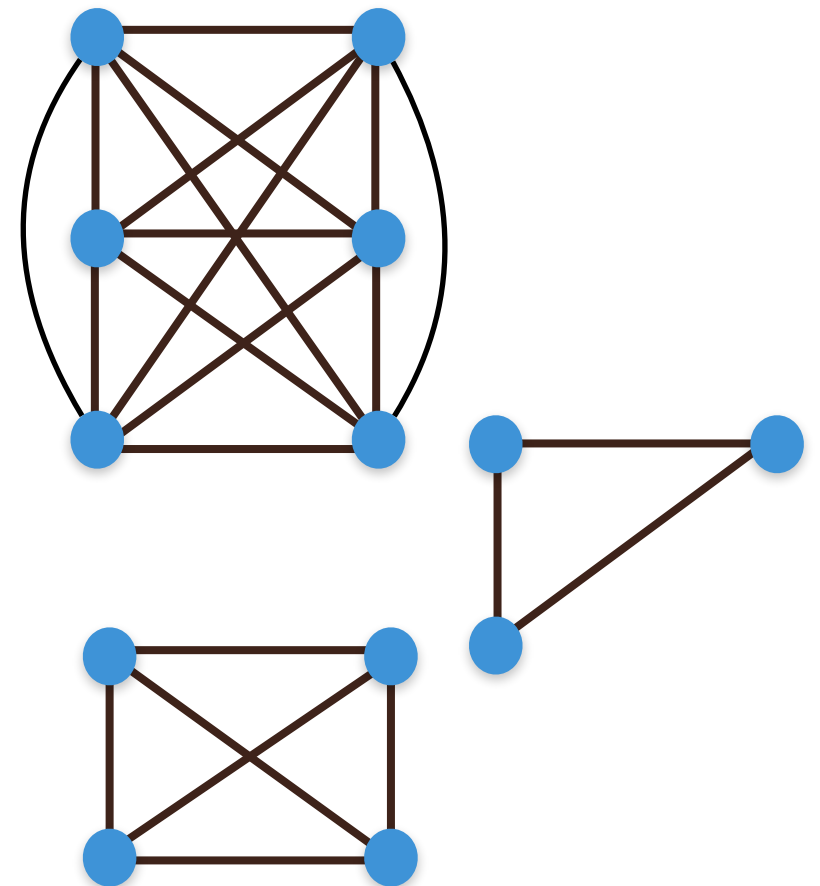
Instance: An undirected graph G and an integer k

Question: Does there exist a set of at most k edge additions and deletions to G so that the resulting graph is a 3-cluster graph?

Parameter: k



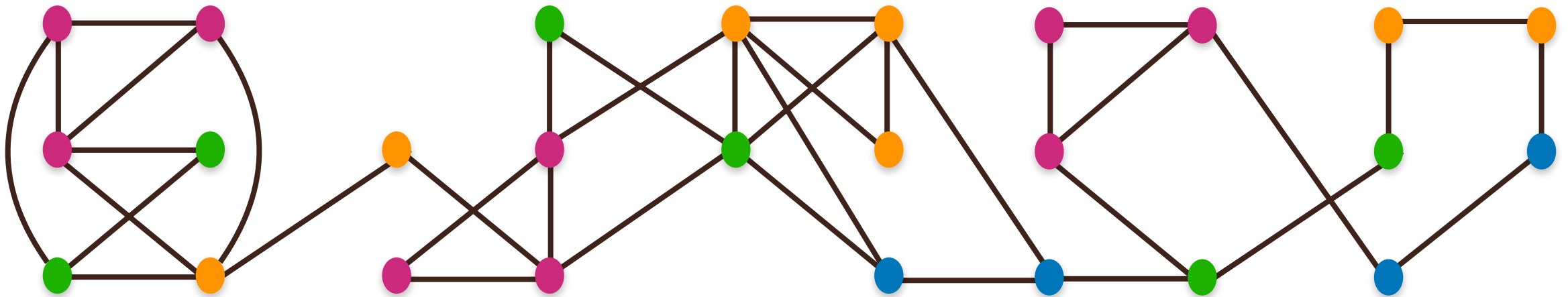
5 deletions + 3 additions



3-cluster graph

Chromatic Coding Algorithm

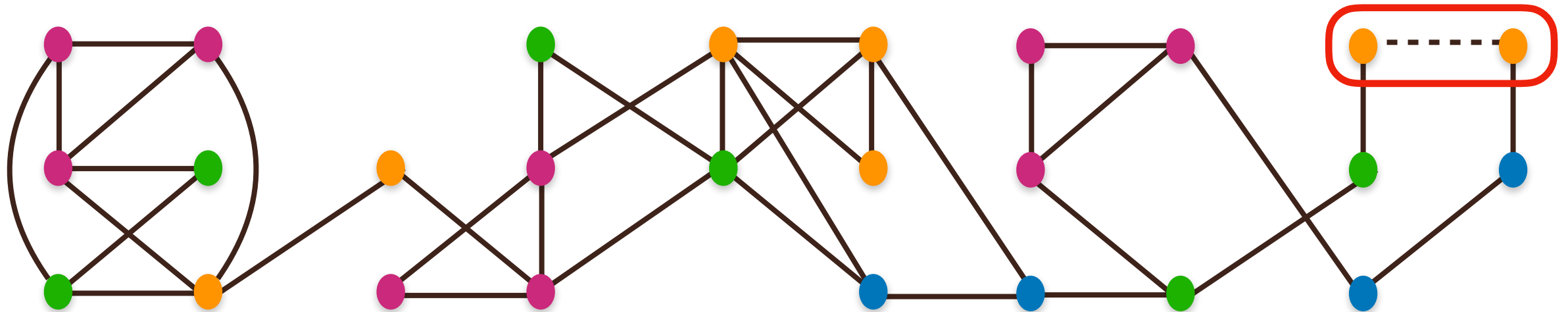
- * Randomly color the vertices of G using $q = \lceil (8k)^{.5} \rceil$ colours



- * Focus on finding a solution in which each edge is properly colored

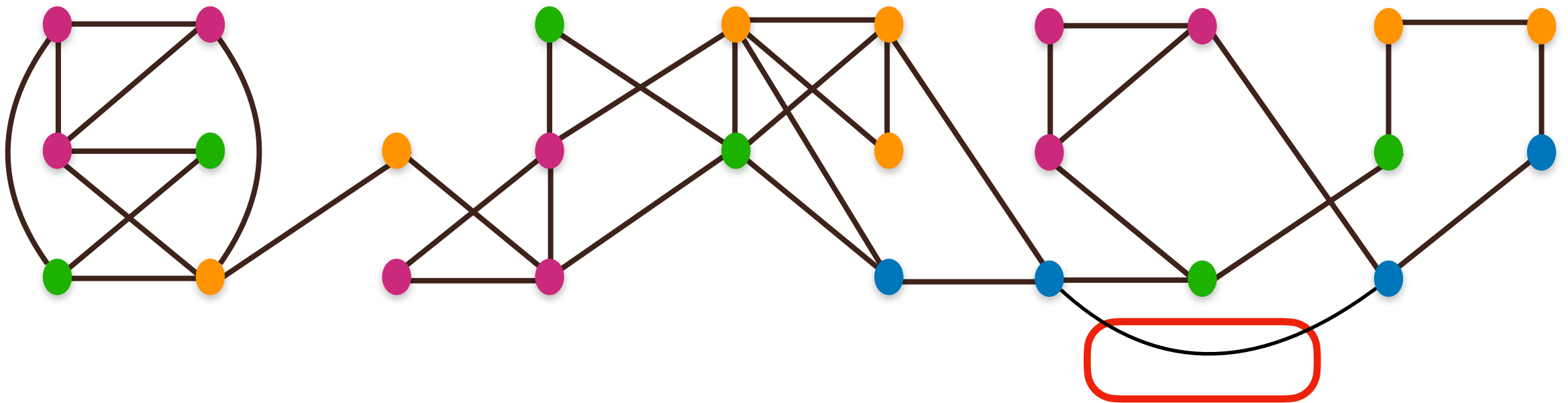
Chromatic Coding Algorithm

- * Randomly color the vertices of G using $q = \lceil (8k)^{1/5} \rceil$ colours
- * Focus on finding a solution in which each edge is properly colored



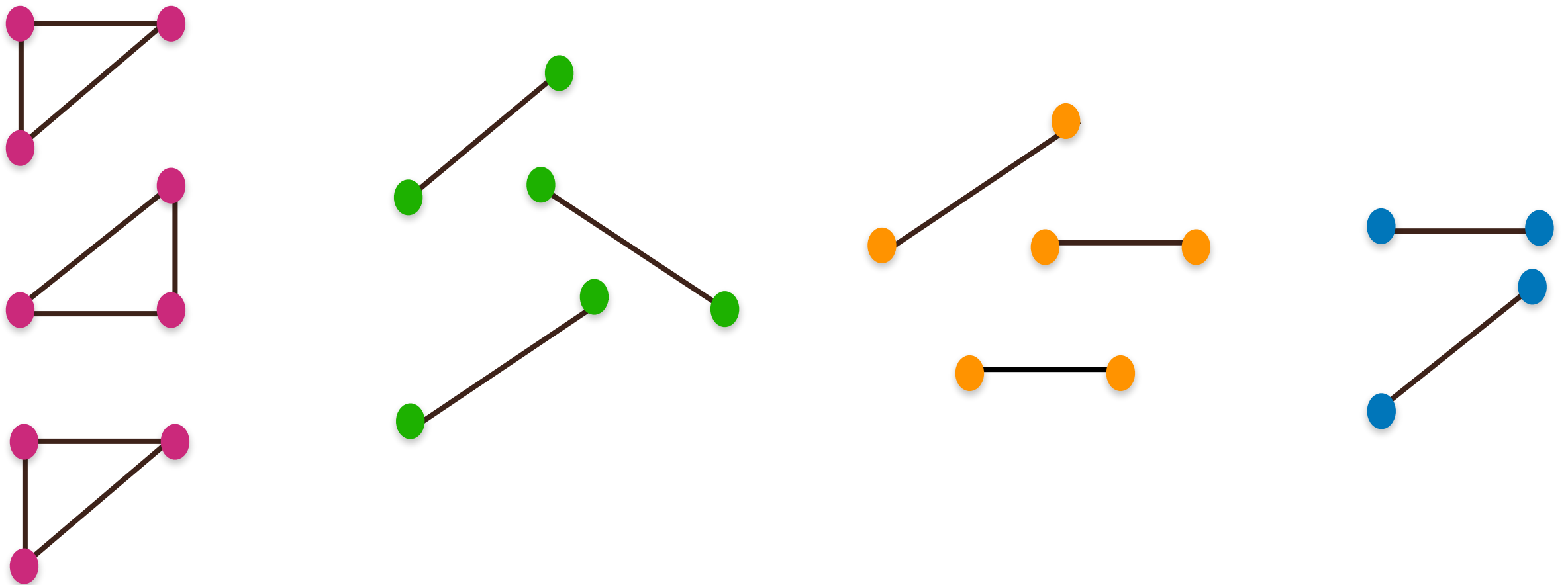
Chromatic Coding Algorithm

- * Randomly color the vertices of G using $q = \lceil (8k)^{.5} \rceil$ colours
- * Focus on finding a solution in which each edge is properly colored



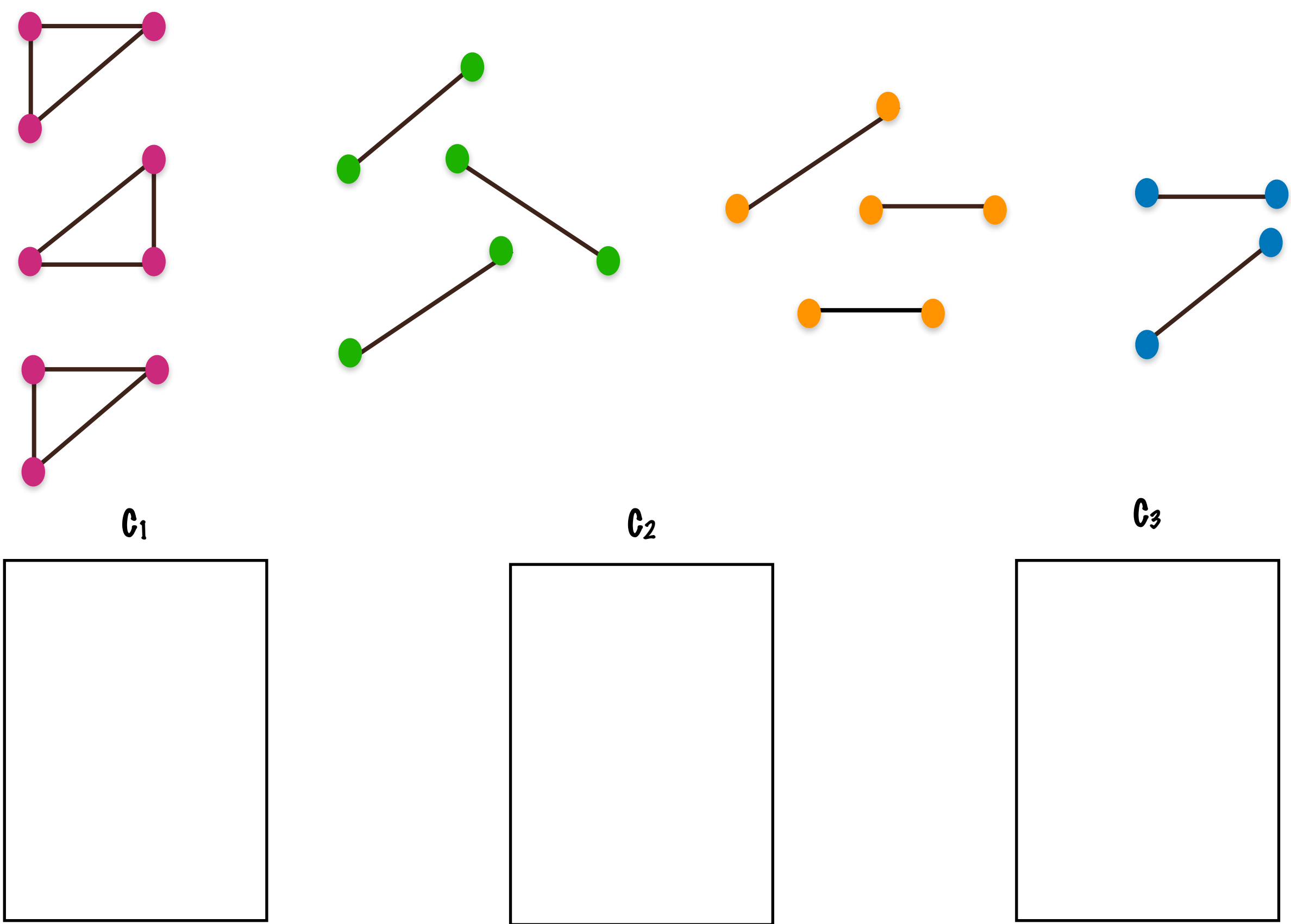
Chromatic Coding Algorithm

- * Randomly color the vertices of G using $q = \lceil (8k)^{.5} \rceil$ colours
- * Focus on finding a solution in which each edge is properly colored
 - * Colorful solution

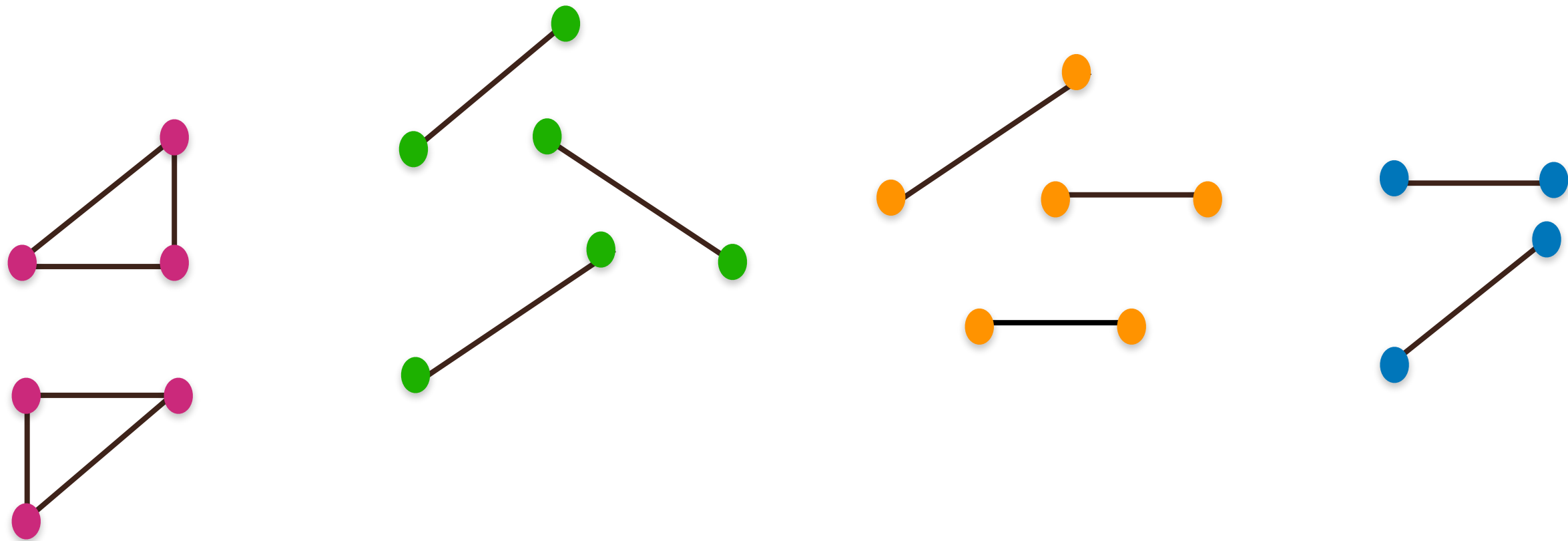


- * Each colour class is a cluster graph with at most d components
 - * Otherwise, no colorful solution

Chromatic Coding Algorithm



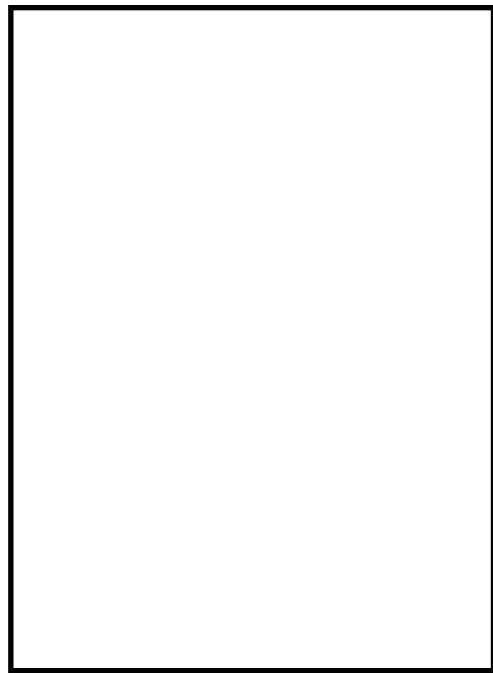
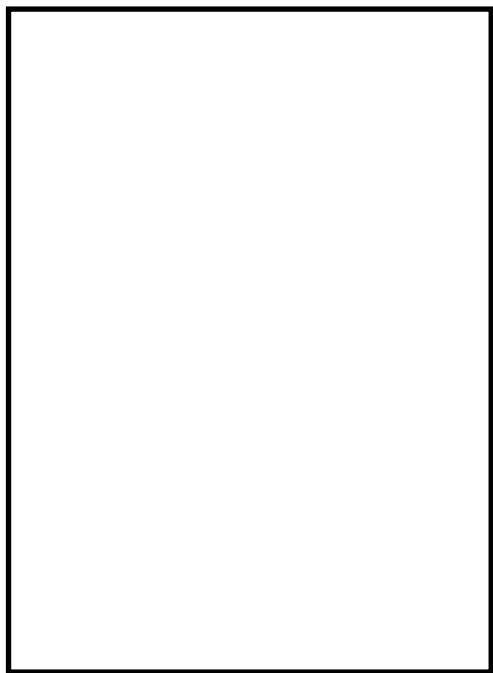
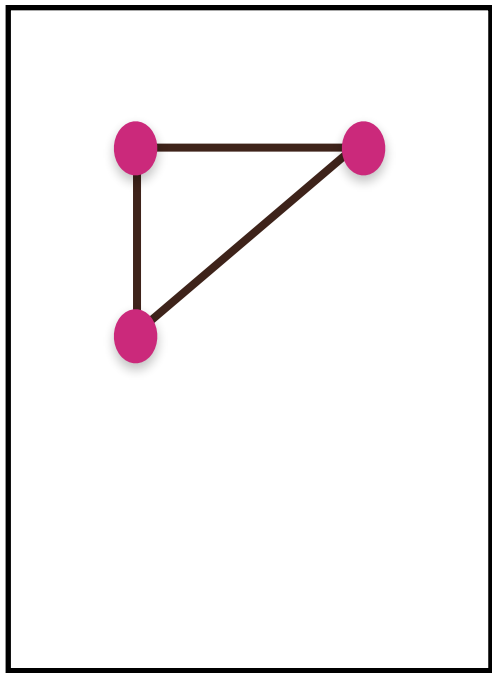
Chromatic Coding Algorithm



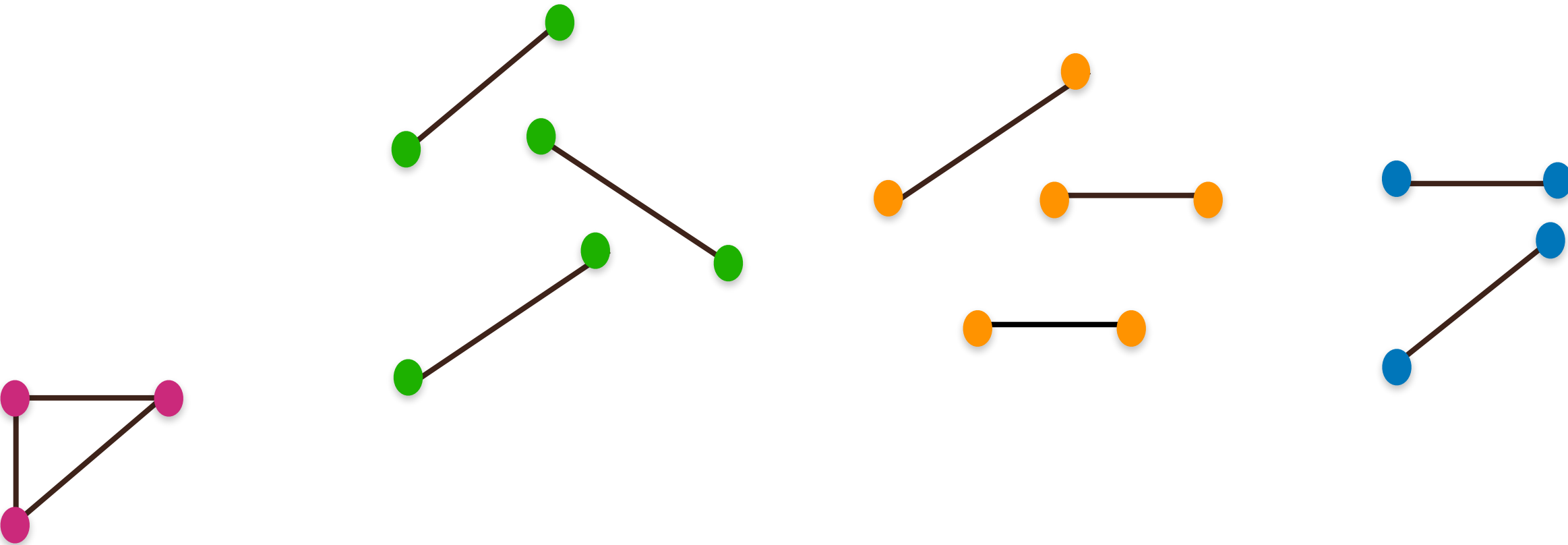
C_1

C_2

C_3



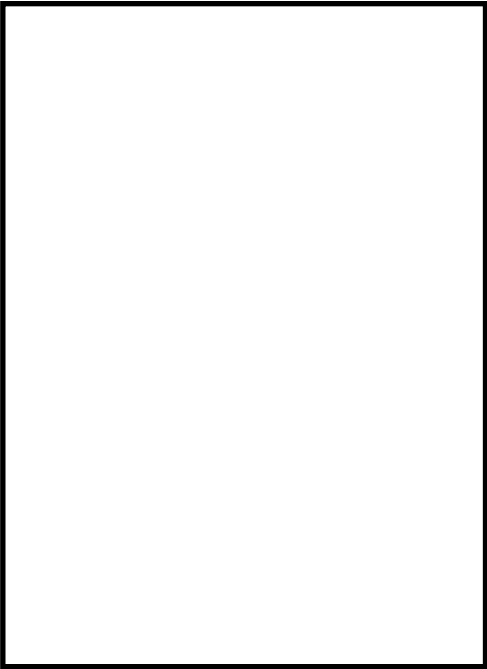
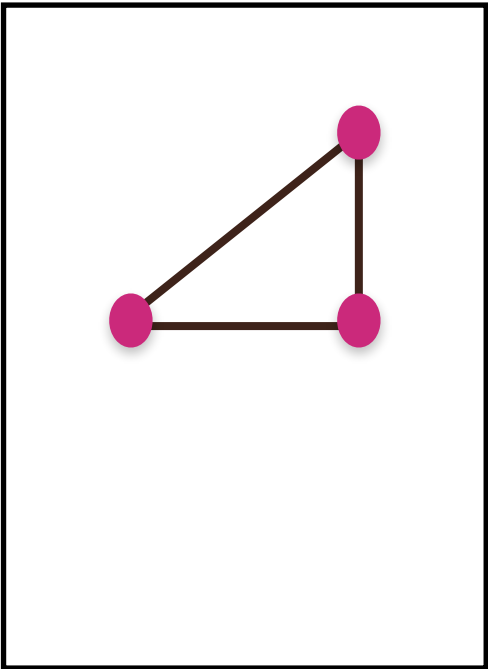
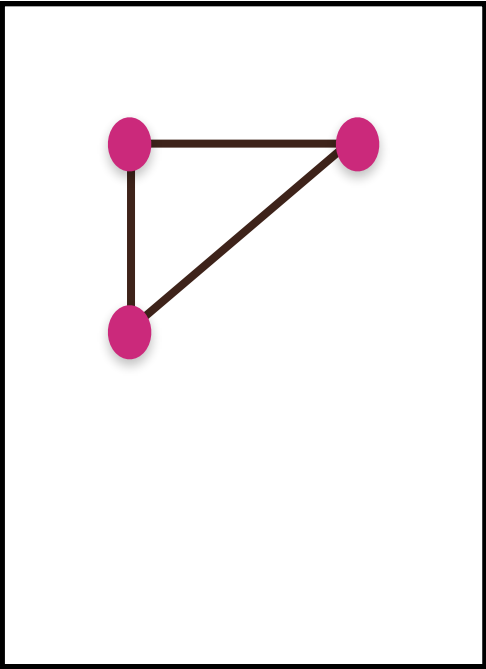
Chromatic Coding Algorithm



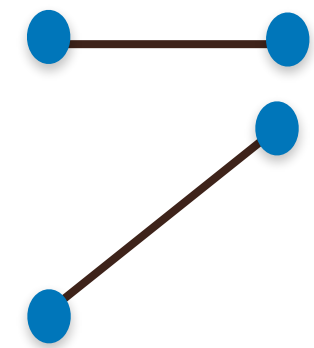
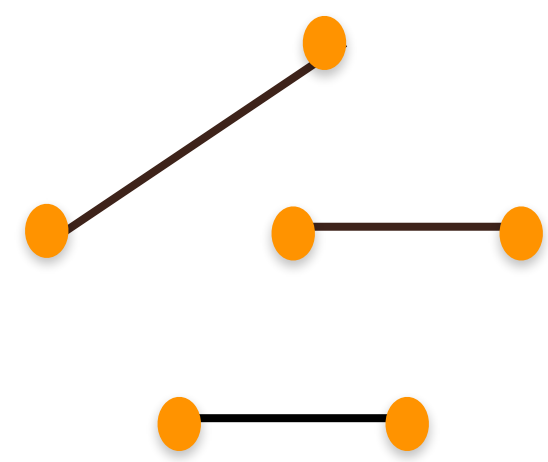
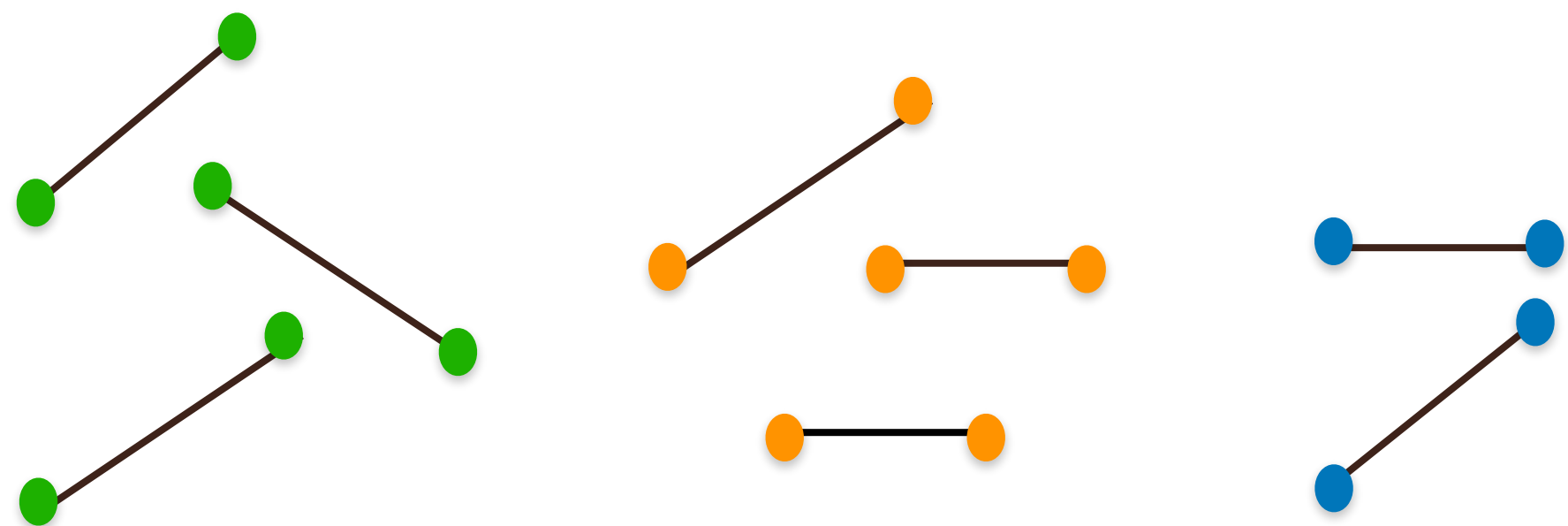
C_1

C_2

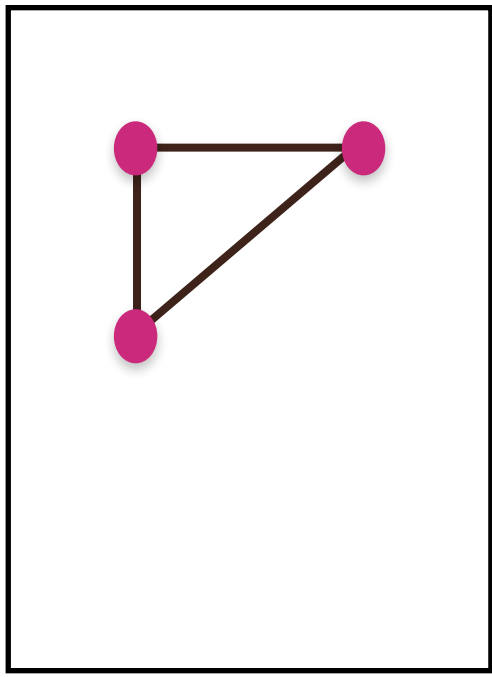
C_3



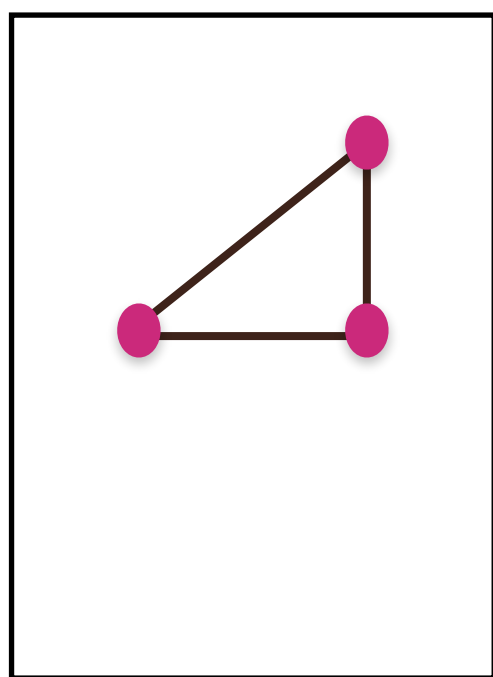
Chromatic Coding Algorithm



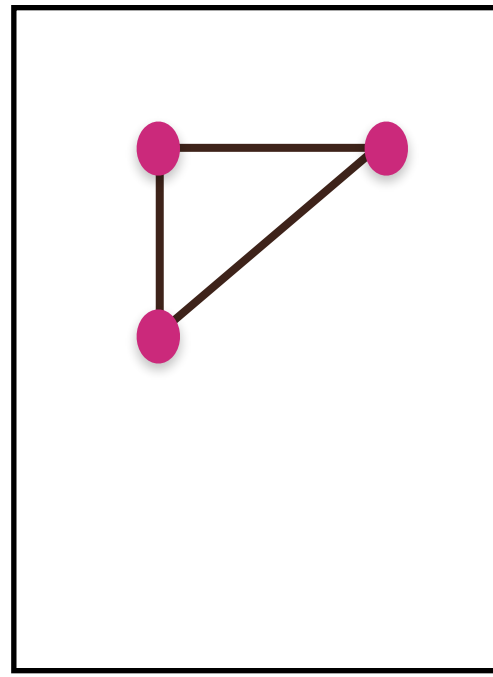
C_1



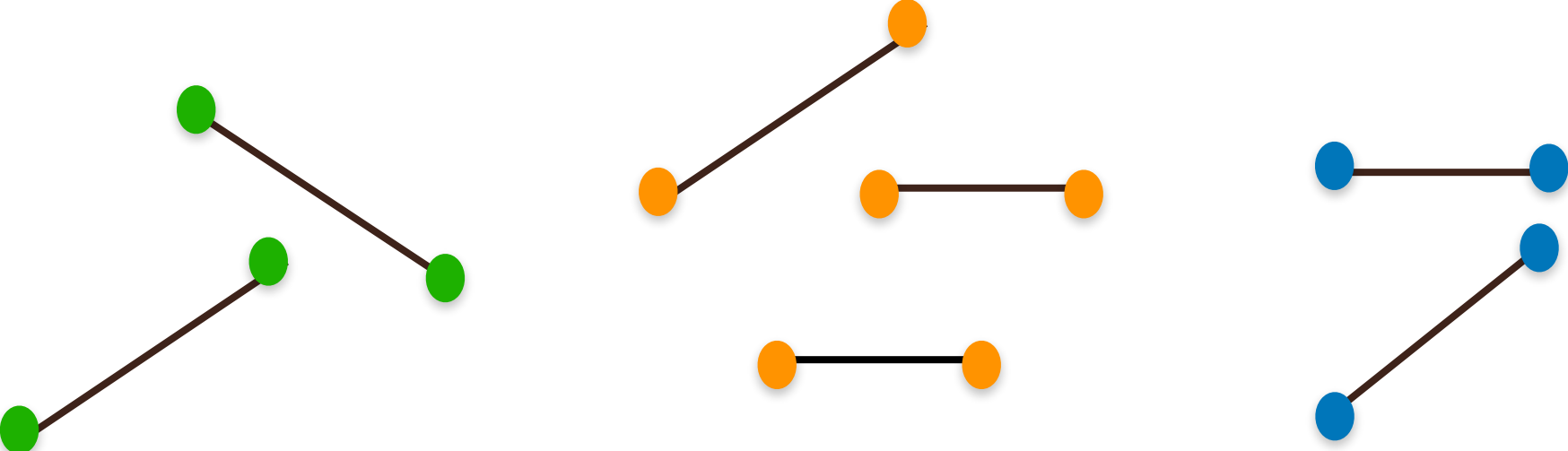
C_2



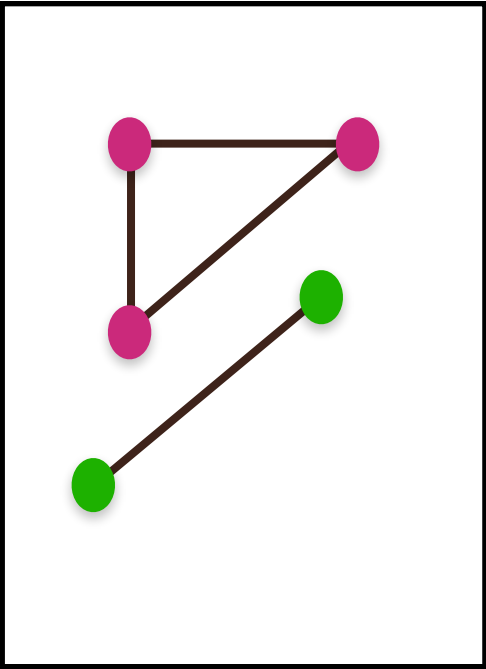
C_3



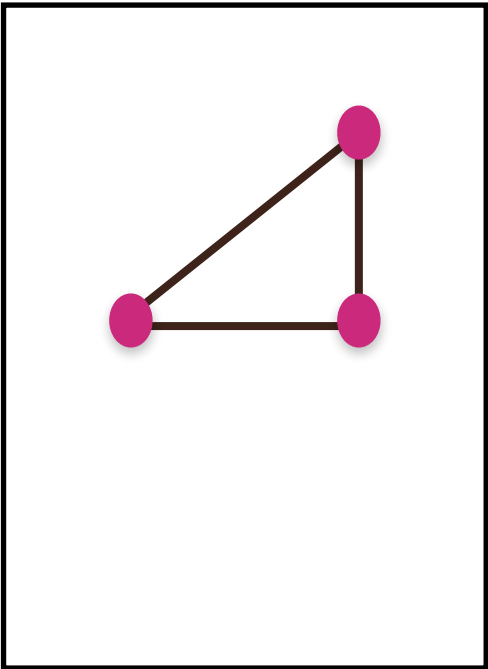
Chromatic Coding Algorithm



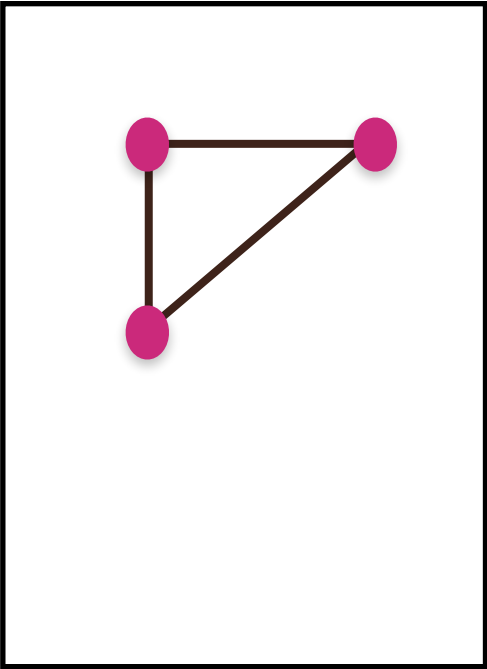
C_1



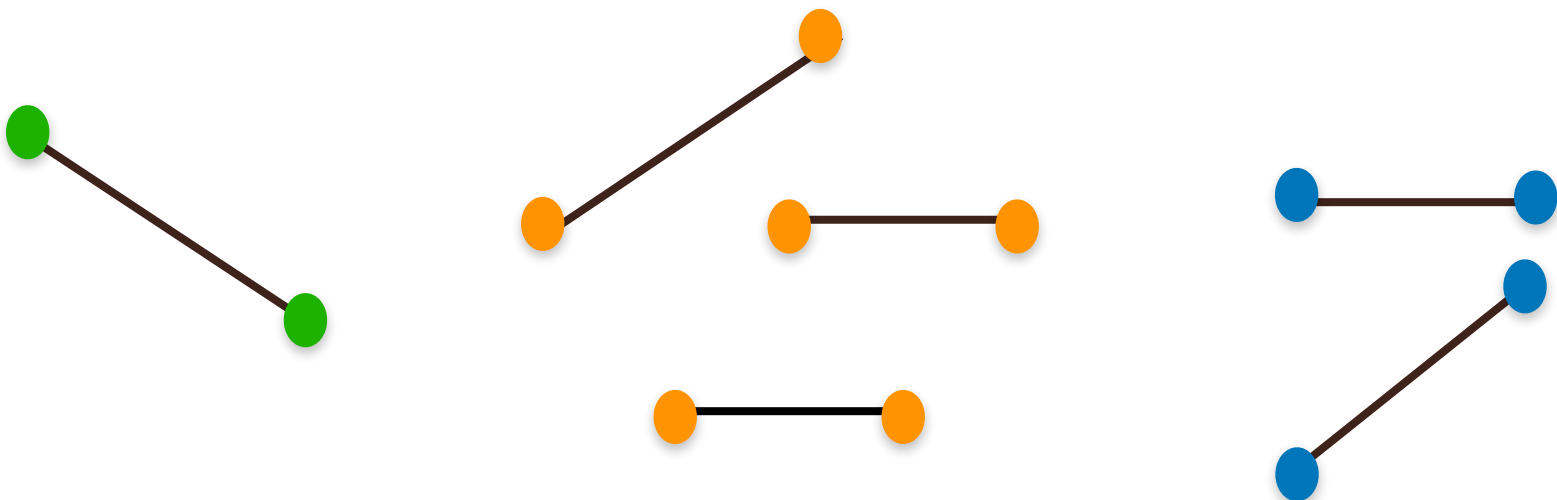
C_2



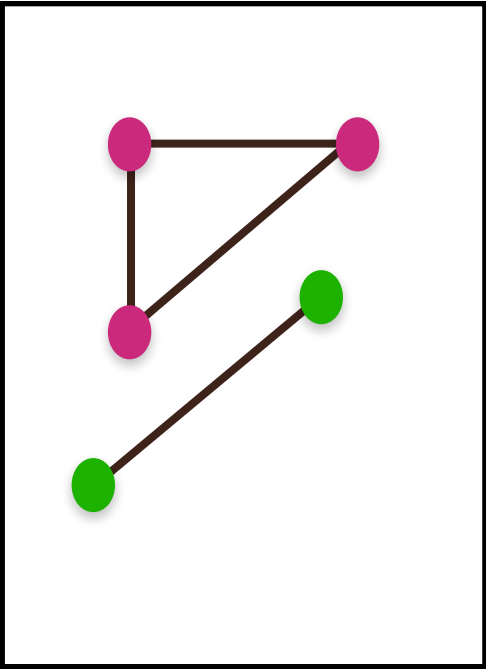
C_3



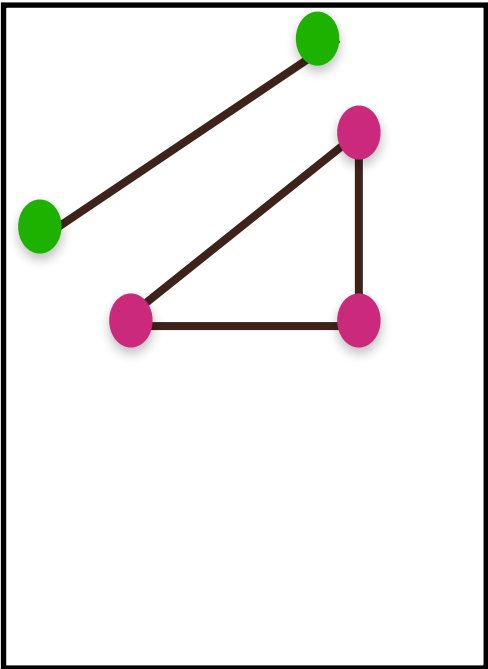
Chromatic Coding Algorithm



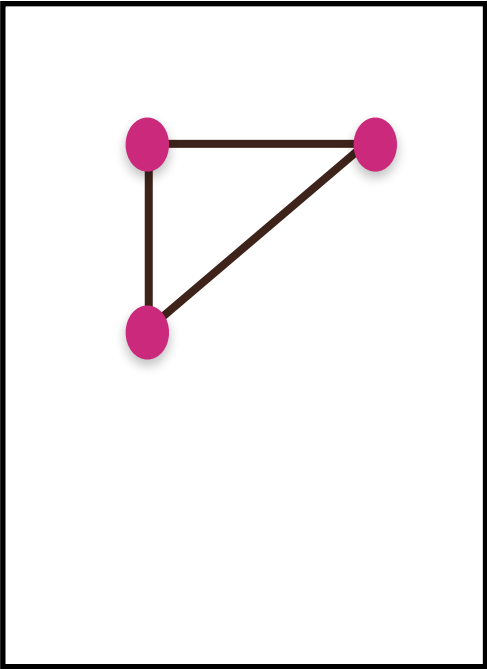
C_1



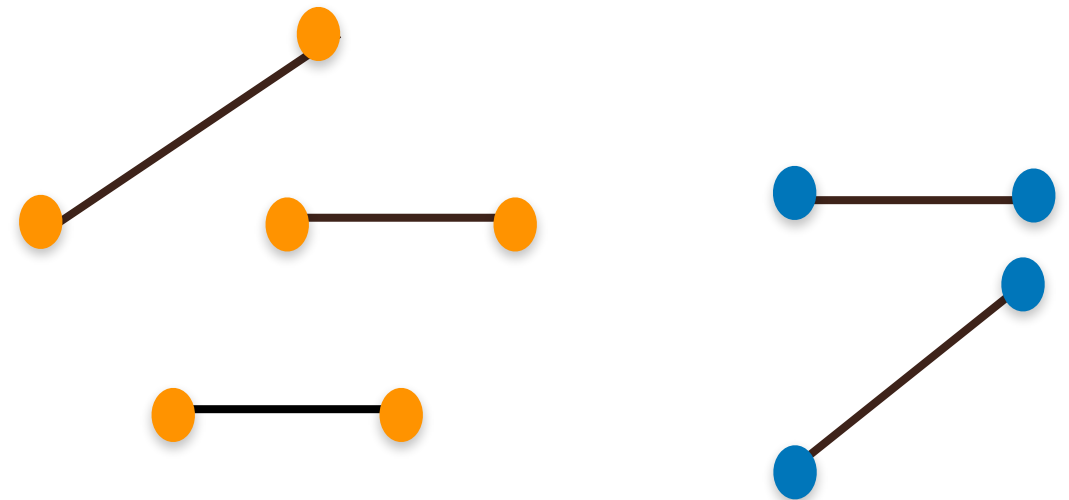
C_2



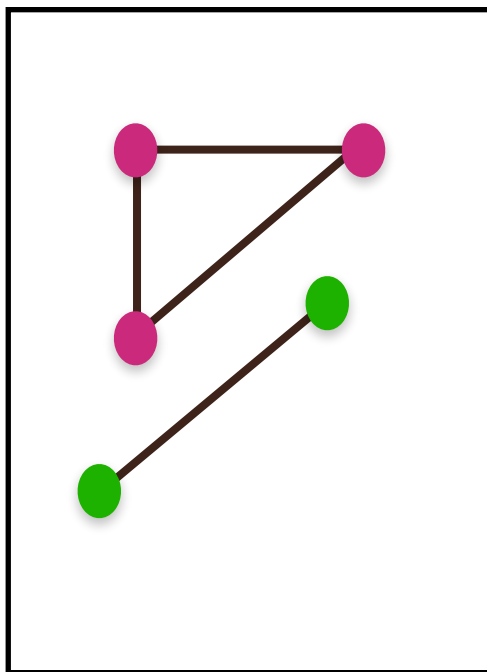
C_3



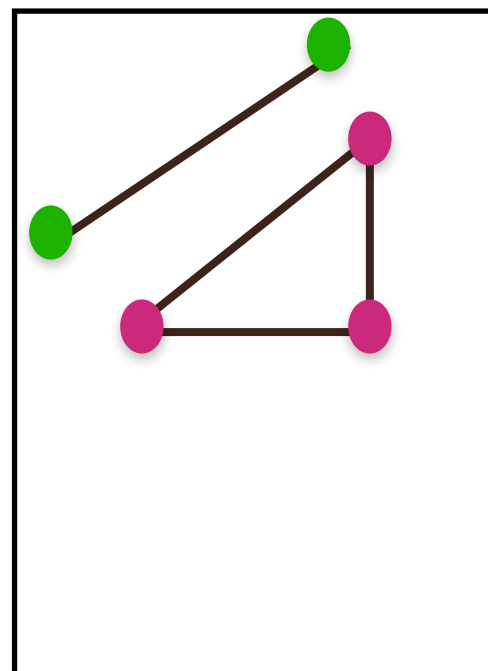
Chromatic Coding Algorithm



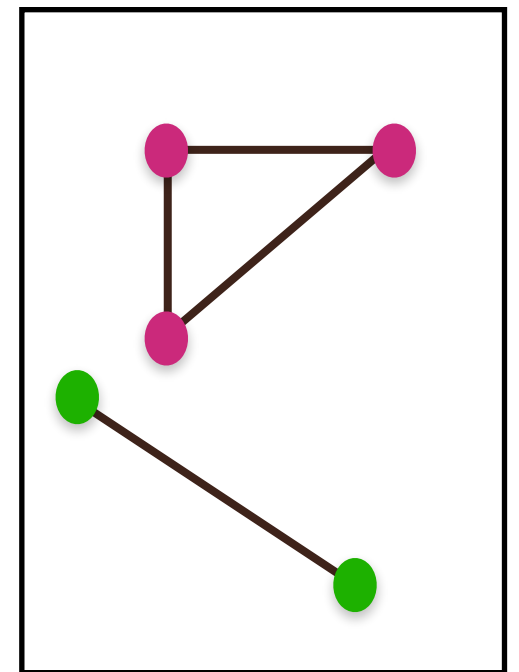
C_1



C_2



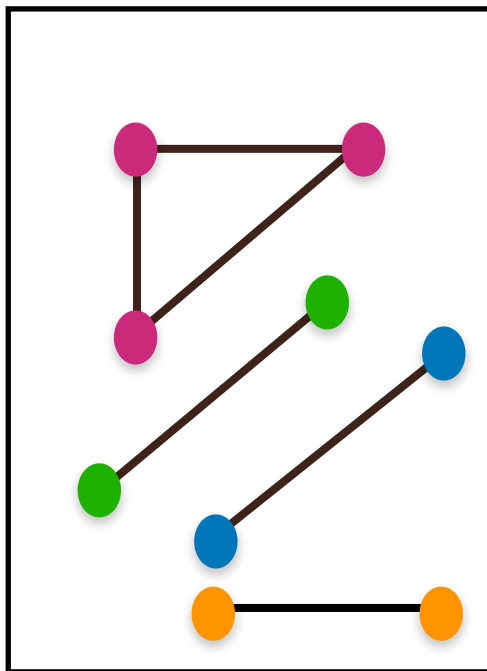
C_3



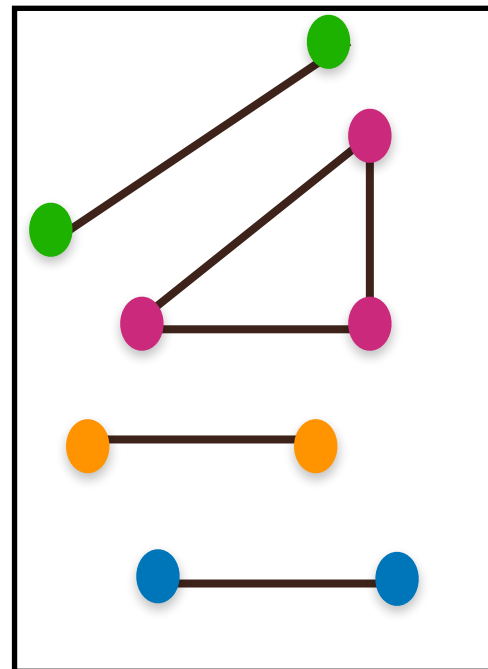
Chromatic Coding Algorithm

- * At most d^d choices

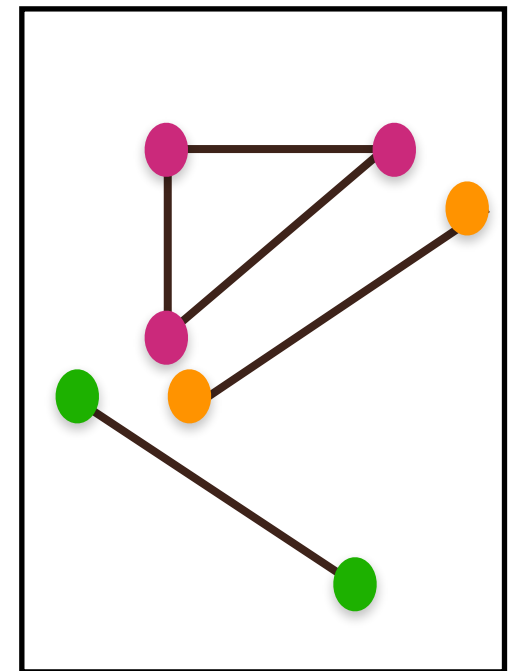
C_1



C_2



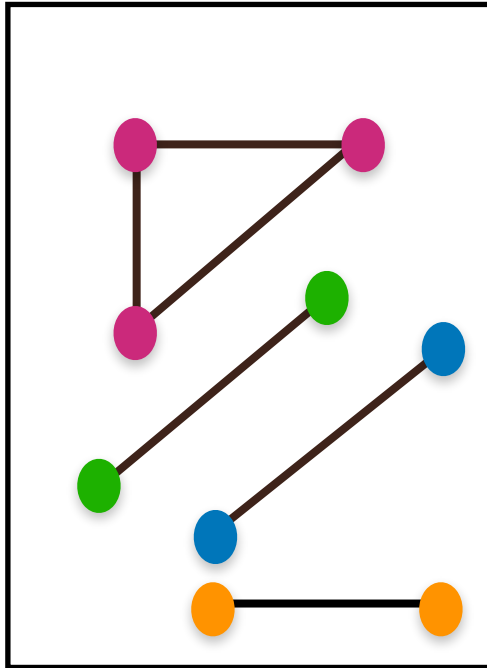
C_3



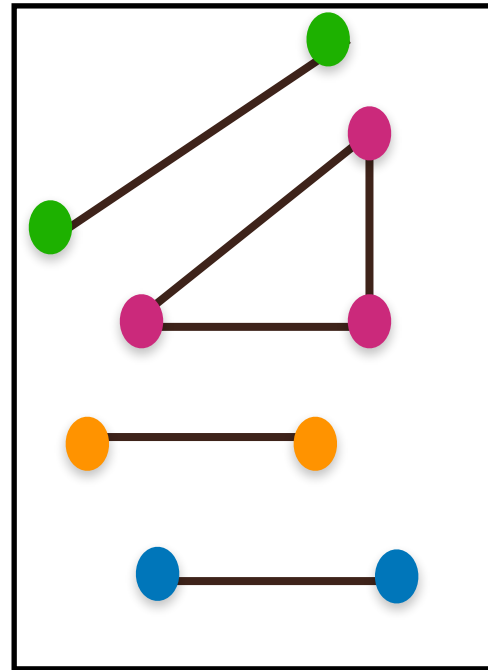
- * For each choice, find a solution that respects that choice

Chromatic Coding Algorithm

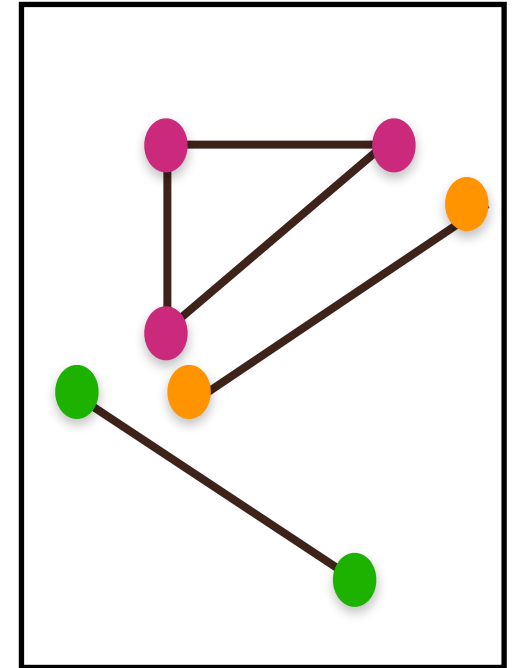
C_1



C_2



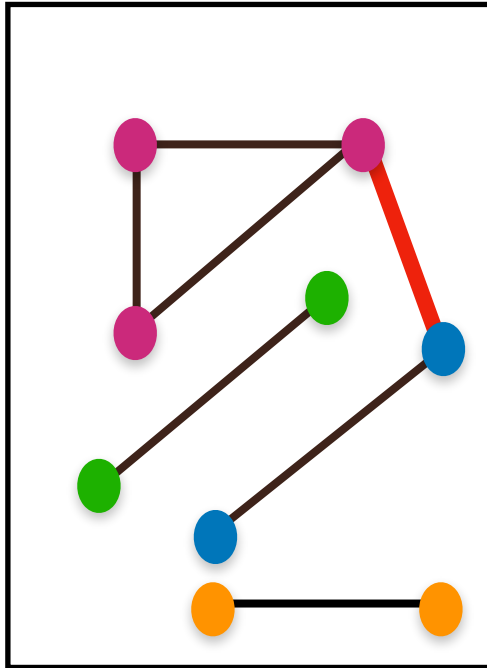
C_3



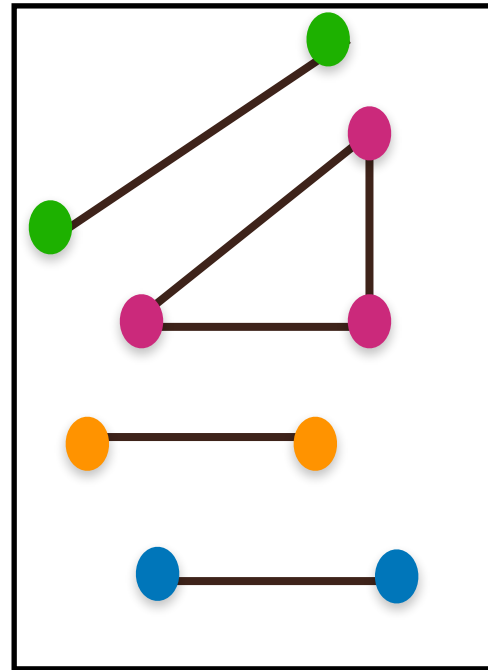
- * Add an edge between every pair of non-adjacent vertices that are guessed to be in the same component
- * Delete the edge between every pair of adjacent vertices that are guessed to be in different components

Chromatic Coding Algorithm

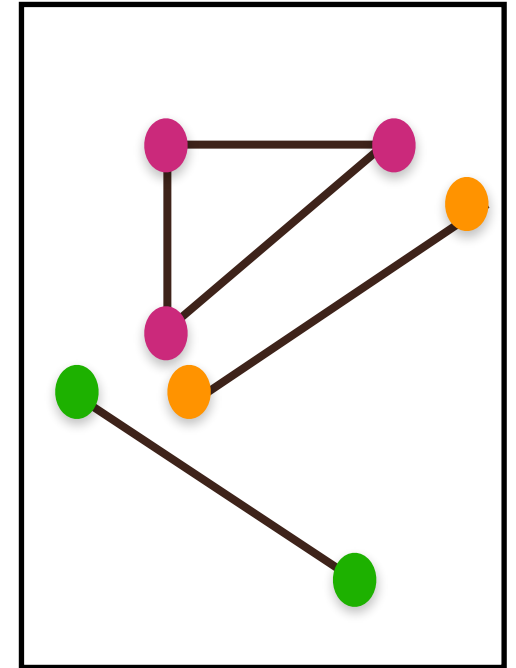
C_1



C_2



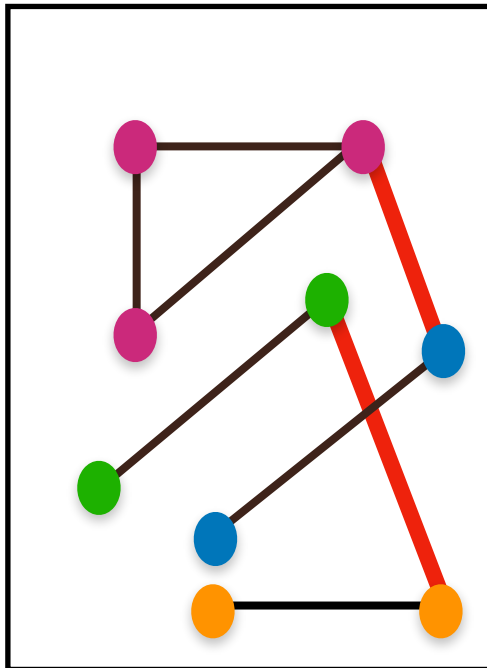
C_3



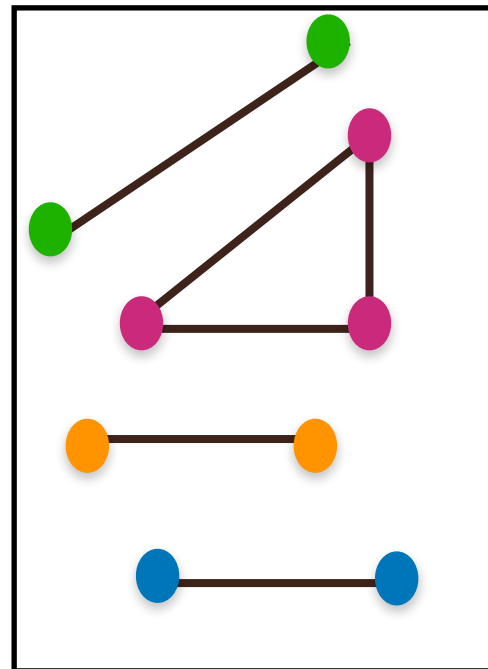
- * Add an edge between every pair of non-adjacent vertices that are guessed to be in the same component
- * Delete the edge between every pair of adjacent vertices that are guessed to be in different components

Chromatic Coding Algorithm

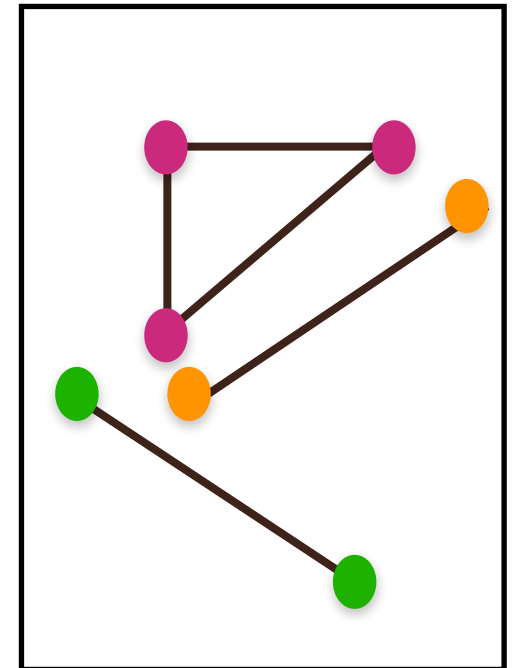
C_1



C_2



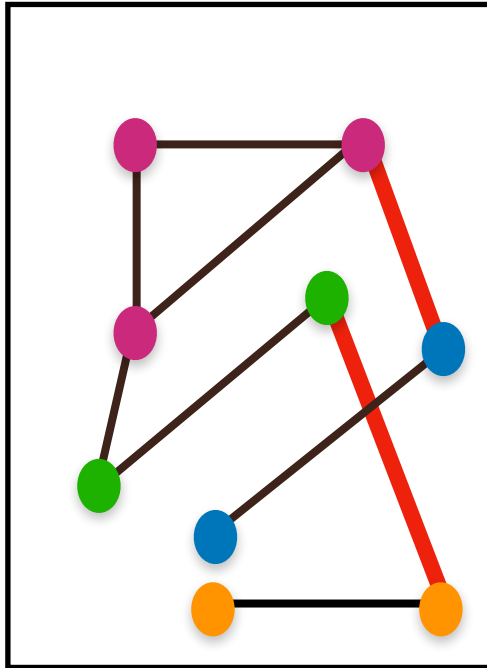
C_3



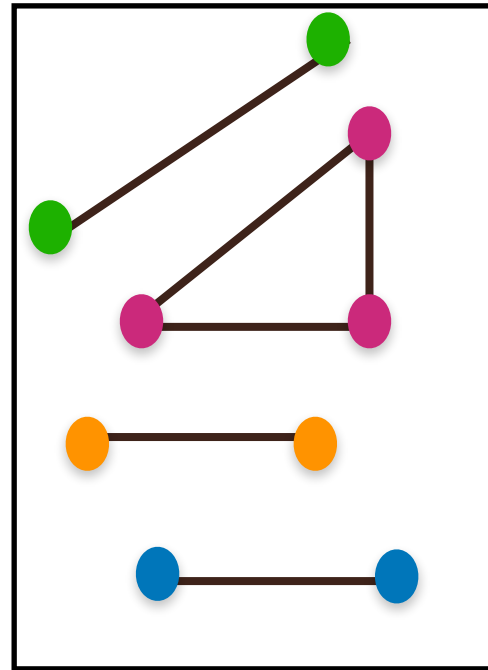
- * Add an edge between every pair of non-adjacent vertices that are guessed to be in the same component
- * Delete the edge between every pair of adjacent vertices that are guessed to be in different components

Chromatic Coding Algorithm

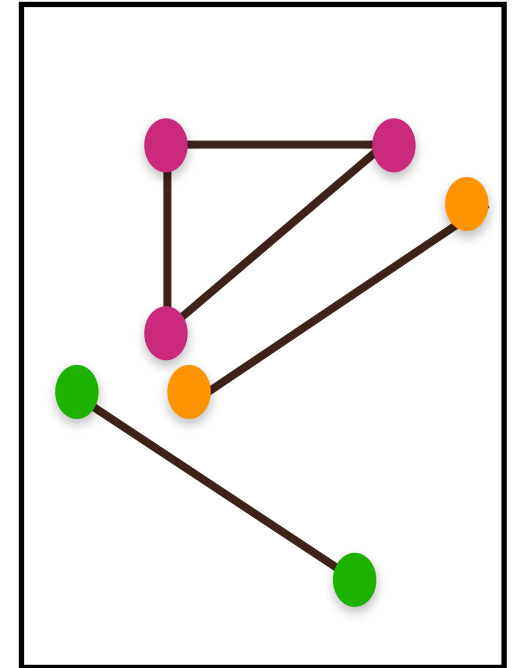
C_1



C_2

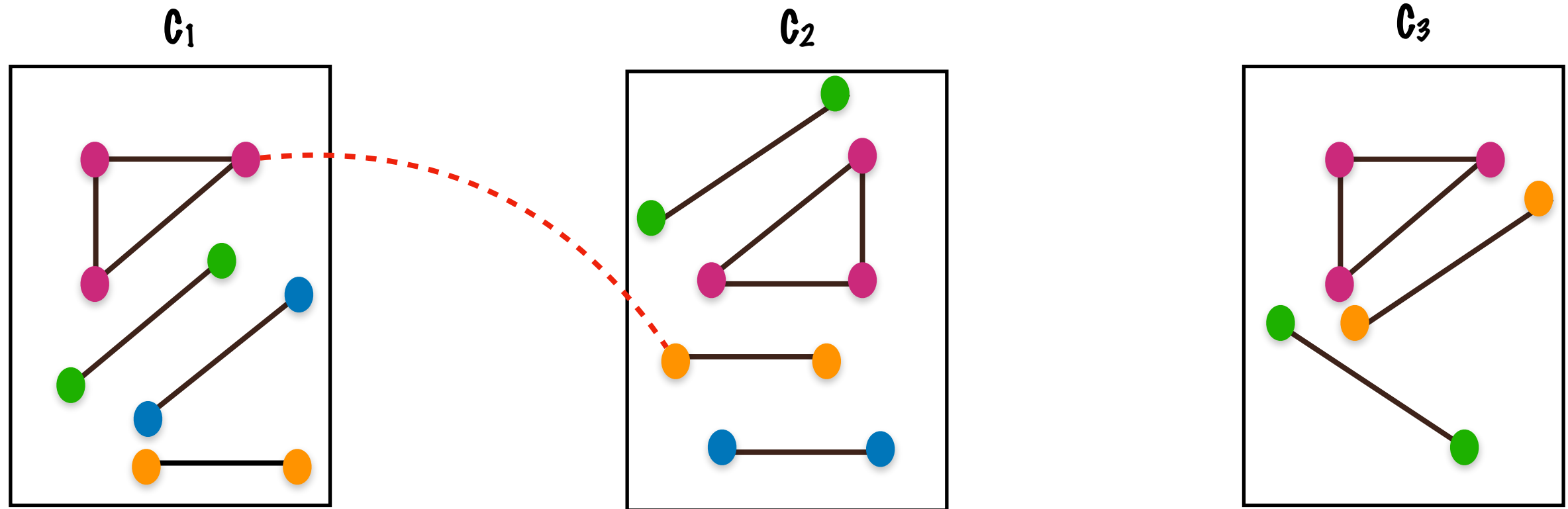


C_3



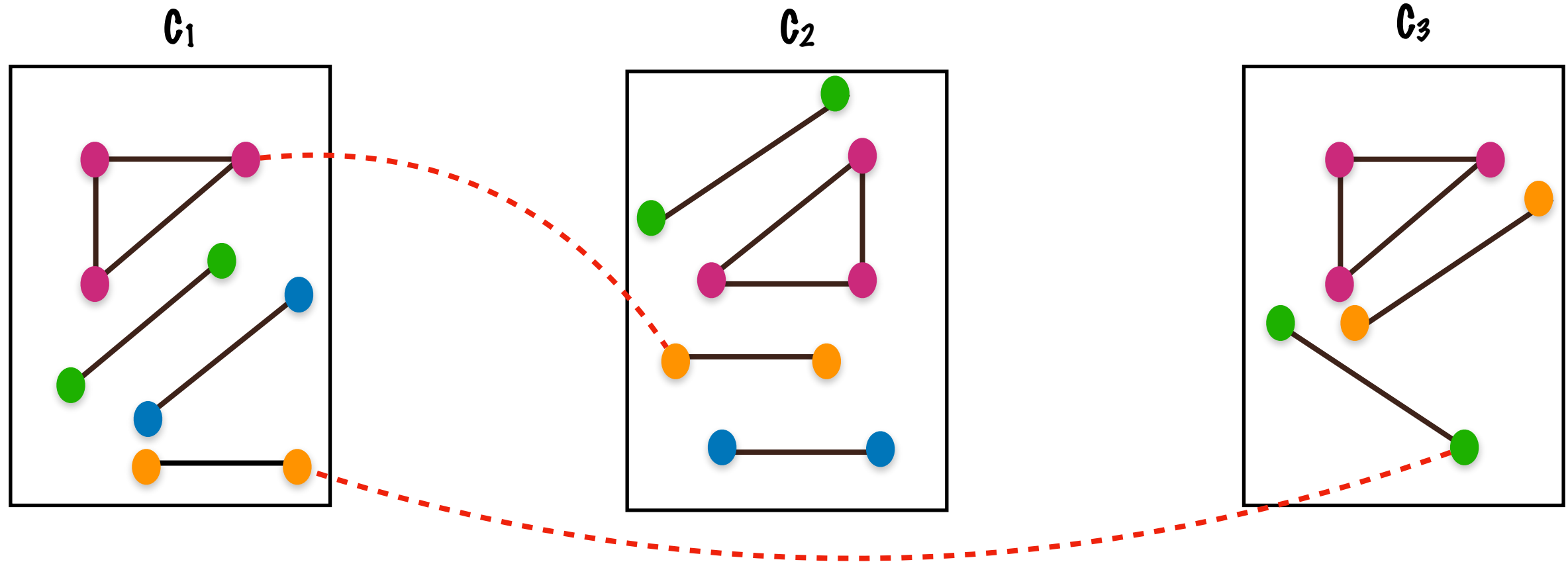
- * Add an edge between every pair of non-adjacent vertices that are guessed to be in the same component
- * Delete the edge between every pair of adjacent vertices that are guessed to be in different components

Chromatic Coding Algorithm



- * Add an edge between every pair of non-adjacent vertices that are guessed to be in the same component
- * Delete the edge between every pair of adjacent vertices that are guessed to be in different components

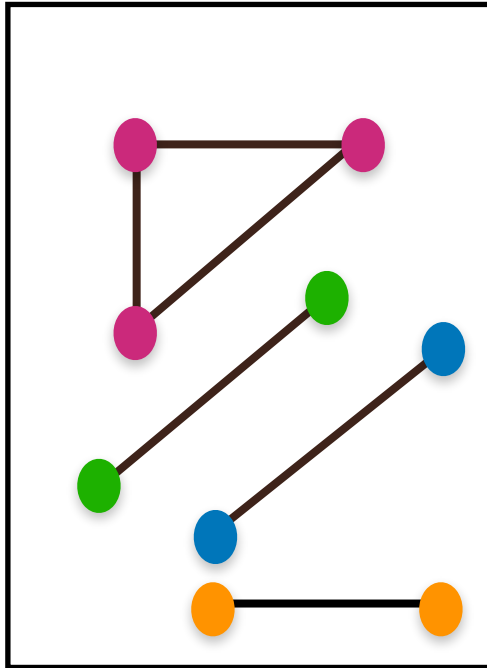
Chromatic Coding Algorithm



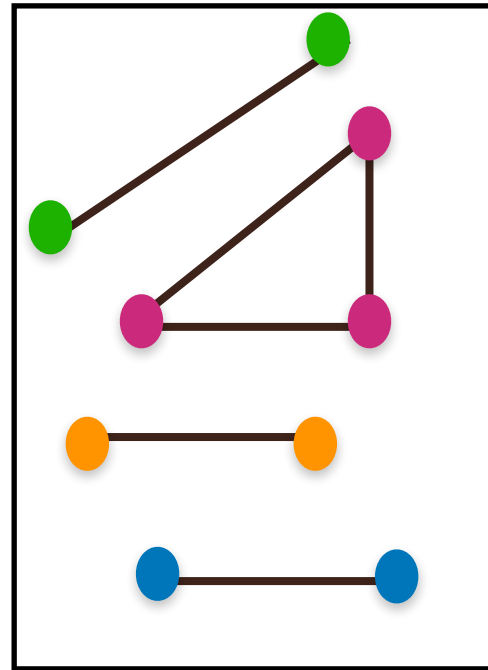
- * Add an edge between every pair of non-adjacent vertices that are guessed to be in the same component
- * Delete the edge between every pair of adjacent vertices that are guessed to be in different components

Chromatic Coding Algorithm

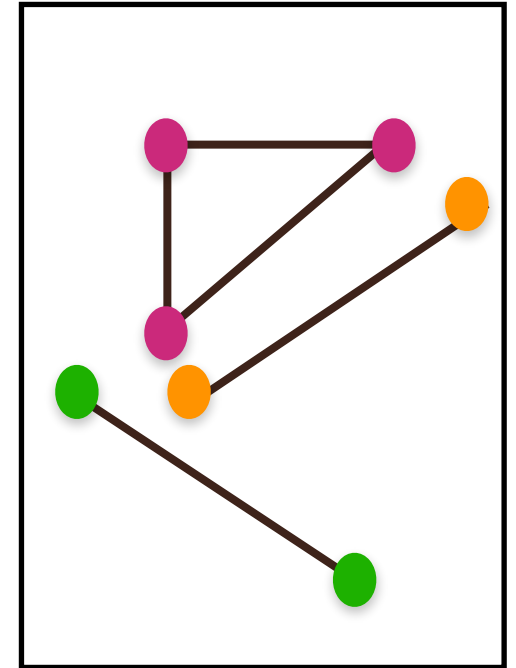
C_1



C_2



C_3



- * Add an edge between every pair of non-adjacent vertices that are guessed to be in the same component
- * Delete the edge between every pair of adjacent vertices that are guessed to be in different components
- * If total no. of edge modifications is $\leq k$, then we have a colorful solution

$2^{O(\sqrt{k} \cdot d \log d)} n^{O(1)}$ randomized algorithm

Analysis

- * Running Time: $2^{O(\sqrt{k} \cdot d \log d)} n^{O(1)}$
- * Correctness:
 - * If (G, k) is a no-instance then Algorithm is correct
 - * If (G, k) is a yes-instance
 - * The colouring need not color the edges of any solution properly
 - * Success probability $\geq 2^{-\sqrt{k}/2}$

Theorem: d -Clustering can be solved in randomized $2^{O(\sqrt{k} \cdot d \log d)} n^{O(1)}$ time, with success probability at least $2^{-\sqrt{k}/2}$.

- * By repeating the algorithm $2^{O(\sqrt{k}/2)}$ times,

Theorem: d -Clustering can be solved in randomized $2^{O(\sqrt{k} \cdot d \log d)} n^{O(1)}$ time, with constant success probability.

Analysis

Lemma: If the vertices of a simple graph G on k edges are coloured independently and uniformly at random with $\lceil (8k)^{.5} \rceil$ colors, then the probability that $E(G)$ is properly colored is \geq at least $2^{-\sqrt{k/2}}$.

- * Let v_1 be a vertex of min degree in $G_0 = G$
- * Let v_2 be a vertex of min degree in $G_1 = G - v_1$
- * Let v_3 be a vertex of min degree in $G_2 = G - \{v_1 \cup v_2\}$
- * Let v_4 be a vertex of min degree in $G_3 = G - \{v_1 \cup v_2 \cup v_3\}$
- * ...
- * Let v_n be a vertex of min degree in $G_{n-1} = G - \{v_1 \cup v_2 \cup v_3 \cup \dots \cup v_{n-1}\}$
- * G_n is the empty graph

Analysis

Lemma: If the vertices of a simple graph G on k edges are coloured independently and uniformly at random with $\lceil (8k)^{.5} \rceil$ colors, then the probability that $E(G)$ is properly colored is \geq at least $2^{-\sqrt{k/2}}$.

- * v_i is a vertex of min degree in $G_{i-1} = G - \{v_1 \cup v_2 \cup v_3 \cup \dots \cup v_{i-1}\}$
- * d_i is the degree of v_i in G_{i-1}
- * $d_i \leq |V(G_{i-1})| - 1$ as graph is simple
- * $2k = 2|E(G)| \geq 2|E(G_{i-1})| \geq d_i |V(G_{i-1})|$
- * $d_i \leq \sqrt{2k}$

Analysis

Lemma: If the vertices of a simple graph G on k edges are coloured independently and uniformly at random with $\lceil (8k)^{.5} \rceil$ colors, then the probability that $E(G)$ is properly colored is \geq at least $2^{-\sqrt{k/2}}$.

- * v_i is a vertex of min degree in $G_{i-1} = G - \{v_1 \cup v_2 \cup v_3 \cup \dots \cup v_{i-1}\}$
- * d_i is the degree of v_i in G_{i-1} and $d_i \leq \sqrt{2k}$

Consider the process of randomly coloring $V(G)$ in the order $v_n, v_{n-1}, v_{n-2}, \dots, v_1$

- * P_i : event that $E(G_i)$ is properly colored
 - * $\Pr(P_n) = 1$

Analysis

Identity: $1-x \geq 2^{-2x}$ for $0 \leq x \leq 1/2$

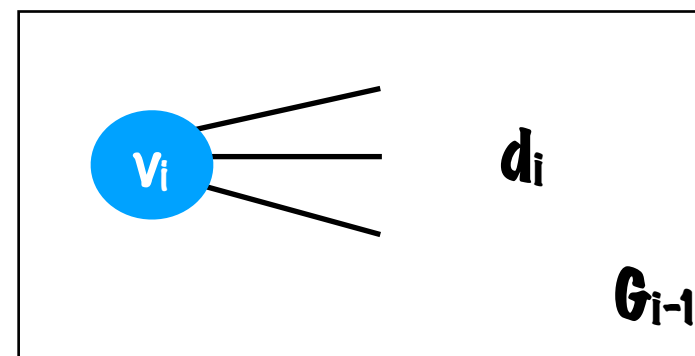
Lemma: If the vertices of a simple graph G on k edges are coloured independently and uniformly at random with $\lceil (8k)^{.5} \rceil$ colors, then the probability that $E(G)$ is properly colored is \geq at least $2^{-\sqrt{k/2}}$.

- * P_i : event that $E(G_i)$ is properly colored
 - * $\Pr(P_n) = 1$
- * $\Pr(P_{i-1} \mid P_i) \geq (q-d_i)/q \geq 2^{-2d_i/q}$

Properly colored

G_i

$V_n, V_{n-1}, V_{n-2}, \dots, V_{i+1}$



$V_n, V_{n-1}, V_{n-2}, \dots, V_{i+1}, v_i$

Analysis

Lemma: If the vertices of a simple graph G on k edges are coloured independently and uniformly at random with $\lceil (8k)^{.5} \rceil$ colors, then the probability that $E(G)$ is properly colored is \geq at least $2^{-\sqrt{k/2}}$.

$$\Pr(P_0) = \Pr(P_0 \mid P_1) \Pr(P_0) = \Pr(P_0 \mid P_1) \Pr(P_1 \mid P_2) \Pr(P_2)$$

$$= \dots = \Pr(P_n) \prod \Pr(P_{i-1} \mid P_i)$$

$$\geq \prod 2^{\{-2d_i/q\}} = 2^{\{-2\sum d_i/q\}} = 2^{\{-2k/\lceil \sqrt{8k} \rceil\}} = 2^{-\sqrt{k/2}}$$

Derandomization

Definition: A (n,k,q) -coloring family F is a family of functions from $[n]$ to $[q]$ such that for every graph G on vertex set $[n]$ with $\leq k$ edges, there is a function f in F that properly colors $E(G)$.

Theorem: For any $n,k \geq 1$, there is a $(n, k, \mathcal{O}(\sqrt{k}))$ -coloring family of size $2^{O(\sqrt{k} \log k)} \log n$ that can be constructed in $2^{O(\sqrt{k} \log k)} n \log n$ time.

Theorem: d -Clustering can be solved in $2^{O(\sqrt{k}(d+\log k))} n^{O(1)}$ time.