Computing Square Colorings Of Graphs

RTA 2023, NISER Bhubaneshwar

Akanksha Agrawal

Indian Institute of Technology Madras

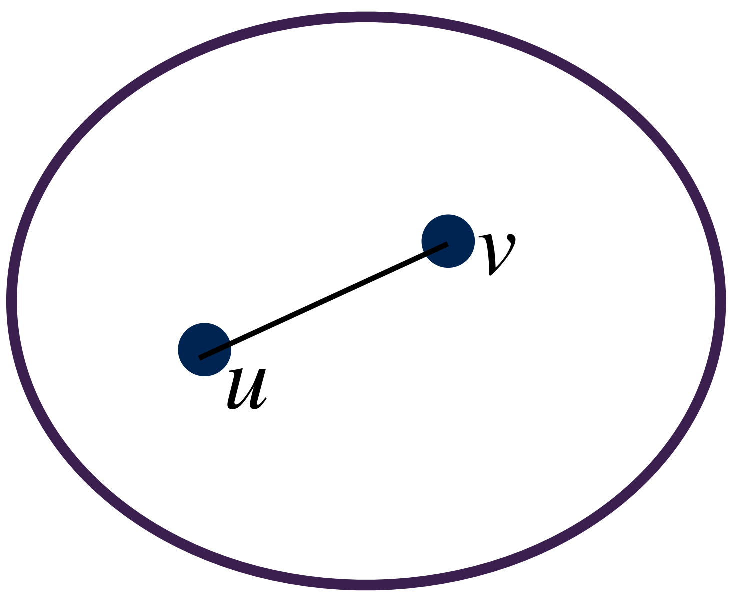
Joint work with Dániel Marx, Daniel Neuen and Jasper Slusallek

Roadmap

- Background & Overview
- Algorithm for Coloring based on dynamic programming over tree decompositions.
- Our algorithm for bounded treewidth graphs
- Our algorithm on planar graphs
- Conclusion & Open problems

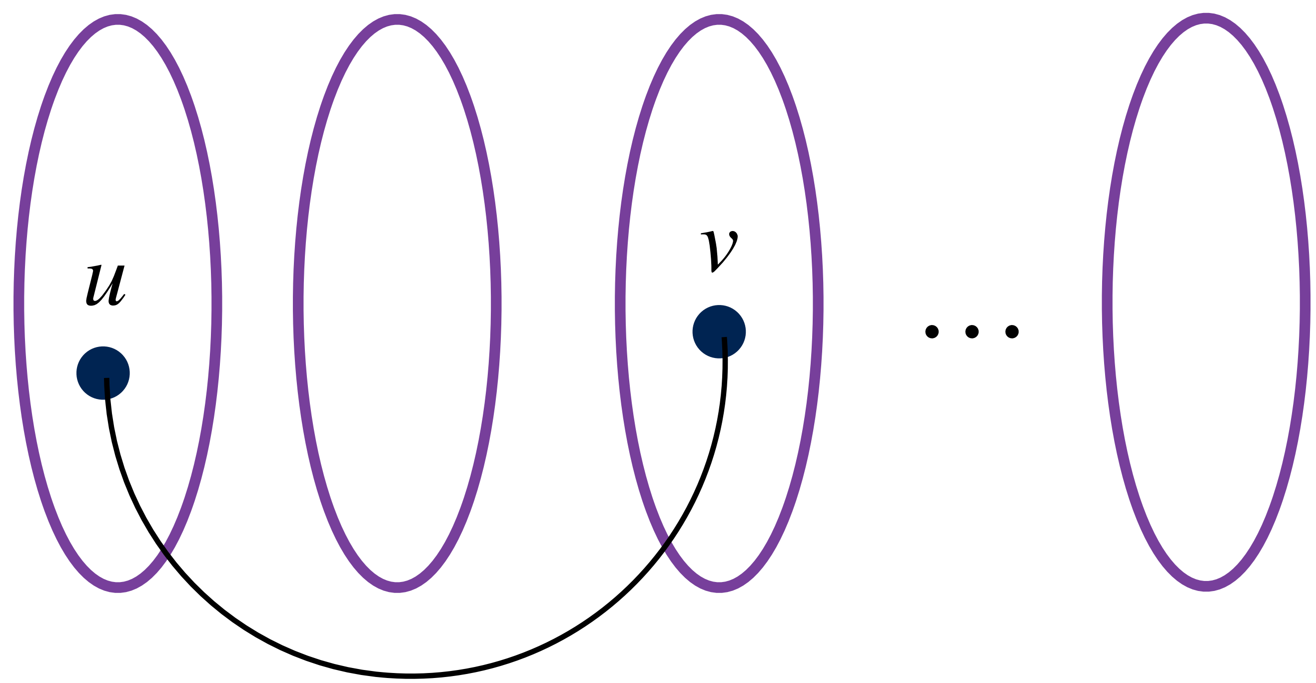
Graph Coloring

Graph G



and integer q

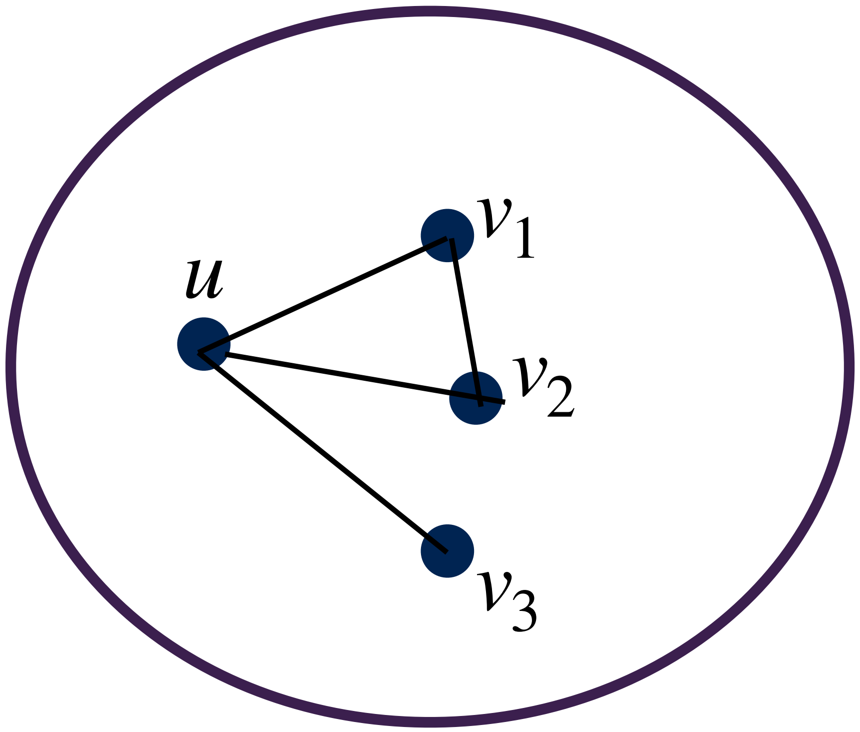
$C_1 \uplus C_2 \uplus C_3 \cdots \uplus C_q = V(G)$



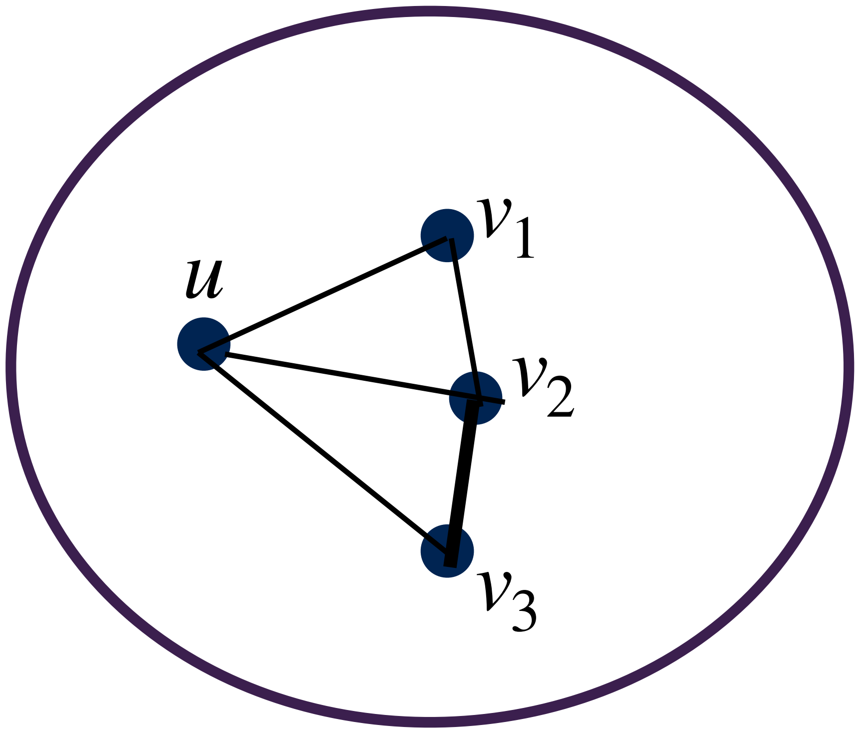
Endpoints of the edges are in different parts

Square of a graph

Graph G



Square of G: G^2

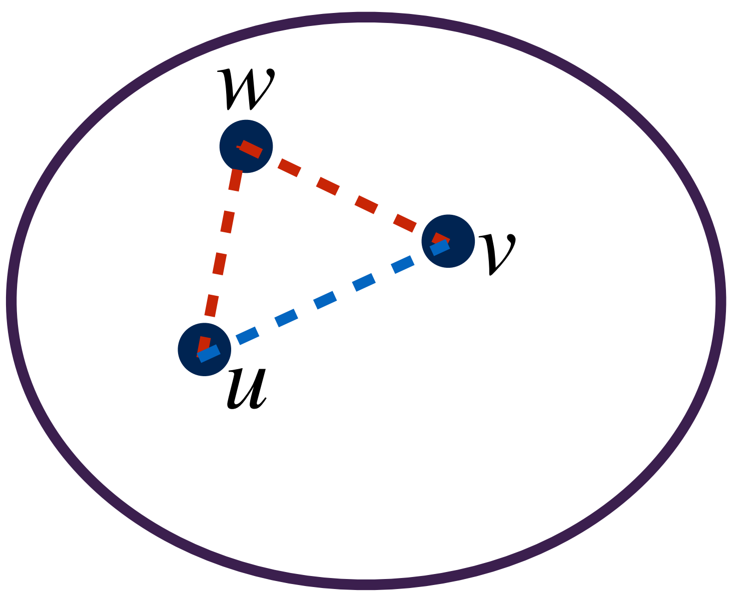


$$V(G^2) = V(G)$$
$$E(G^2) = \{\{u, v\} \mid 1 \leq \mathrm{dist}_G(u, v) \leq 2\}$$

$\mathrm{dist}_G(u, v)$ = the number of edges in the shortest path between u and v in G

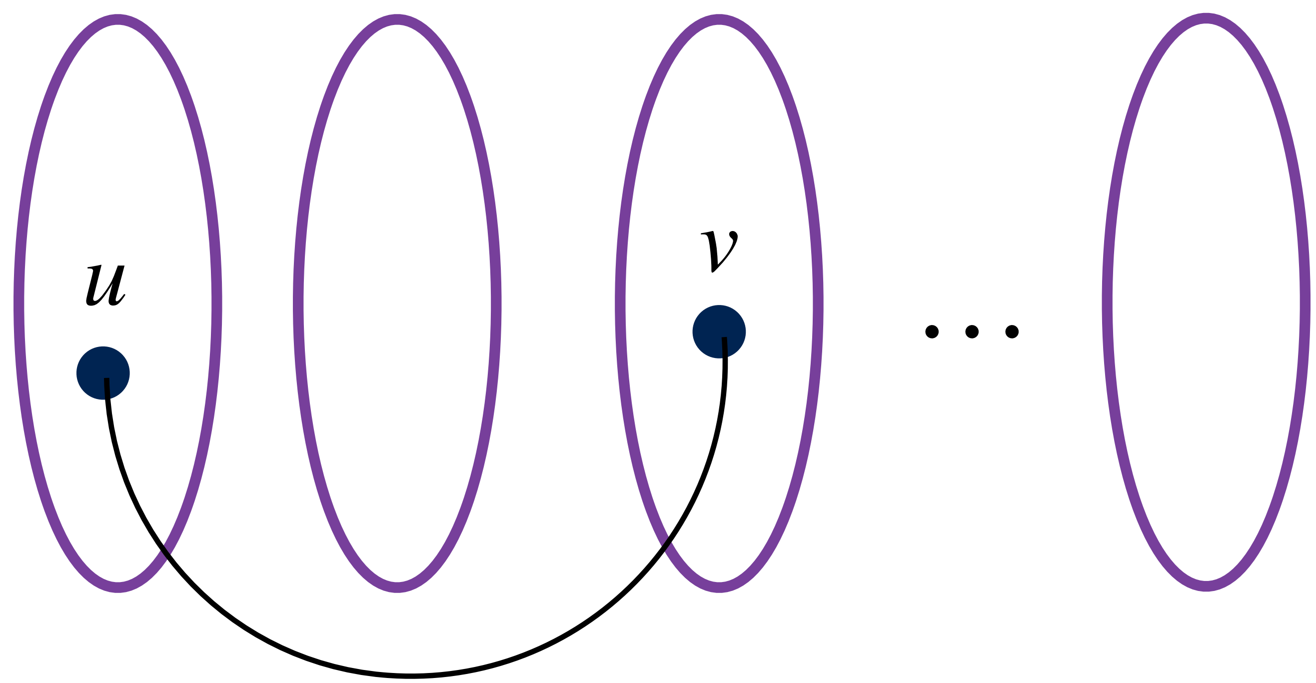
Square Coloring

Graph G



and integer q

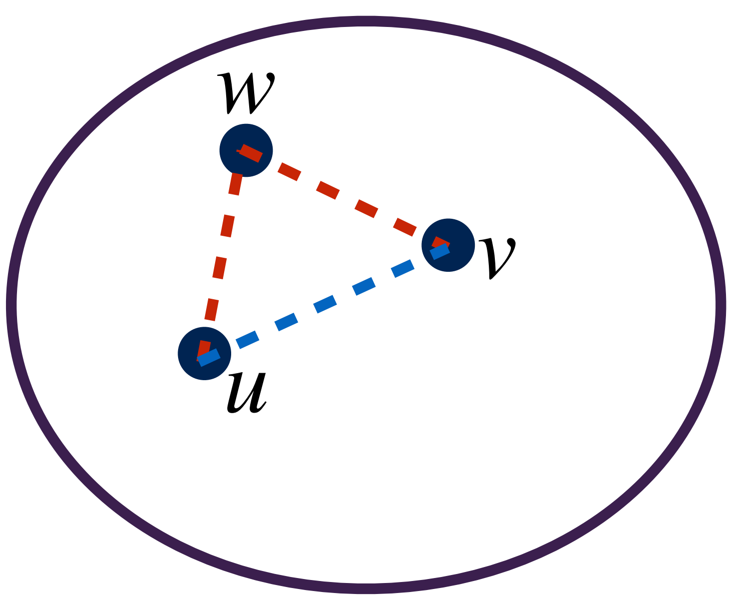
$C_1 \uplus C_2 \uplus C_3 \cdots \uplus C_q = V(G)$



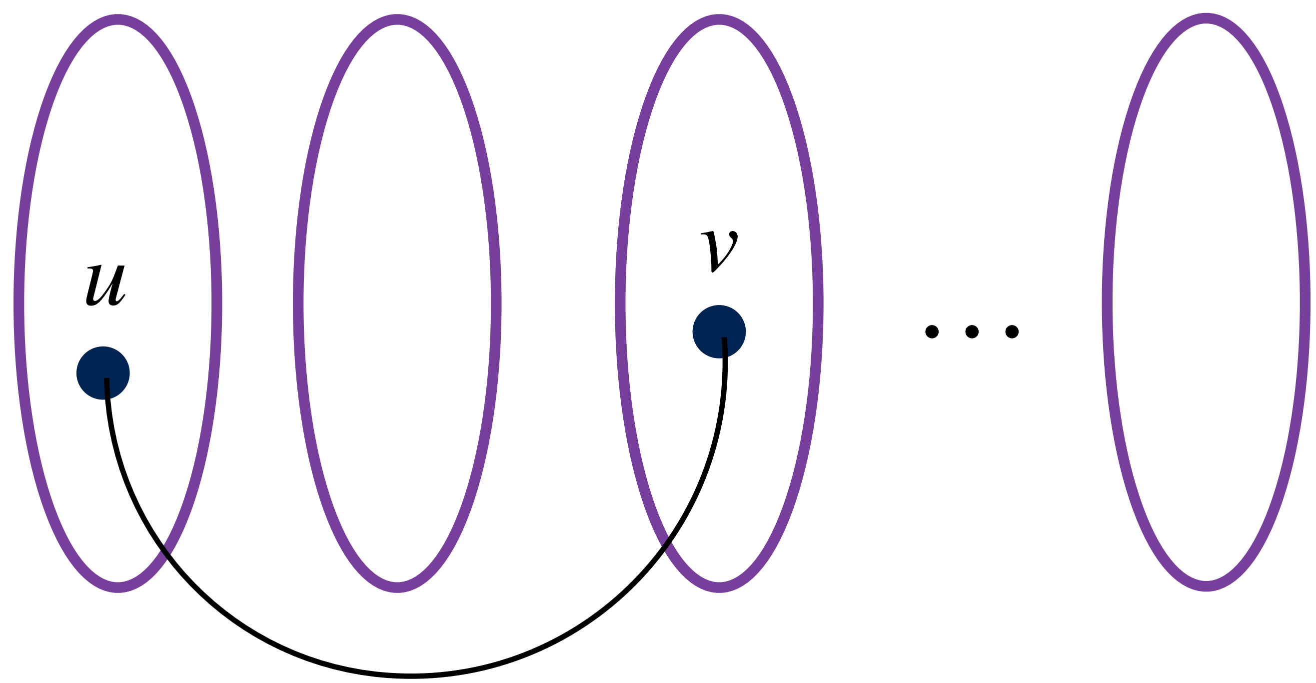
Endpoints of the edges in G^2 are in different parts

Square Coloring

Graph G

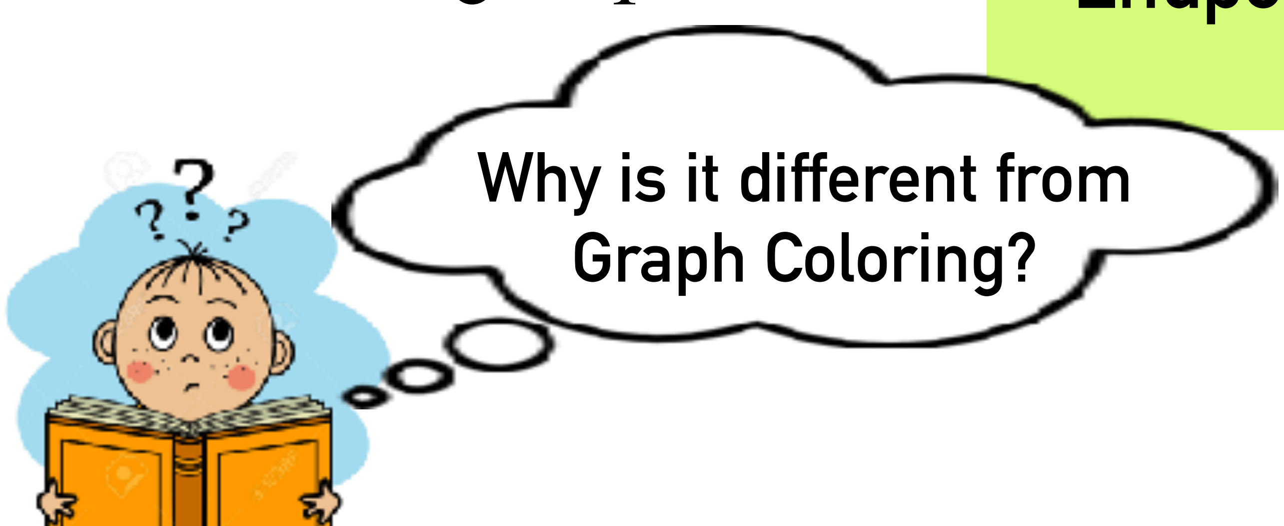


$$C_1 \uplus C_2 \uplus C_3 \cdots \uplus C_q = V(G)$$



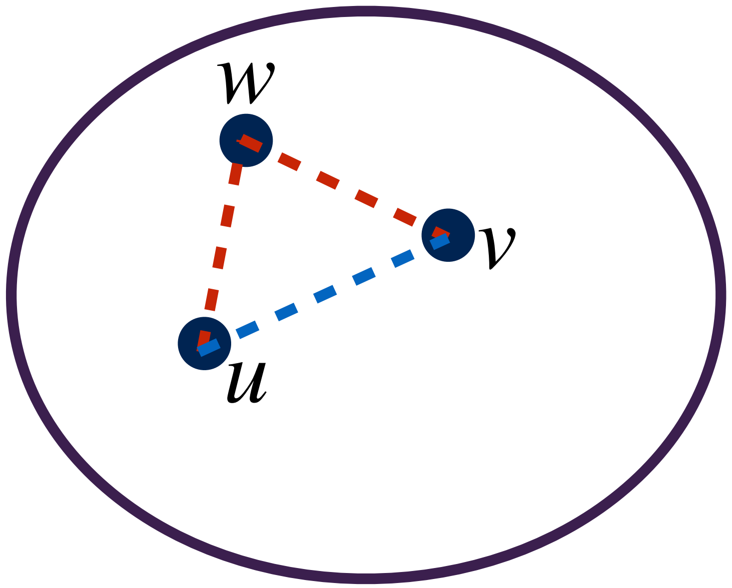
and integer q

Endpoints of the edges in G^2 are in different parts

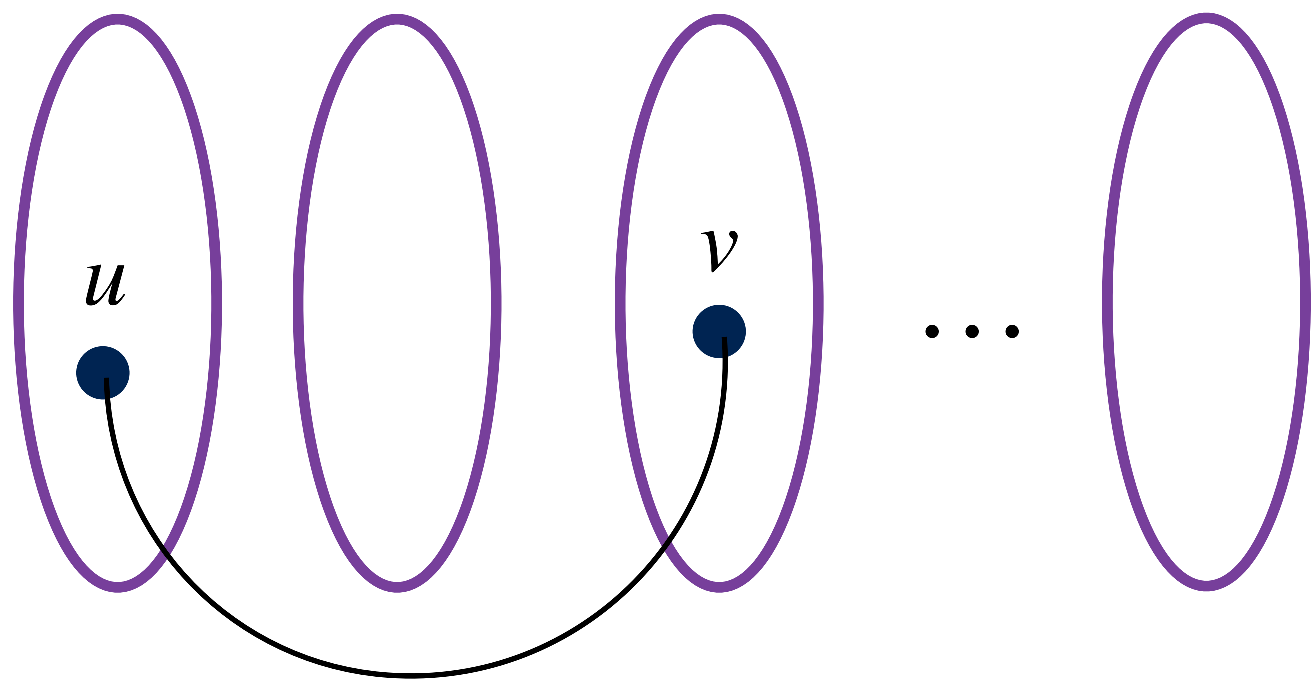


Square Coloring

Graph G

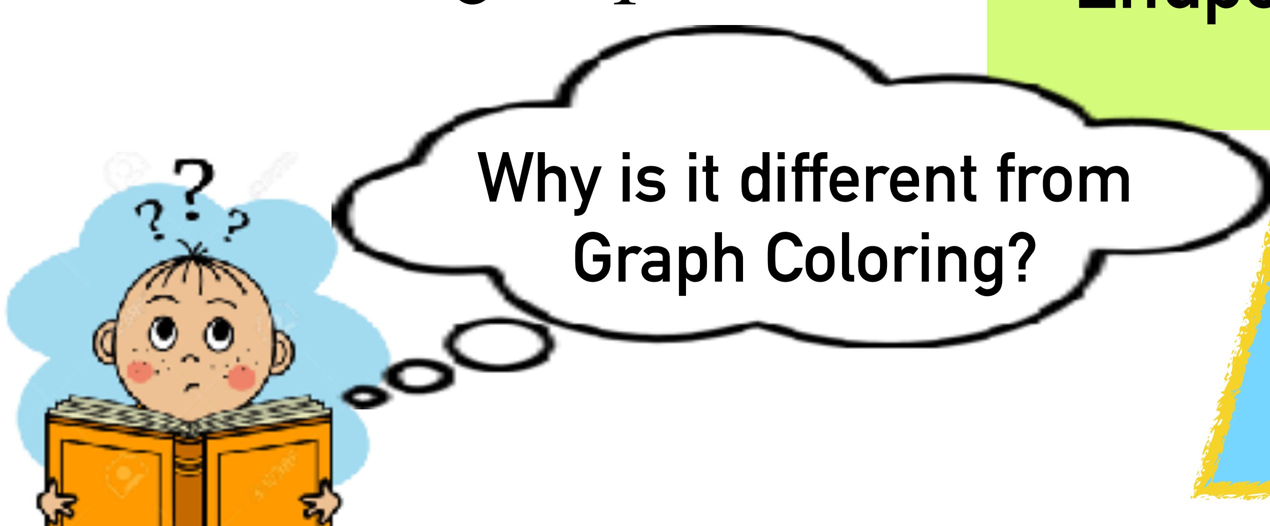


$C_1 \uplus C_2 \uplus C_3 \cdots \uplus C_q = V(G)$



and integer q

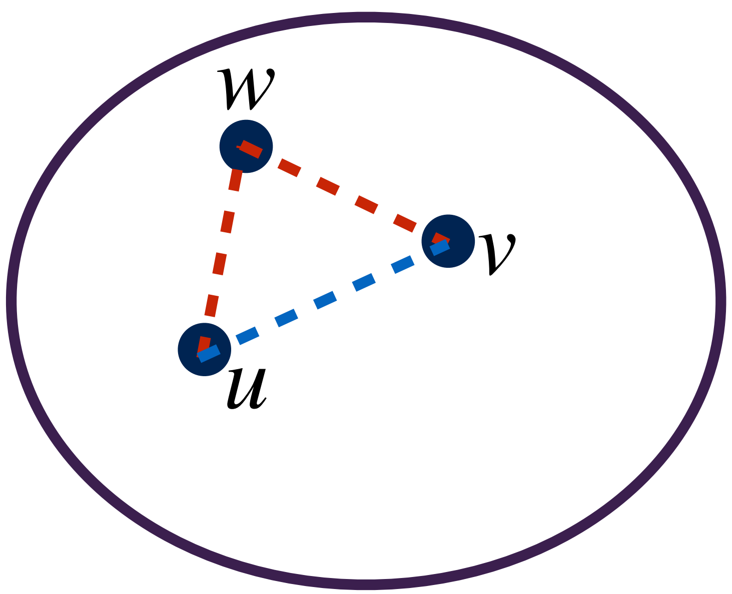
Endpoints of the edges in G^2 are in different parts



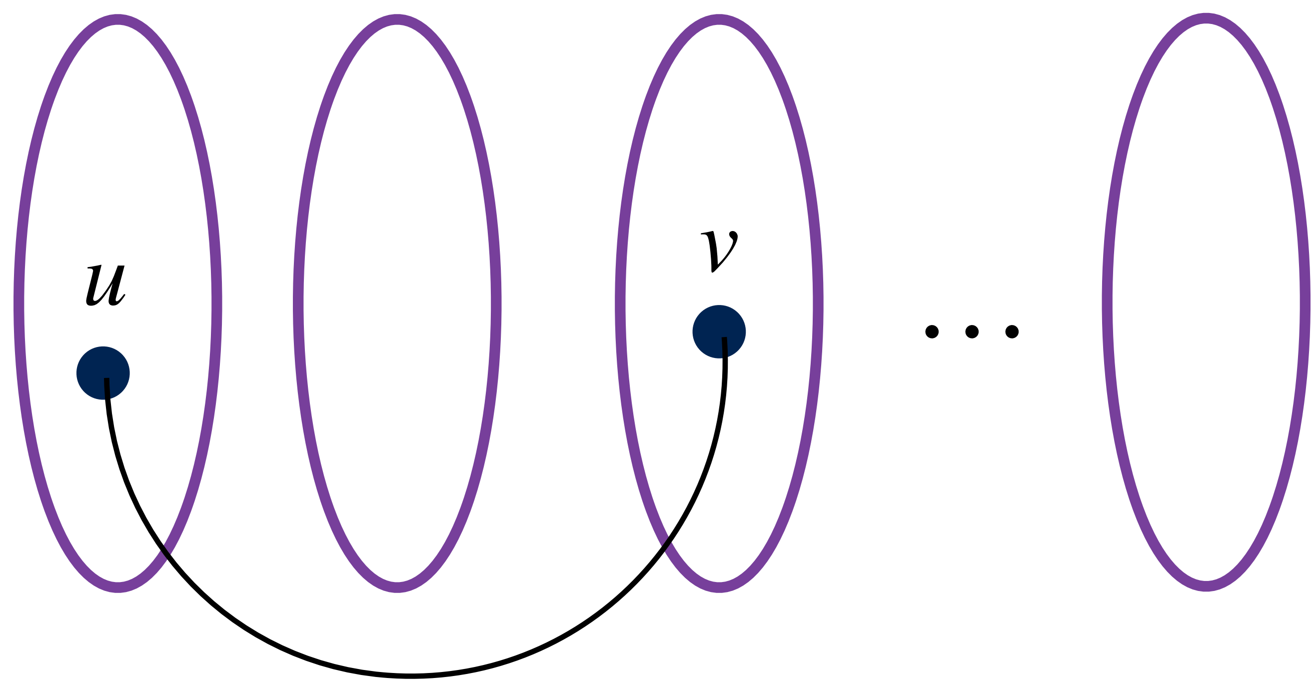
Structural properties of the input graph may be lost

Square Coloring

Graph G



$C_1 \uplus C_2 \uplus C_3 \cdots \uplus C_q = V(G)$

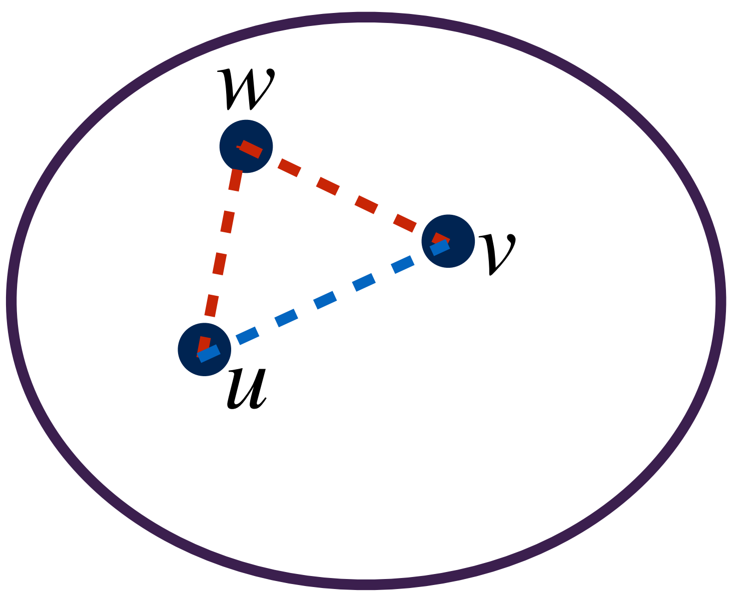


and integer q

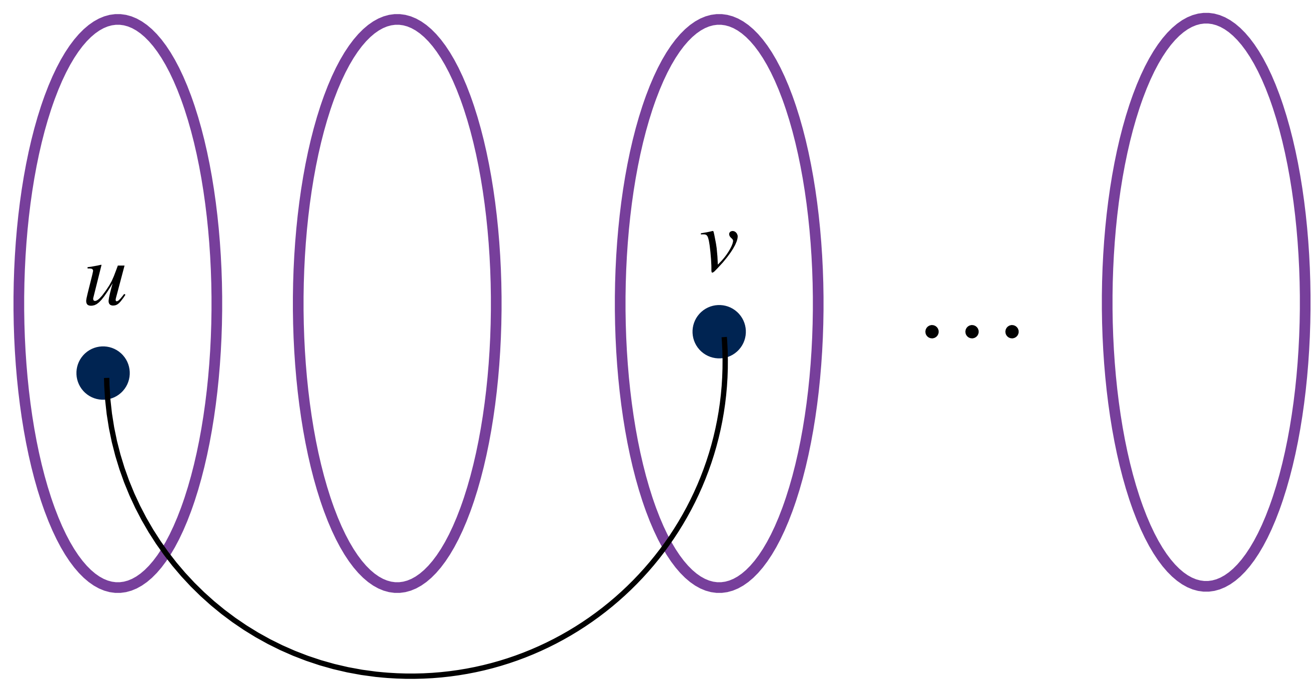
Endpoints of the edges in G^2 are in different parts

Square Coloring

Graph G



$$C_1 \uplus C_2 \uplus C_3 \cdots \uplus C_q = V(G)$$



and integer q

Endpoints of the edges in G^2 are in different parts

Exploit the properties of the input graph to obtain fast algorithms

Parameterized Problems

* Classical problem coupled with an integer, called the parameter, with each of the instance.

These integers measure a certain property of the input or output, or both

EXAMPLES

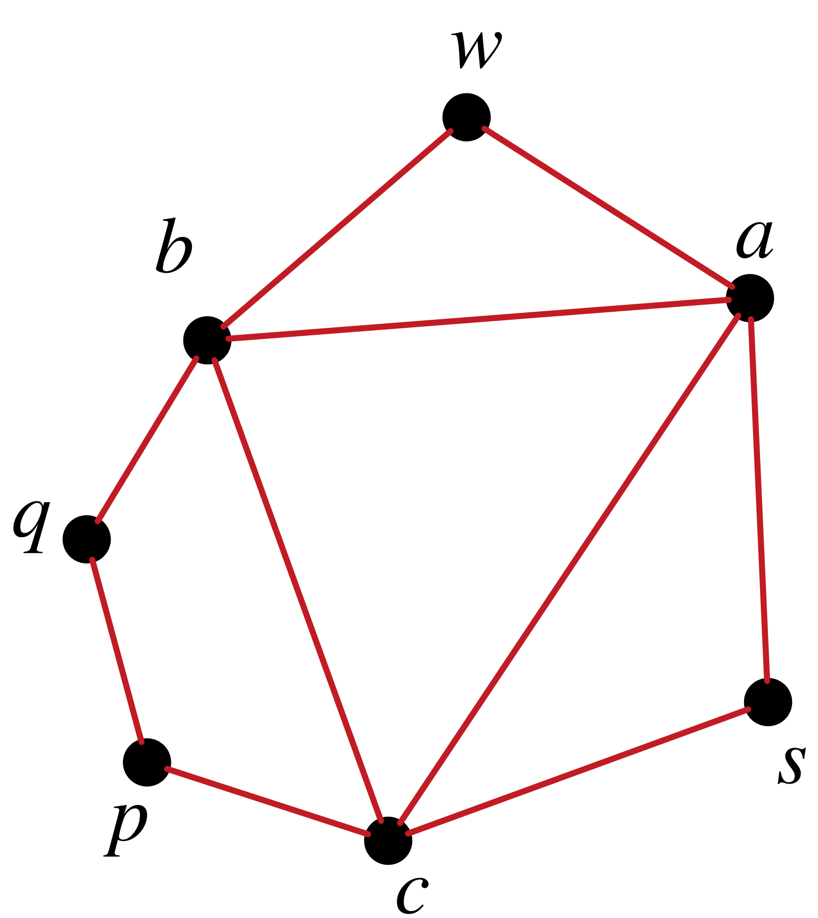
- Most classical one–the input size
- Radix Sort: Maximum number of bits
- How tree-like is the input graph

Parameterized Problems

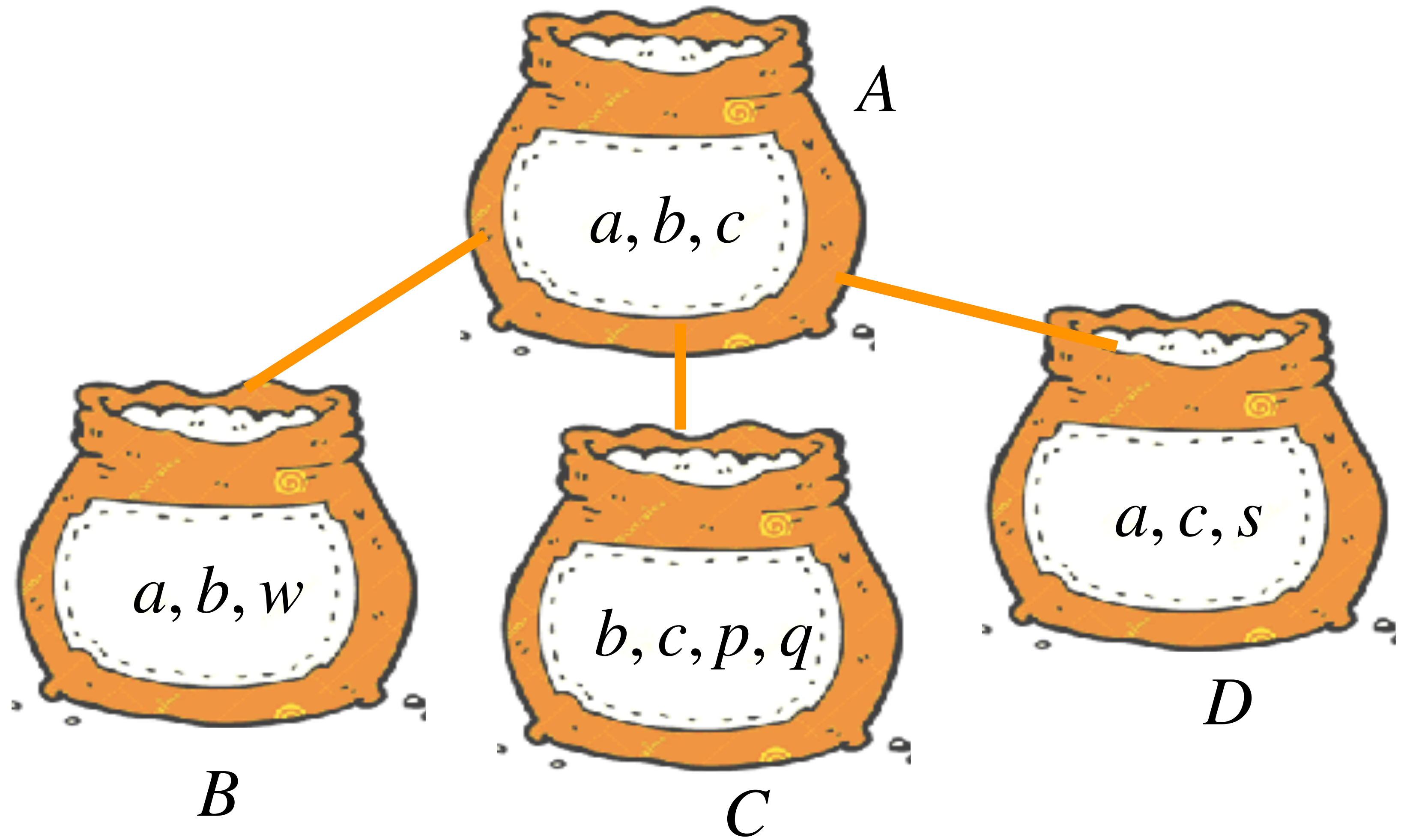
Fixed Parameter Tractable (FPT) Algorithms:

An algorithm for a parameterized problem running in time $f(k) \cdot \text{poly}(n)$, where k is the parameter and n is the input size.

Tree Decomposition & Treewidth



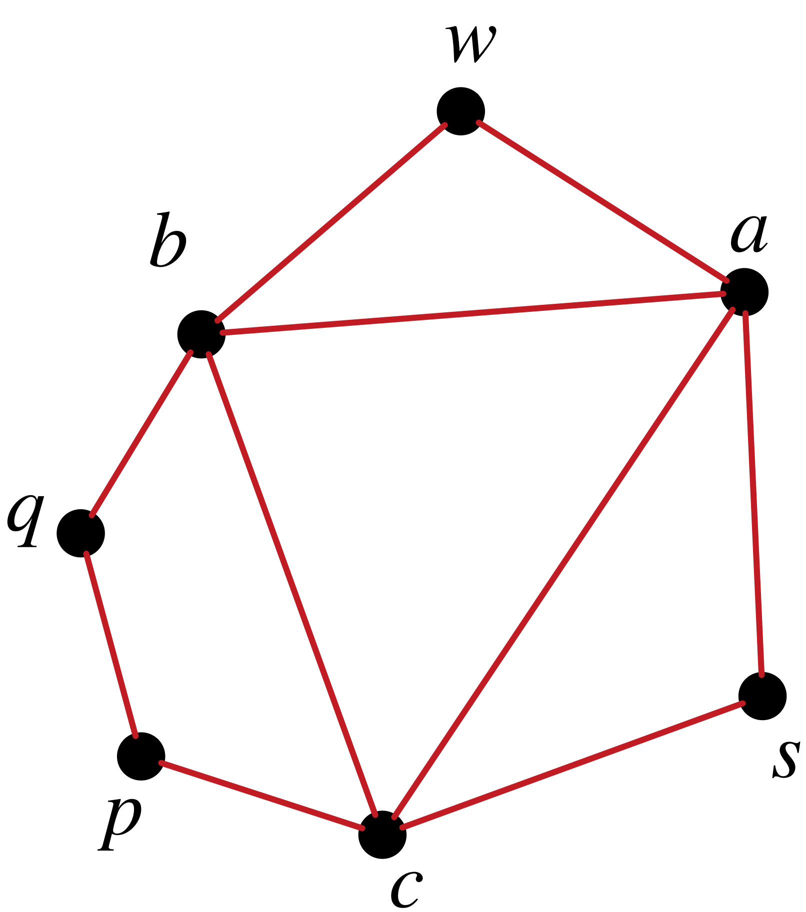
Graph G



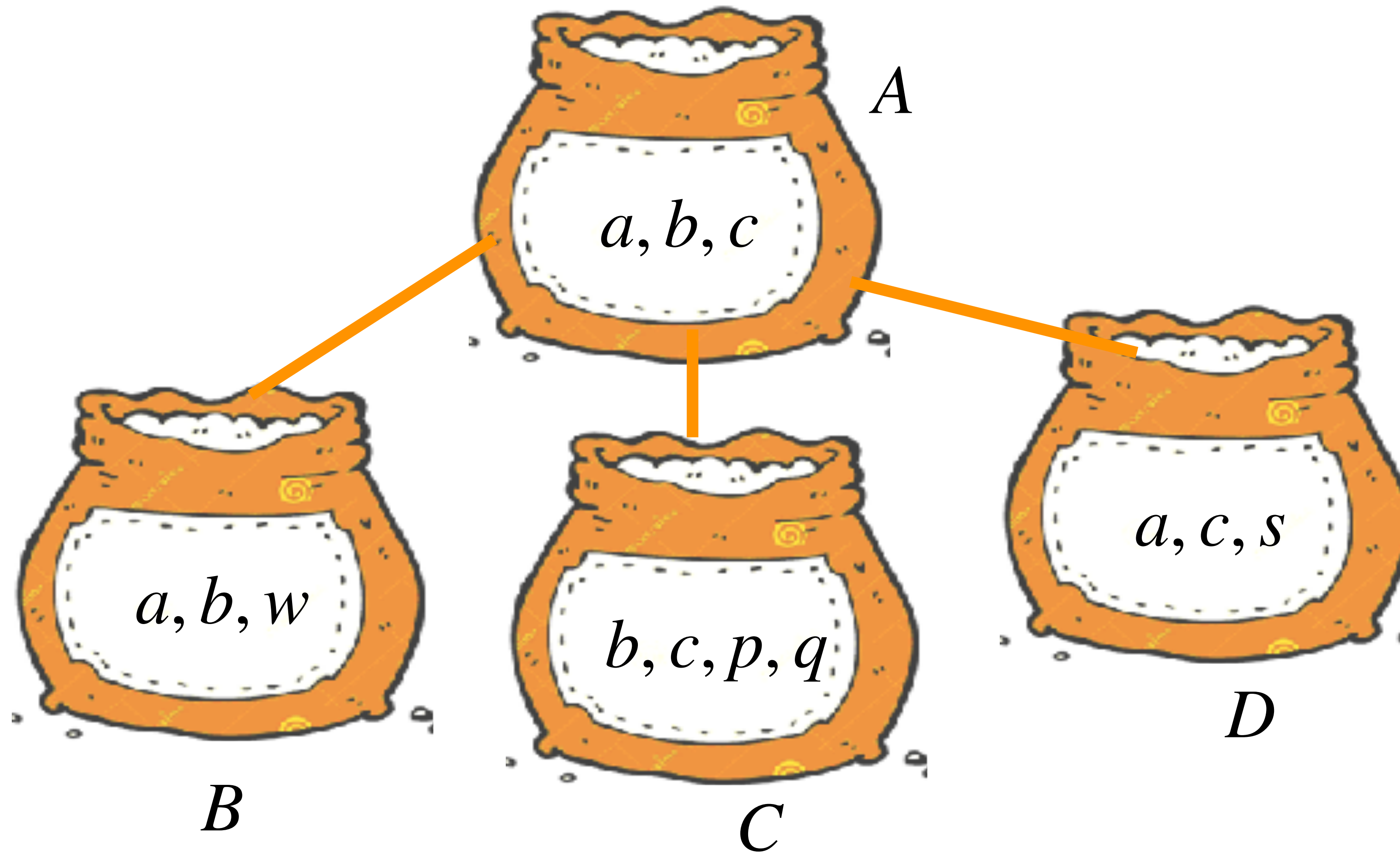
Tree Decomposition $(T, \beta : V(T) \rightarrow 2^{V(G)})$

- Each vertex is contained in at least one bag.
- Both endpoints of an edge are contained in some bag.
- The set of bags containing a vertex forms a connected subtree.

Tree Decomposition & Treewidth



Graph G



Tree Decomposition $(T, \beta : V(T) \to 2^{V(G)})$

- Each vertex is contained in at least one bag.
- Both endpoints of an edge are contained in some bag.
- The set of bags containing a vertex forms a connected subtree.

Width of (T, β): Maximum bag size - 1
Treewidth, $\mathrm{tw}(G)$: best such width

Tree Decomposition & Treewidth

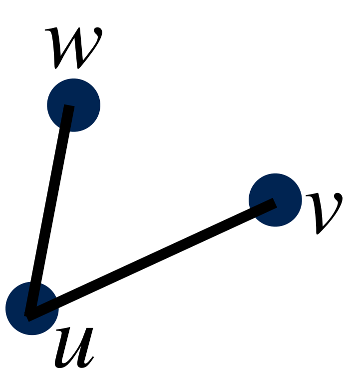
Easy to check:

If G is a star, its treewidth is 1 and the treewidth of G^2 is $n-1$

A Few Of The Prior Results

	Coloring	Square Coloring
NP-completeness:	For each fixed $q \geq 3$	For each fixed $q \geq 4$
Some polynomial time cases:	For each $q \leq 2$	For each $q \leq 3$

Note a graph G with max. degree Δ, for G^2 we need at least $\Delta + 1$ colors. Thus, for $q \leq 3$, G must be of paths and cycles



A Few Of The Prior Results

	Coloring	Square Coloring
NP-completeness:	For each fixed $q \geq 3$	For each fixed $q \geq 4$
Some polynomial time cases:	For each $q \leq 2$	For each $q \leq 3$
	Interval graphs	Interval graphs
	Chordal graphs	Trees

A Few Of The Prior Results

	Coloring	Square Coloring
FPT Results:	$q^{O(\mathrm{tw})} \cdot n$	
	$\Downarrow$	
	$2^{O(\mathrm{tw} \log \mathrm{tw})} \cdot n$	

For q can be assumed to be at most $\mathrm{tw} + 1$

A Few Of The Prior Results

	Coloring	Square Coloring
FPT Results:	$2^{O(\mathrm{tw} \log \mathrm{tw})} \cdot n$	

A Few Of The Prior Results

	Coloring	Square Coloring
FPT Results:	$2^{O(\mathrm{tw}\log \mathrm{tw})} \cdot n$	$(q+1)^{2^{8\mathrm{tw}}+1} \cdot n^{O(1)}$

A Few Of The Prior Results

	Coloring	Square Coloring
FPT Results:	$2^{O(\mathrm{tw} \log \mathrm{tw})} \cdot n$	$(q+1)^{2^{8\mathrm{tw}}+1} \cdot n^{O(1)}$
		No FPT algorithm param. by treewidth

- Typically, for a parameterized problem either:
  - no FPT algorithm, but there is $n^{O(\mathrm{tw})}$-time algorithm
  - or, it has an FPT algorithm

Our Results For General Graphs

Square Coloring admits an algorithm running in time $(q+1)^{2^{\mathrm{tw}+4}} \cdot n^{O(1)}$.

The above algorithm is essentially the best under ETH:

Square Coloring has no $f(\mathrm{tw}) \cdot n^{(2-\epsilon)^{\mathrm{tw}}}$-time algorithm.

Our Results On Planar Graphs

Planar Square Coloring is NP-hard for each fixed $q \geq 4$.

NOTE: Four Color Theorem => for each $q \geq 4$,
Planar Coloring is in polynomial time

Our Results On Planar Graphs

	Coloring	Square Coloring
FPT Results:	$2^{O(\mathrm{tw}\log\mathrm{tw})}\cdot n$	$(q+1)^{2^{8\mathrm{tw}}+1}\cdot n^{O(1)}$
Planar	$2^{O(\sqrt{n})}$	$2^{O(n\log n)}$

Planar Square Coloring is NP-hard for each fixed $q \geq 4$.

Our Results On Planar Graphs

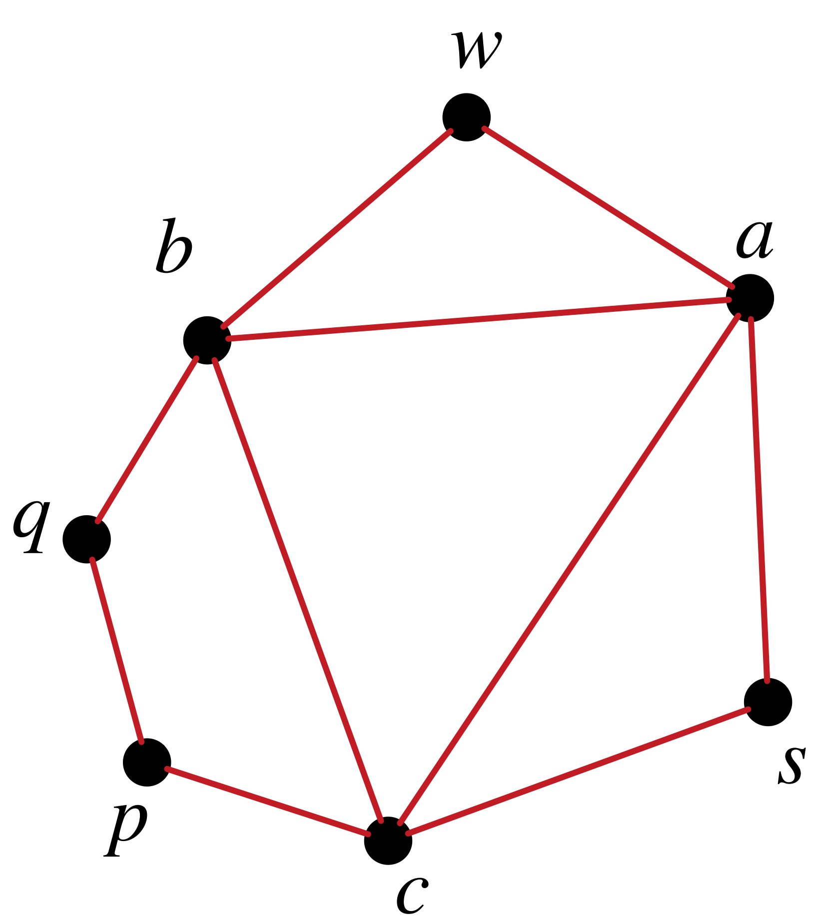
	Coloring	Square Coloring
FPT Results:	$2^{O(\mathrm{tw}\log \mathrm{tw})} \cdot n$	$(q+1)^{2^{8\mathrm{tw}}+1} \cdot n^{O(1)}$
Planar	$2^{O(\sqrt{n})}$	$2^{O(n \log n)}$

Planar Square Coloring is NP-hard for each fixed $q \geq 4$.

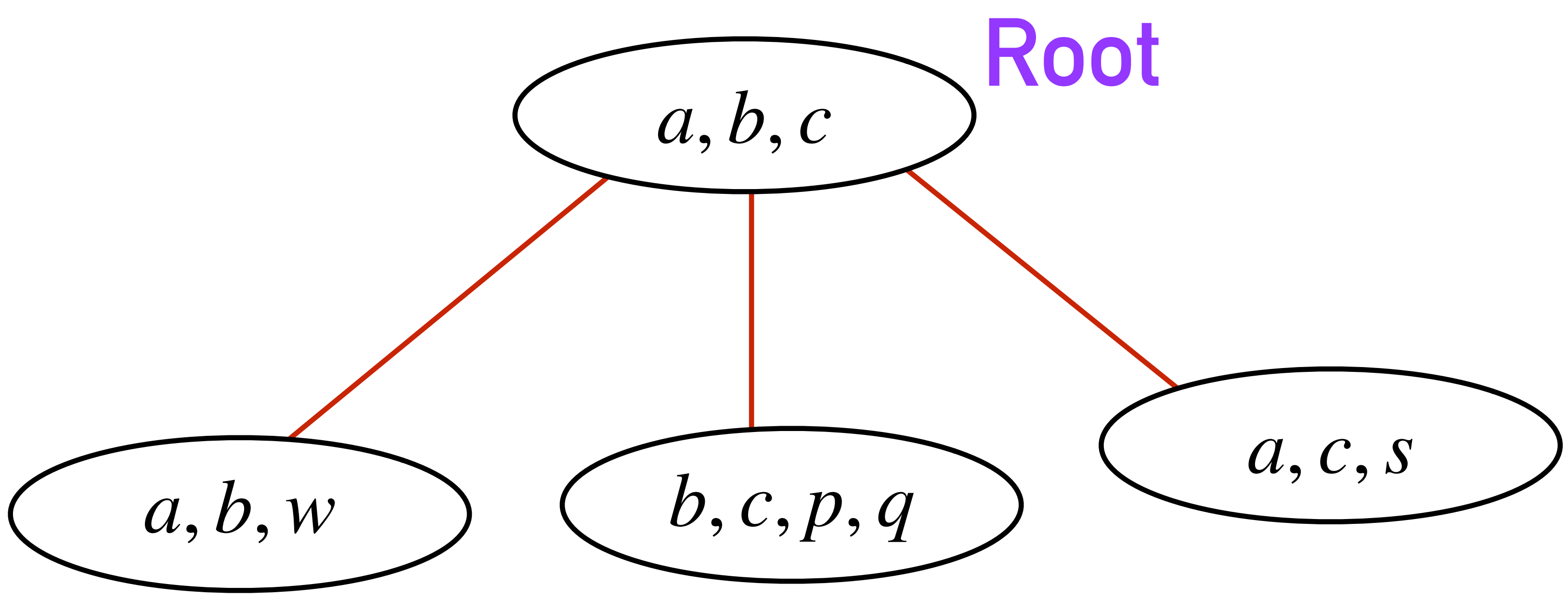
Planar Square Coloring has $2^{O(n^{2/3}\log n)}$-time algorithm.

Algorithm for Coloring param. by treewidth:
dynamic programming over tree decomposition

Recalling Tree Decomposition



Graph G



Tree Decomposition $(T, \beta : V(T) \to 2^{V(G)})$

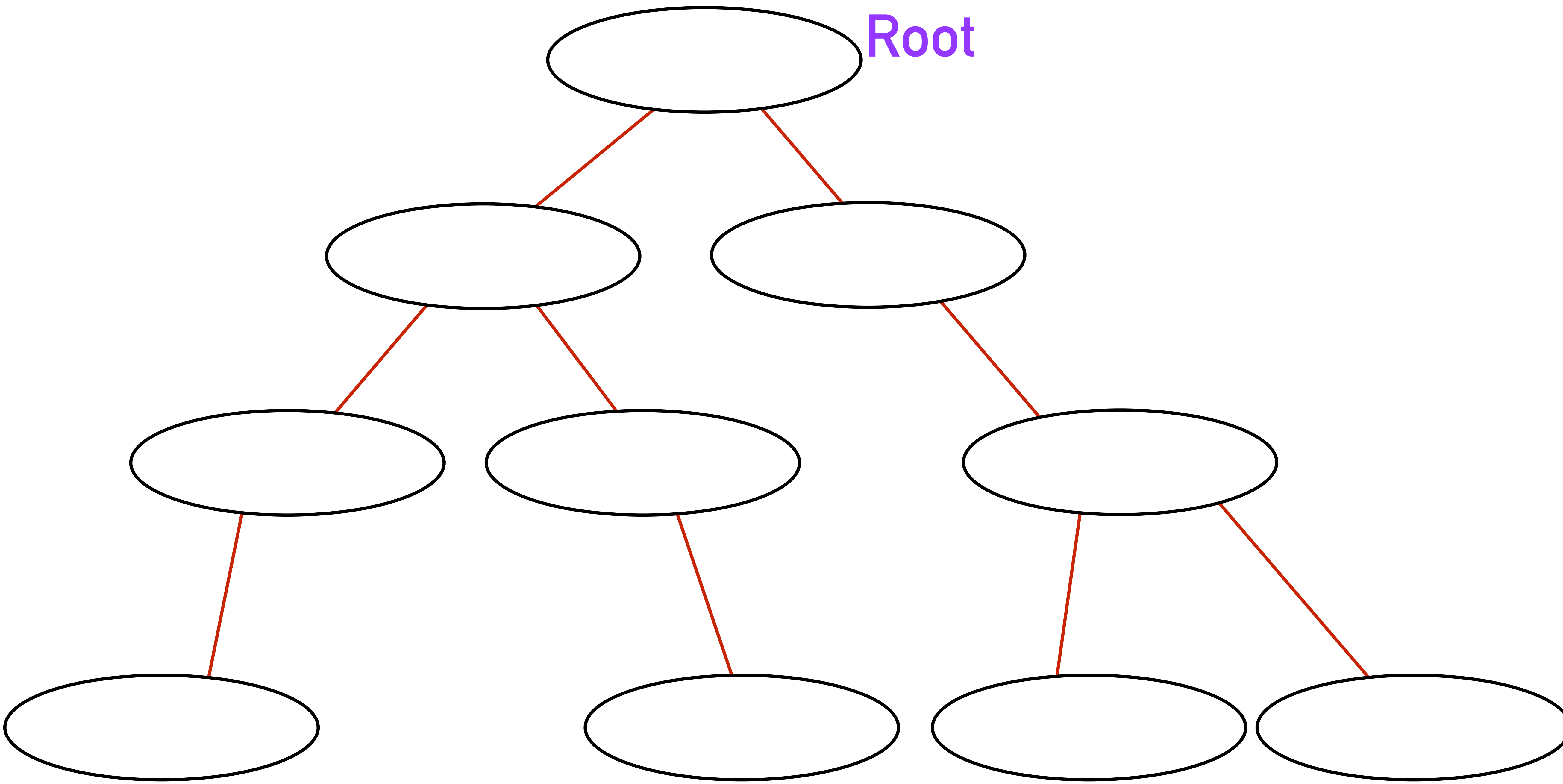
- Each vertex is contained in at least one bag.
- Both endpoints of an edge are contained in some bag.
- The set of bags containing a vertex forms a connected subtree.

Width of (T, β): Maximum bag size - 1

Treewidth, $\mathrm{tw}(G)$: best such width

Algorithm for Coloring

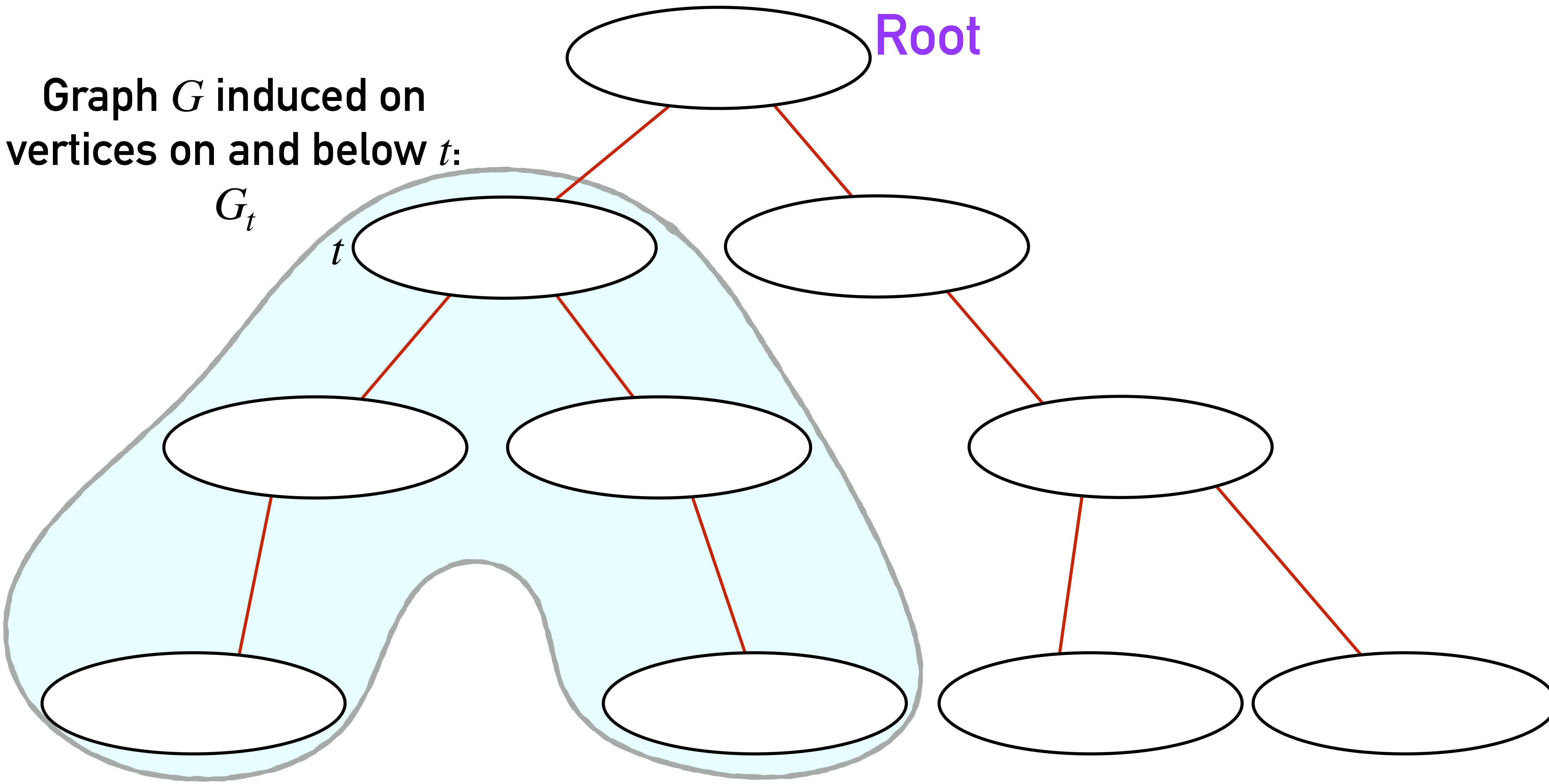
Tree Decomposition $(T, \beta : V(T) \to 2^{V(G)})$ for graph G



- We go in a bottom up-fashion, starting from the leaves.

Algorithm for Coloring

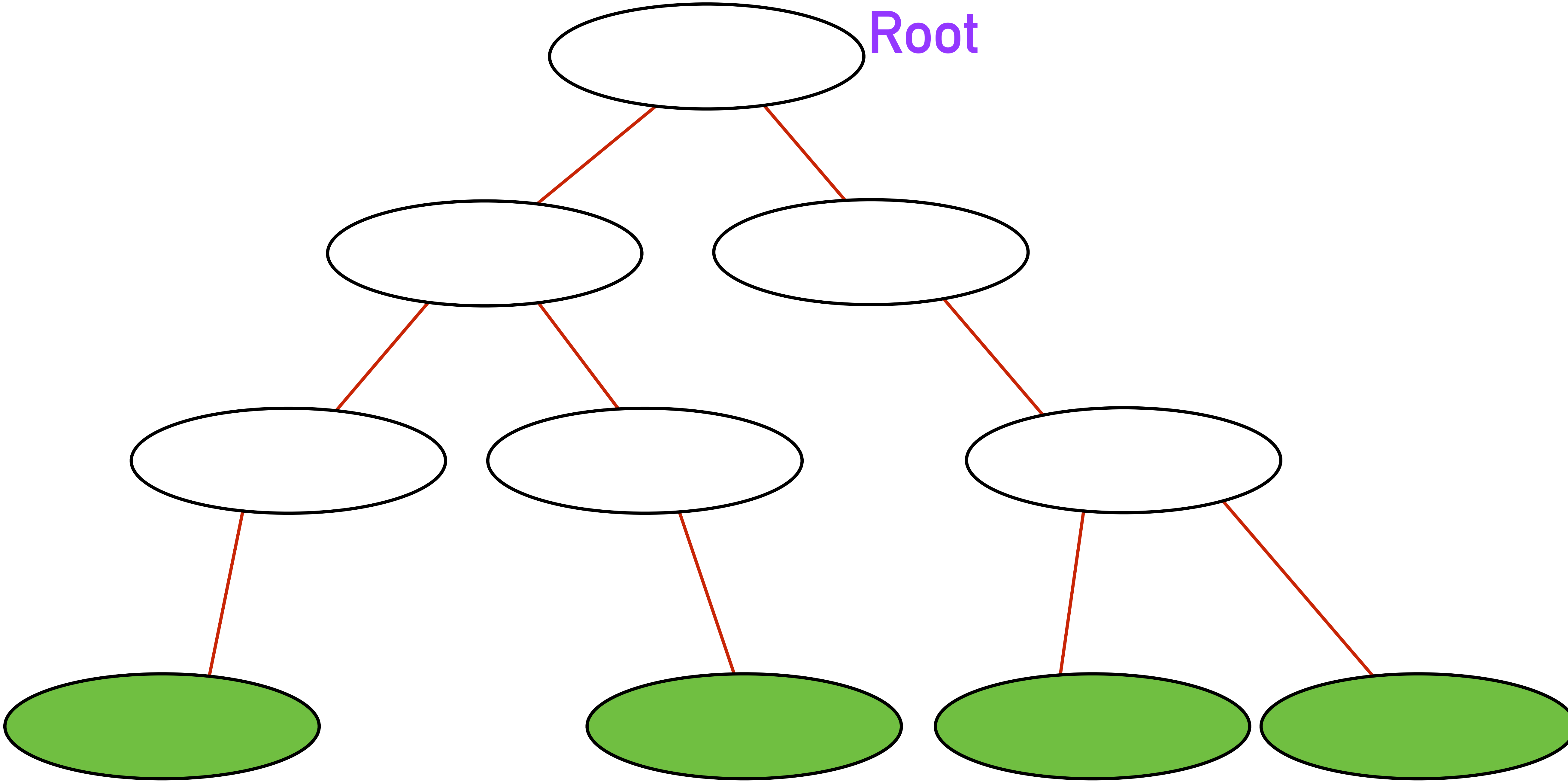
Tree Decomposition $(T, \beta : V(T) \to 2^{V(G)})$ for graph G



- We go in a bottom up-fashion, starting from the leaves.

Algorithm for Coloring

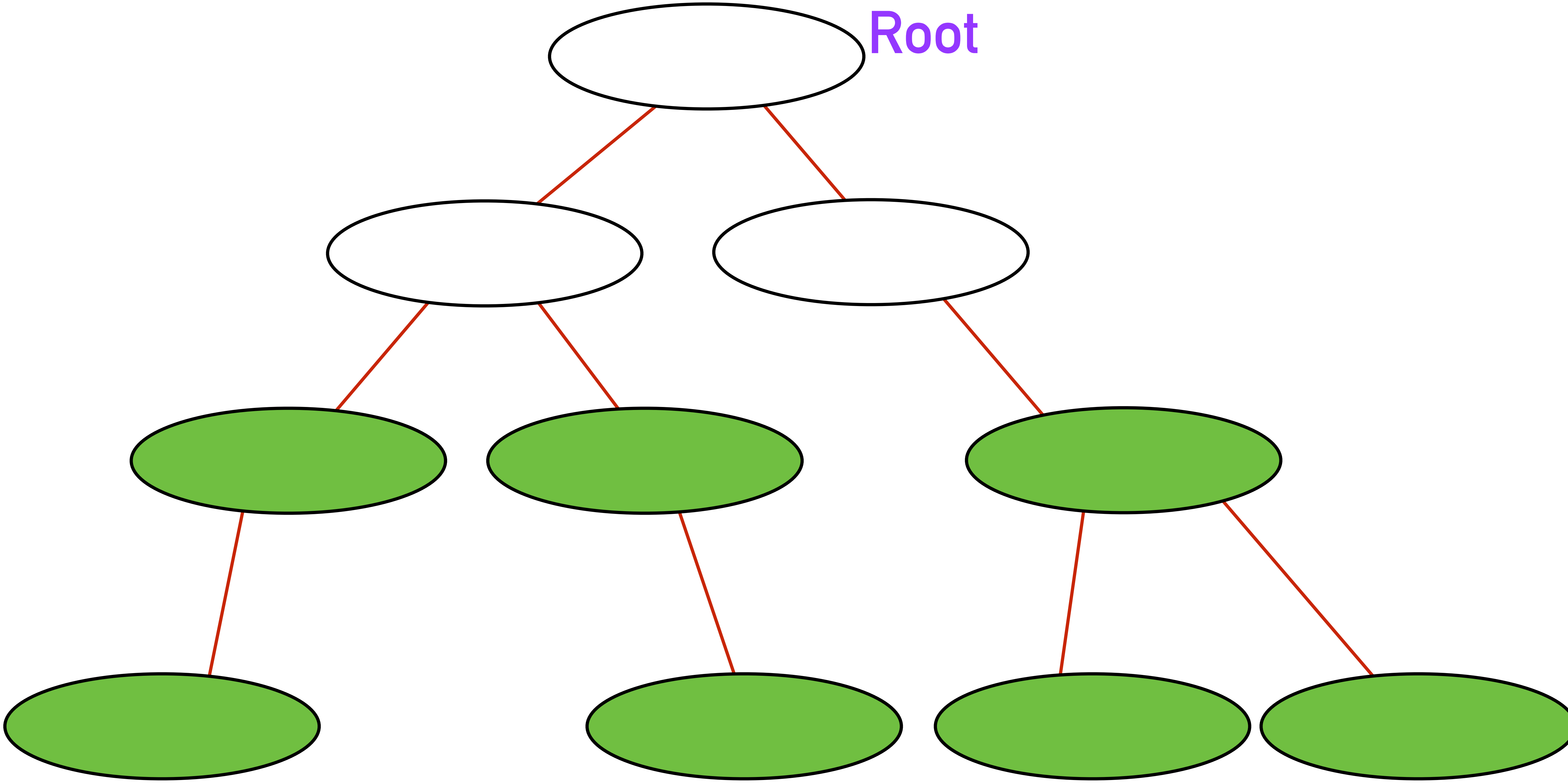
Tree Decomposition $(T, \beta : V(T) \rightarrow 2^{V(G)})$ for graph G



We go in a bottom up-fashion, starting from the leaves.

Algorithm for Coloring

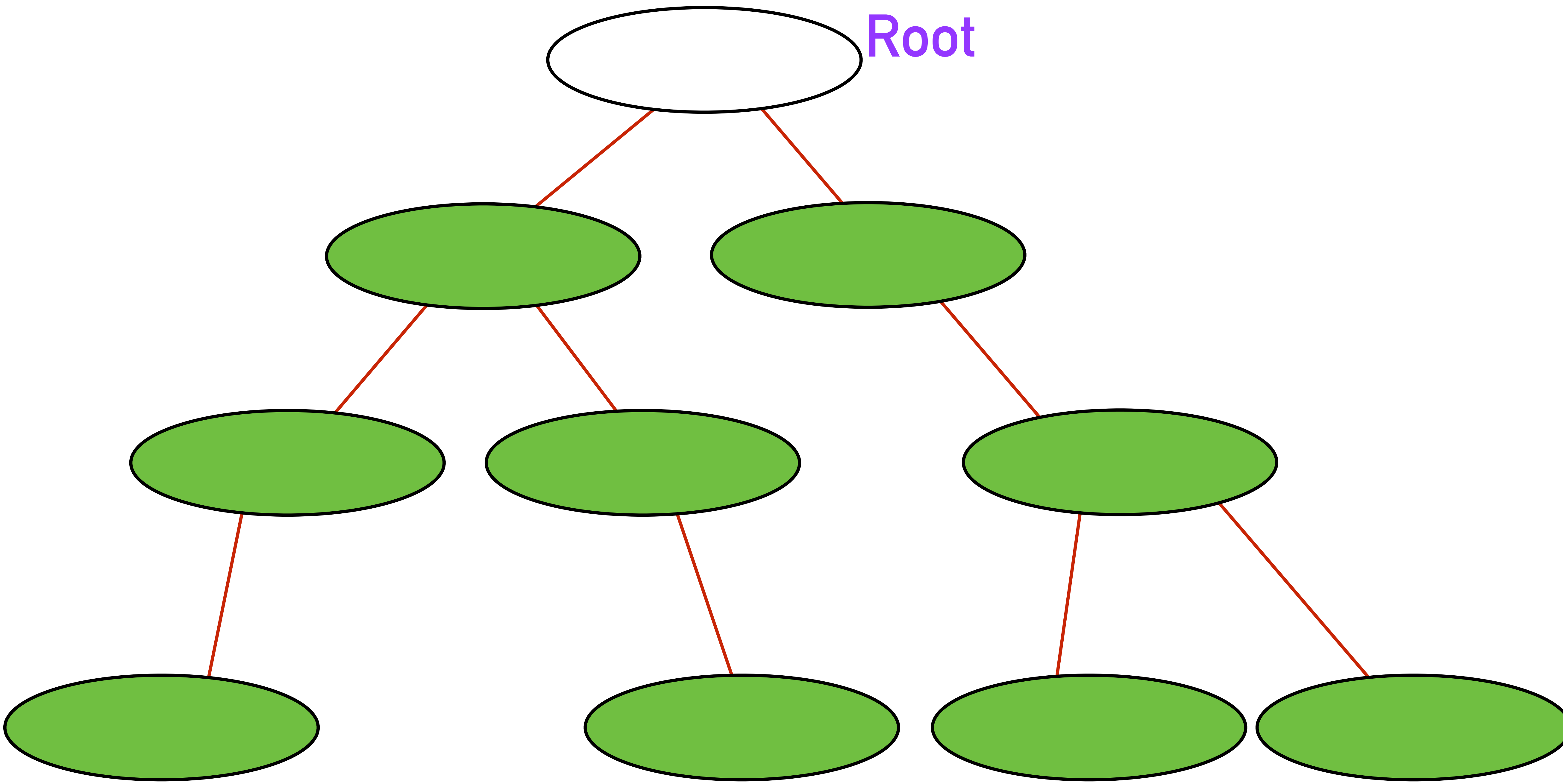
Tree Decomposition $(T, \beta : V(T) \to 2^{V(G)})$ for graph G



We go in a bottom up-fashion, starting from the leaves.

Algorithm for Coloring

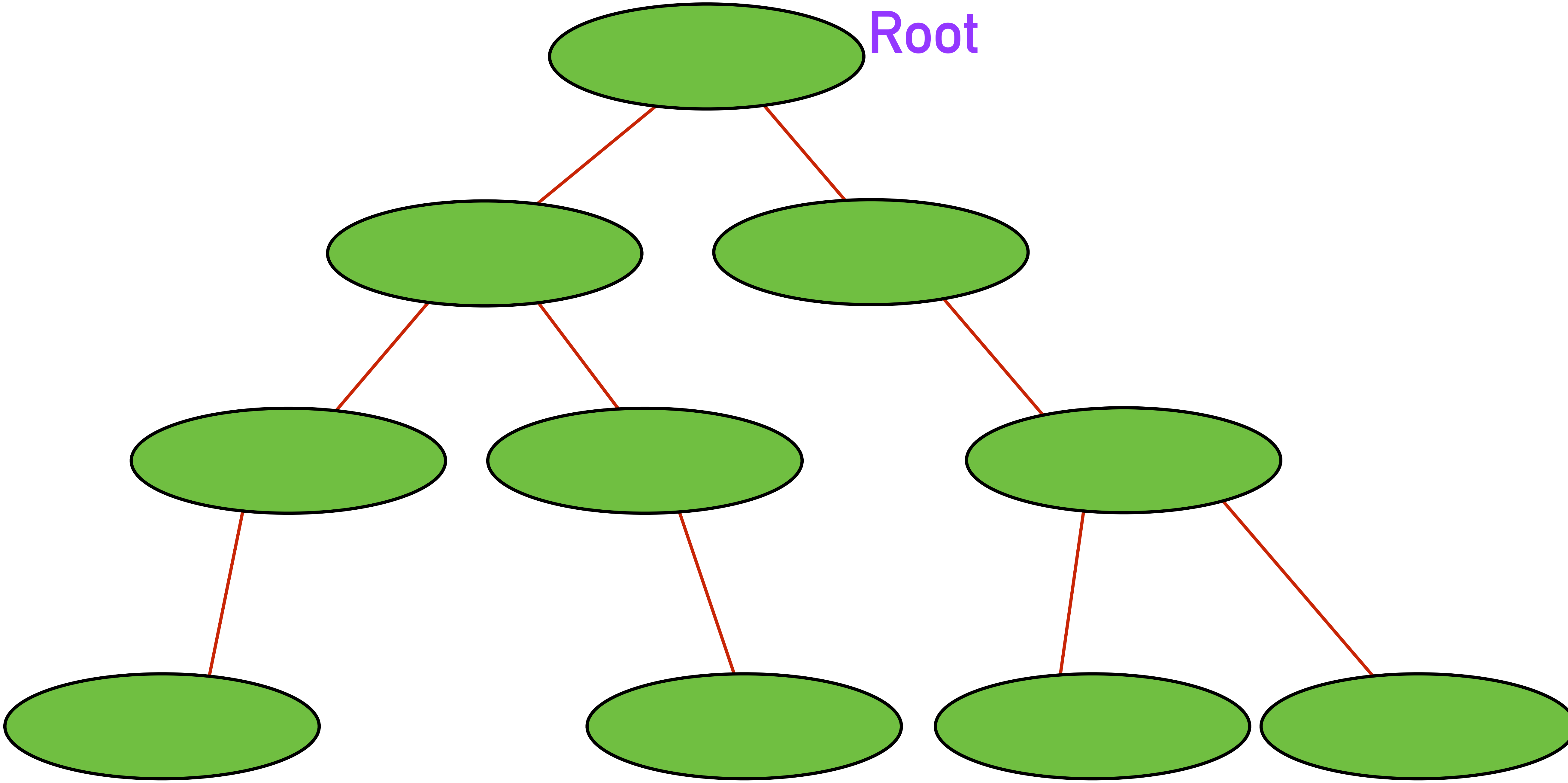
Tree Decomposition $(T, \beta : V(T) \to 2^{V(G)})$ for graph G



* We go in a bottom up-fashion, starting from the leaves.

Algorithm for Coloring

Tree Decomposition $(T, \beta : V(T) \to 2^{V(G)})$ for graph G

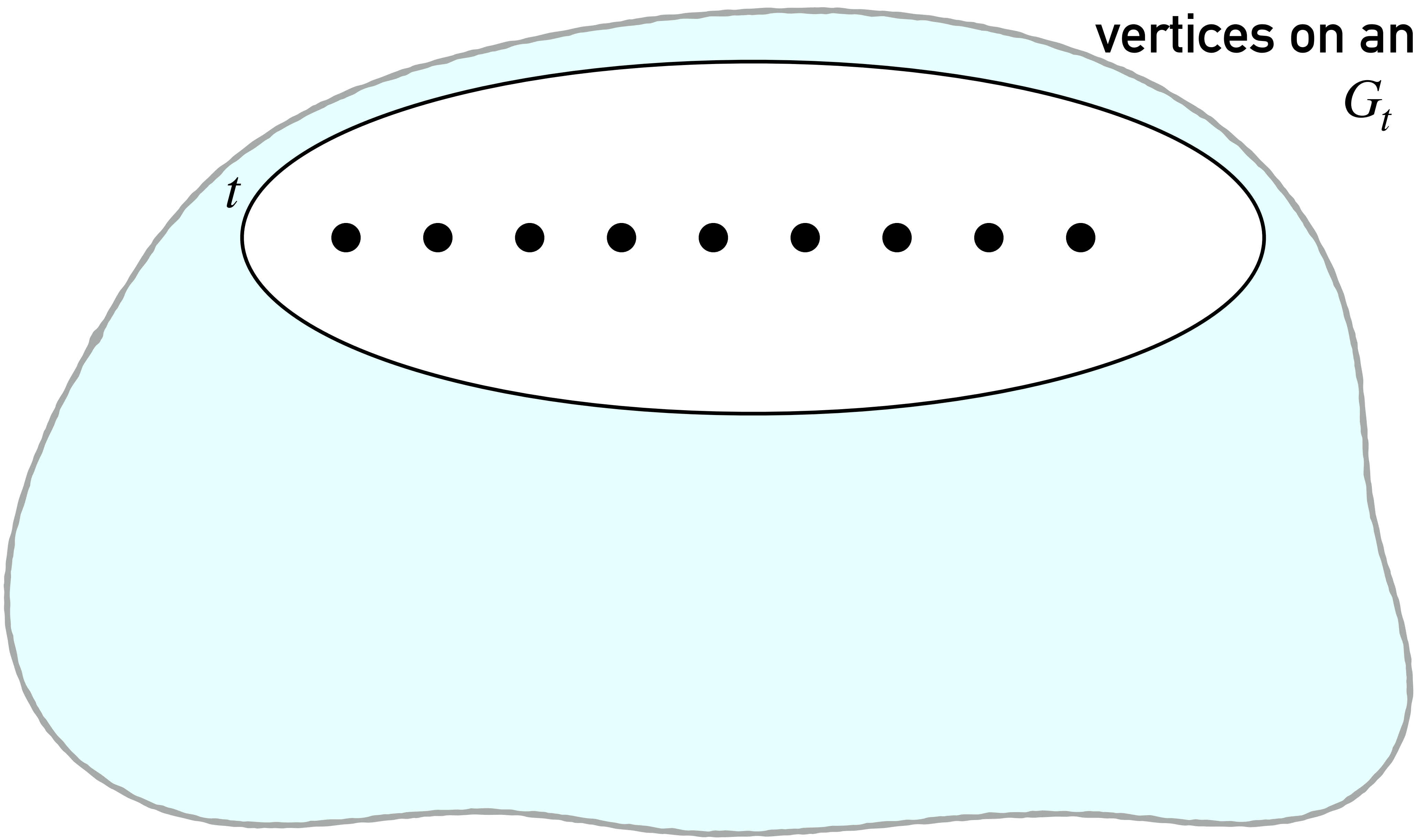


* We go in a bottom up-fashion, starting from the leaves.

Algorithm for Coloring

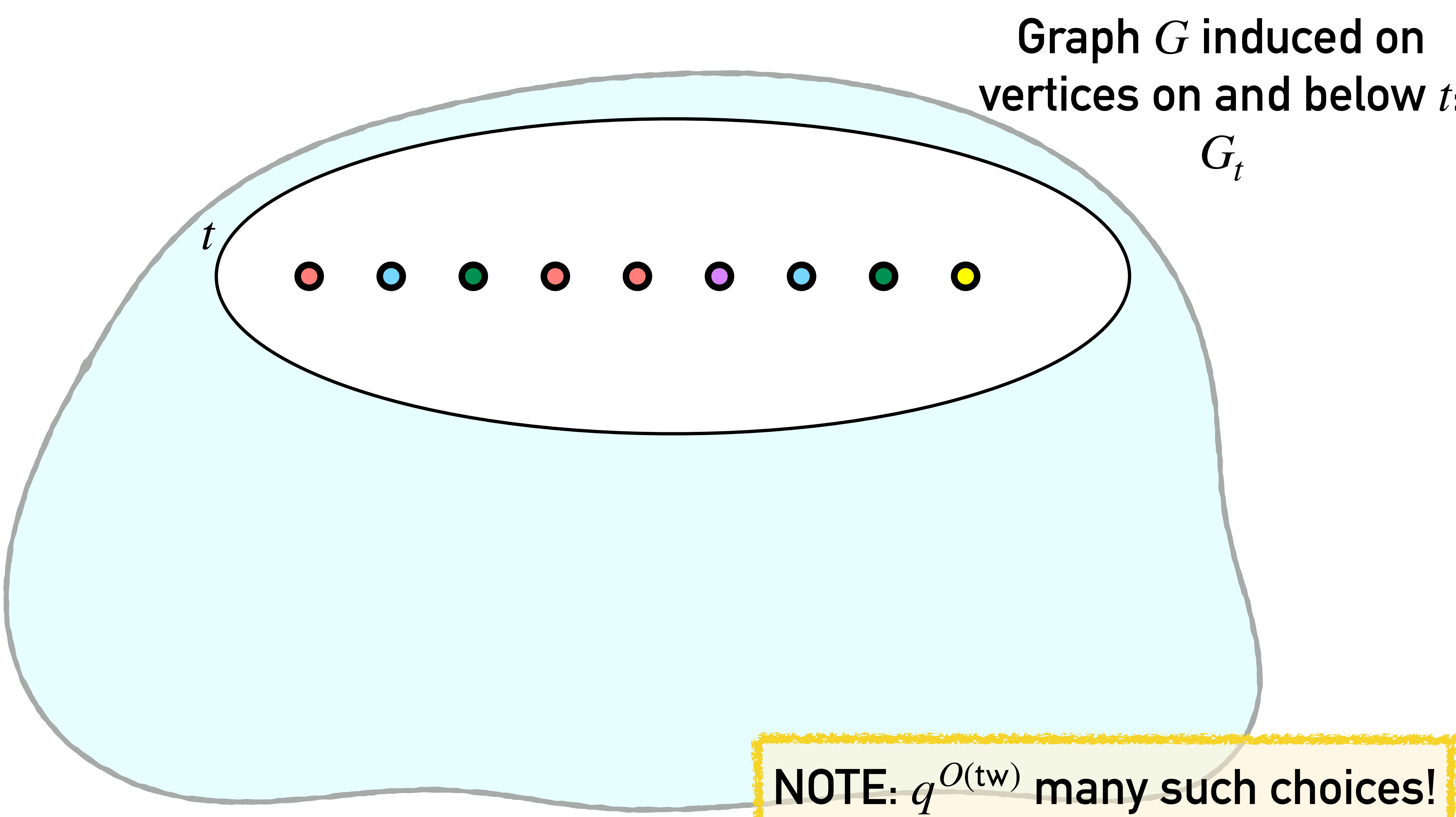
Solving for a bag, when all its descendant bags are resolved.

Graph G induced on vertices on and below t: G_t



Algorithm for Coloring

Solving for a bag, when all its descendant bags are resolved.

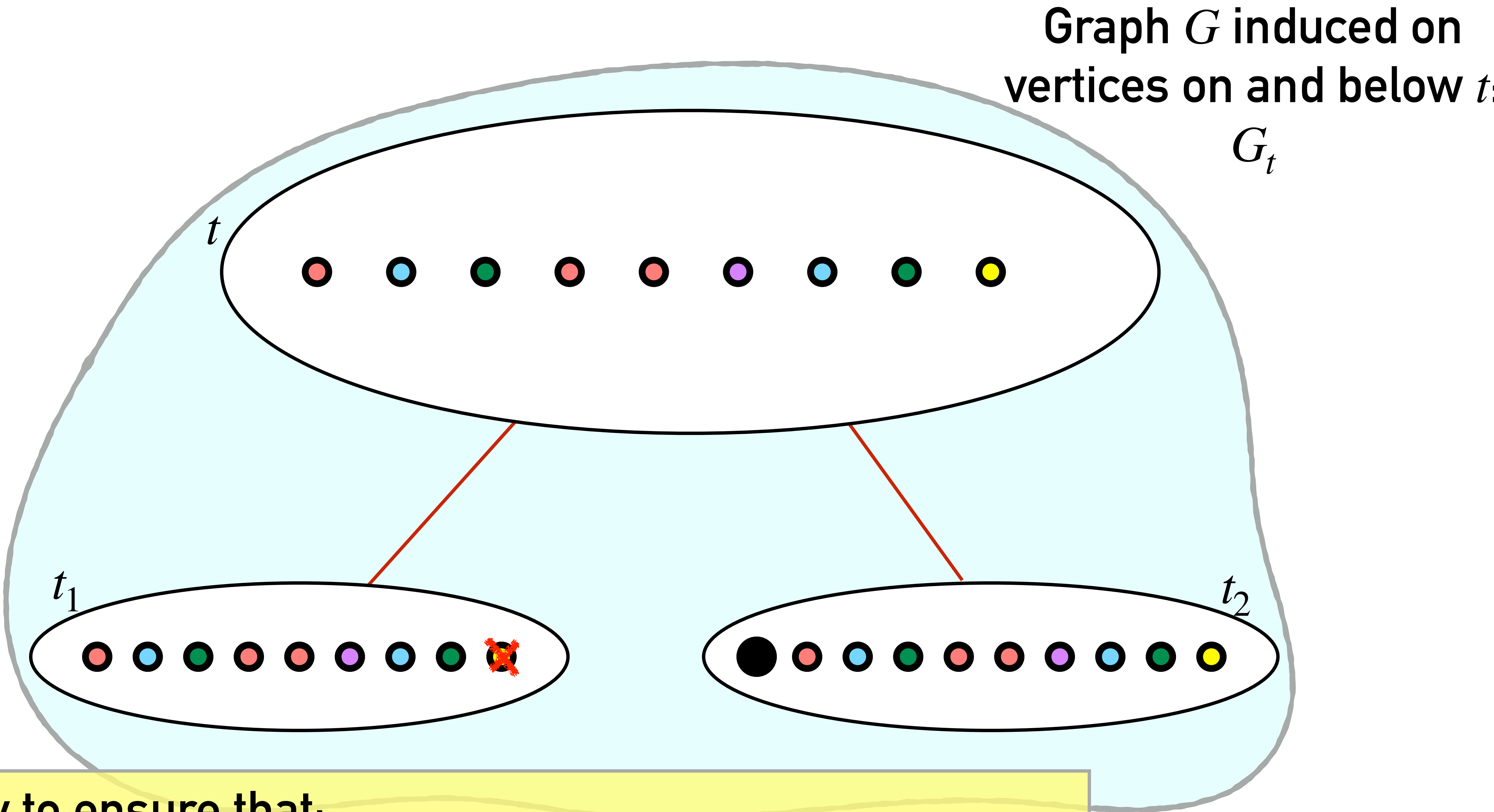


NOTE: $q^{O(\text{tw})}$ many such choices!

- For each $\chi : \beta(t) \rightarrow \{1,2,\cdots,q\}$, check if χ can be extended to a proper coloring for G_t.

Algorithm for Coloring

Solving for a bag, when all its descendant bags are resolved.



Way to ensure that:

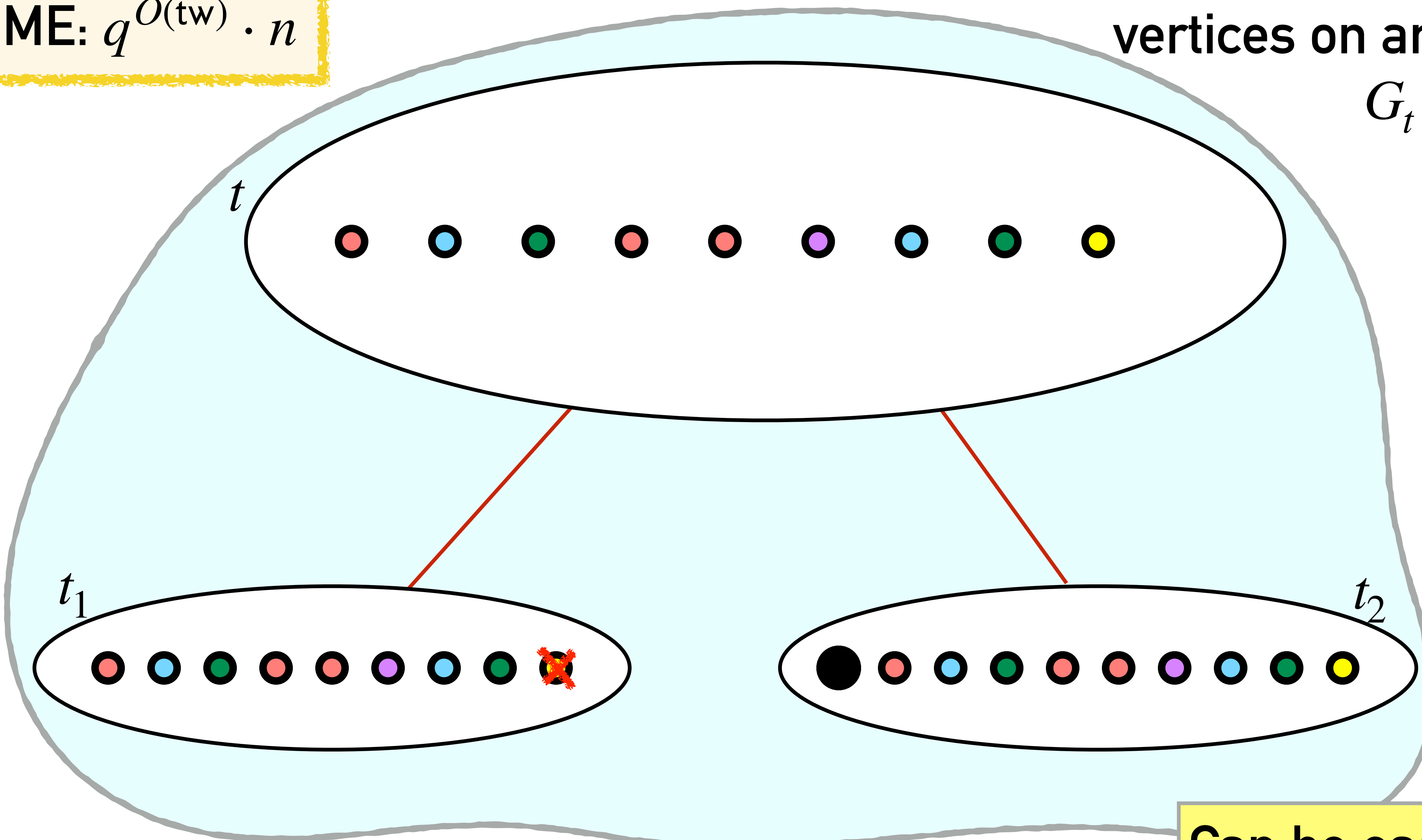
- t has at most 2 children
- $\beta(t)$ varies in at most one vertex from $\beta(t_1)$ and $\beta(t_2)$

Algorithm for Coloring

Solving for a bag, when all its descendant bags are resolved.

RUNTIME: $q^{O(\mathrm{tw})} \cdot n$

Graph G induced on vertices on and below t: G_t

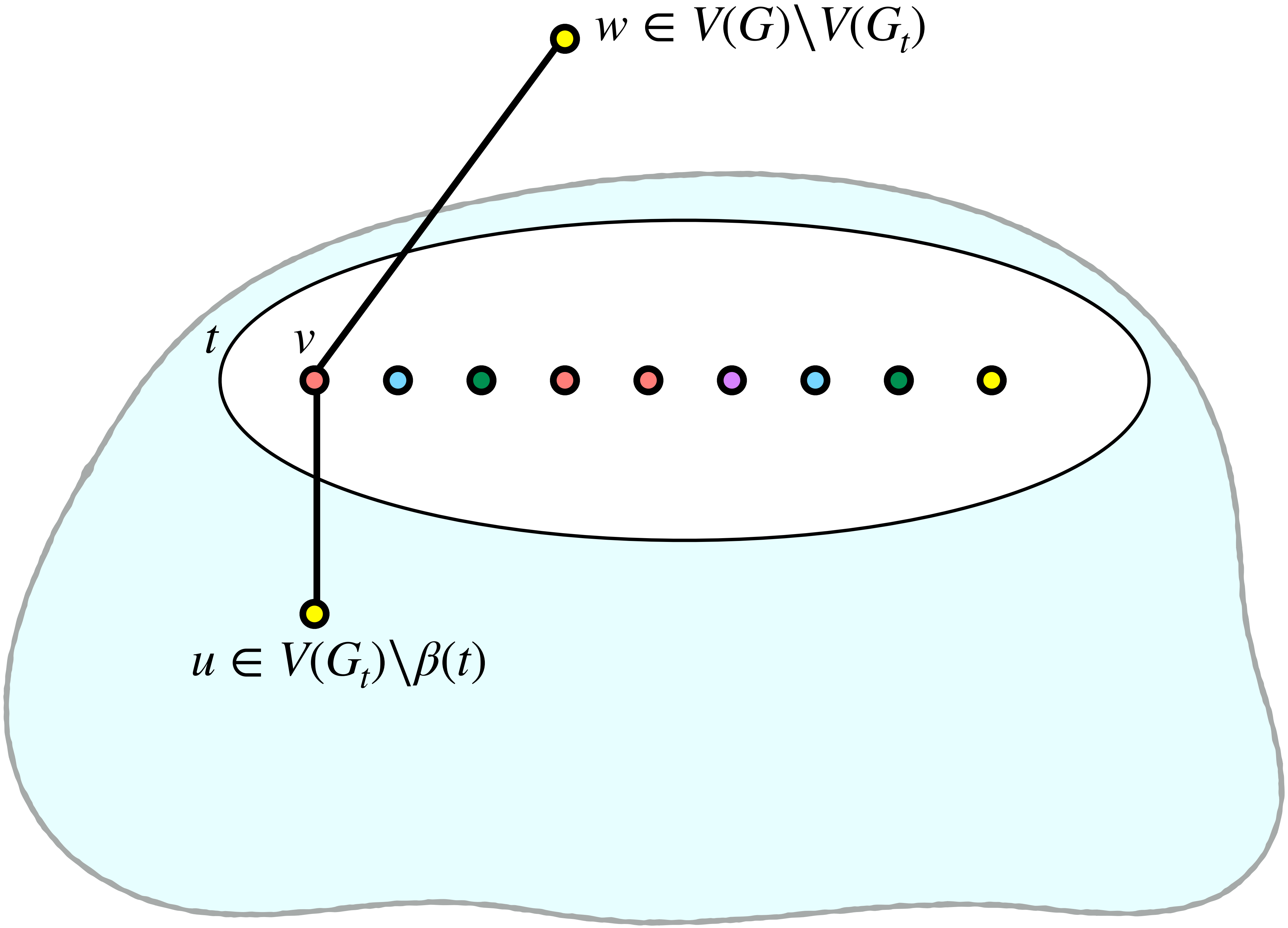


Can be easily done!

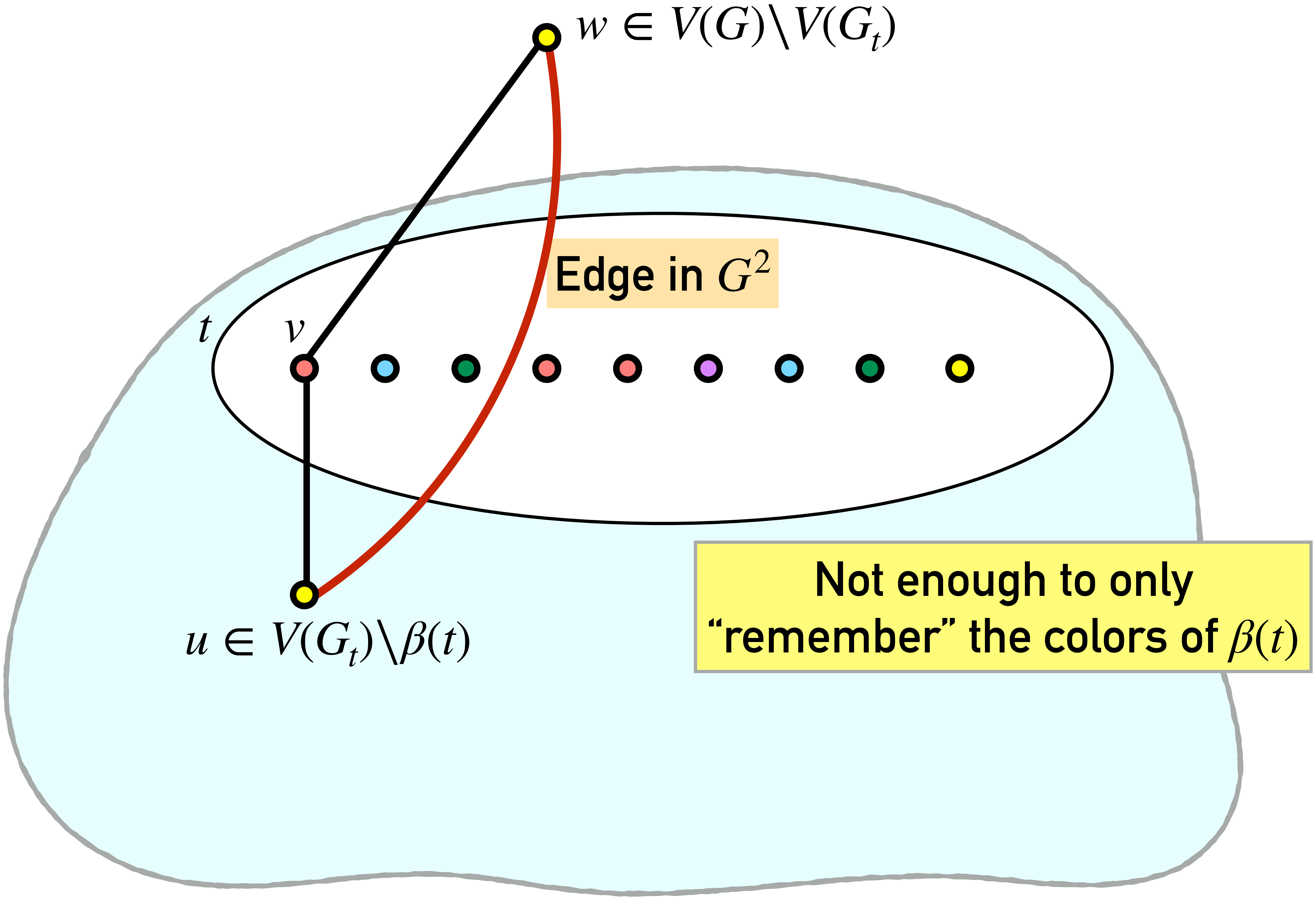
- For each $\chi : \beta(t) \to \{1,2,\cdots,q\}$, check if χ can be extended to a proper coloring for G_t.

Why the previous DP fails for Square Coloring?

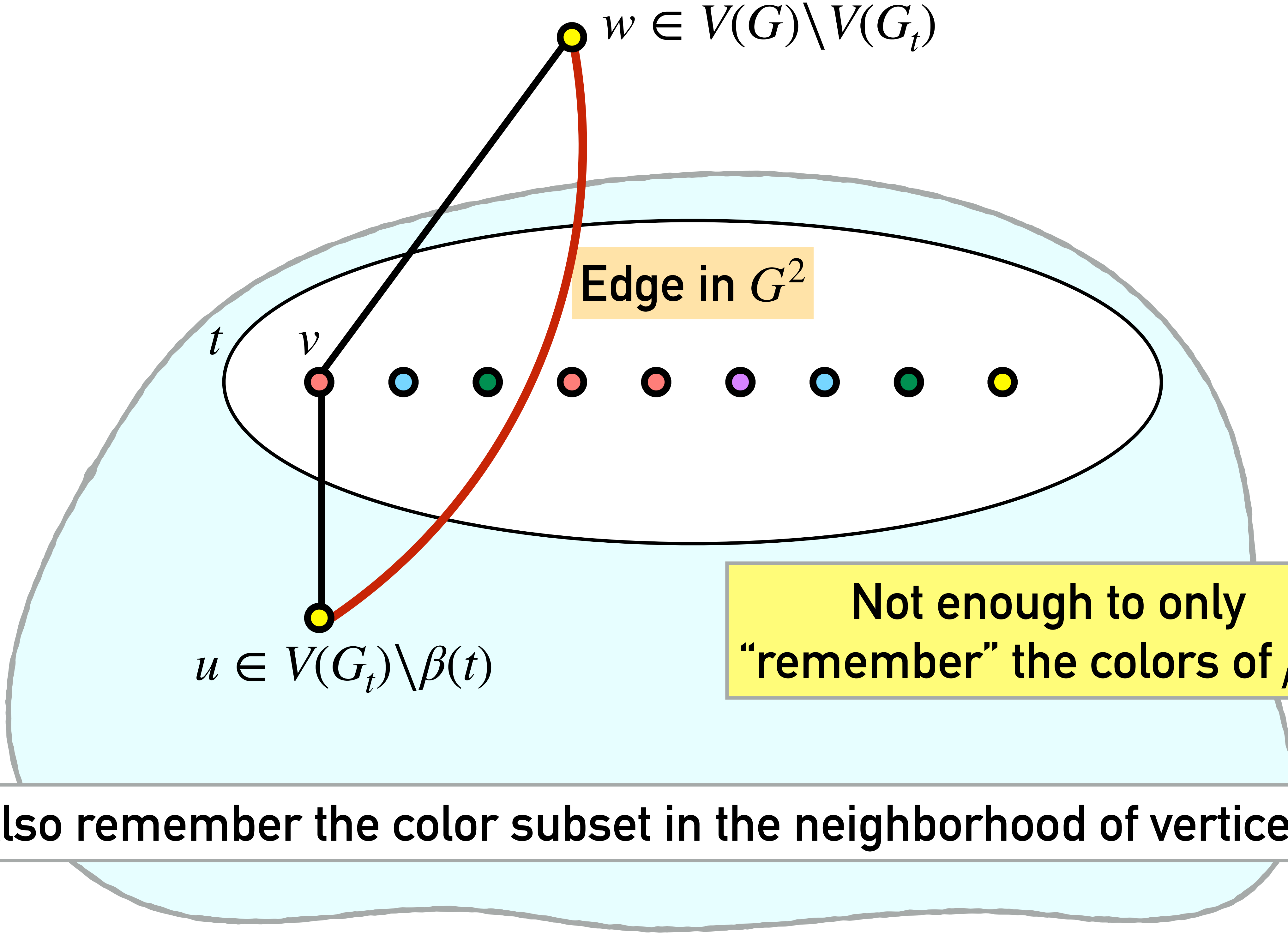
Failure for Square Coloring



Failure for Square Coloring



First Attempt At Fixing the Issue

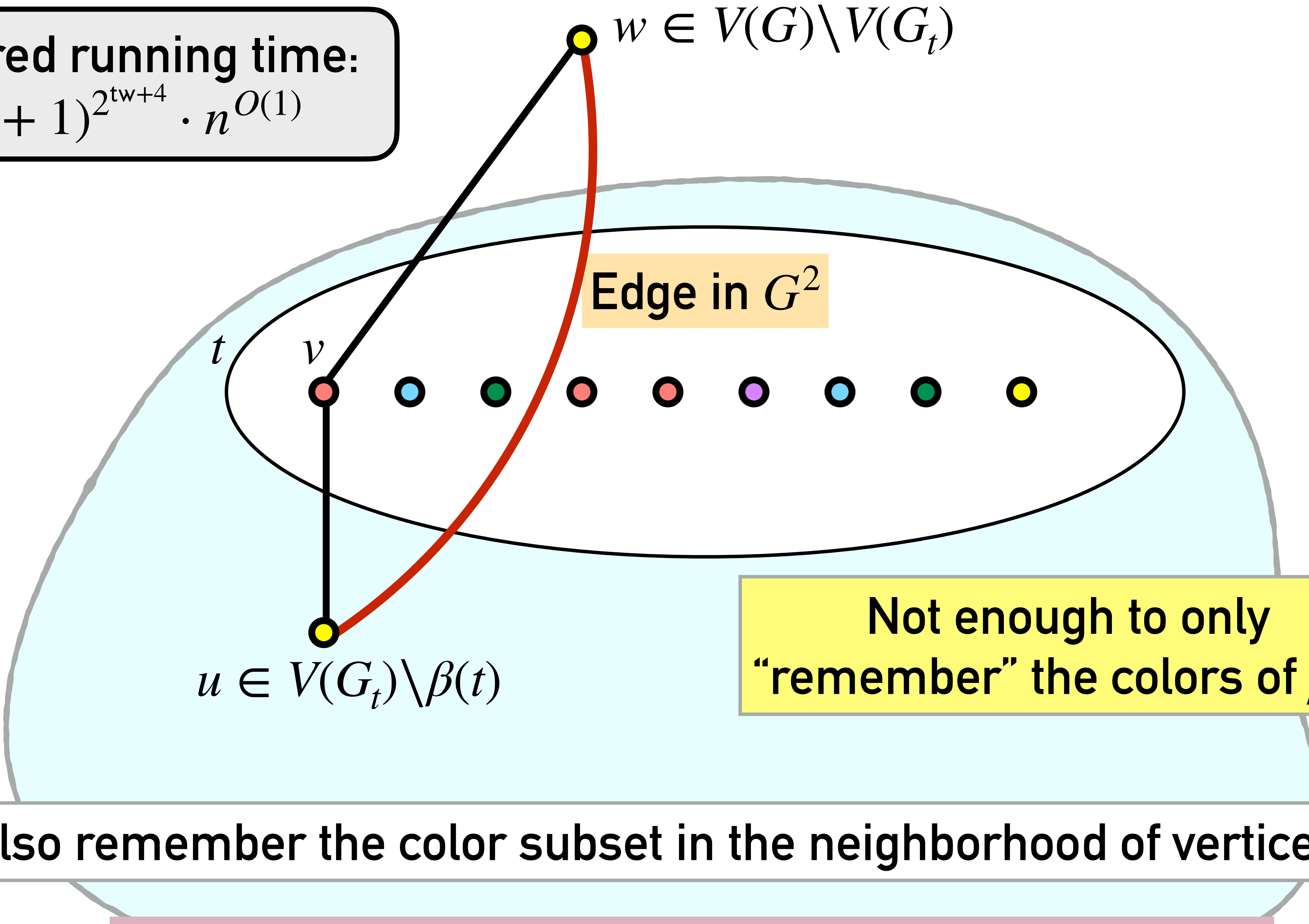


Not enough to only "remember" the colors of $\beta(t)$

- Also remember the color subset in the neighborhood of vertices in $\beta(t)$

Failure Again!

Desired running time:
$(q+1)^{2^{\mathrm{tw}+4}} \cdot n^{O(1)}$



Not enough to only "remember" the colors of $\beta(t)$

✱ Also remember the color subset in the neighborhood of vertices in $\beta(t)$

$(q \cdot 2^q)^{\mathrm{tw}+1}$ many states; too large for the desired running time!

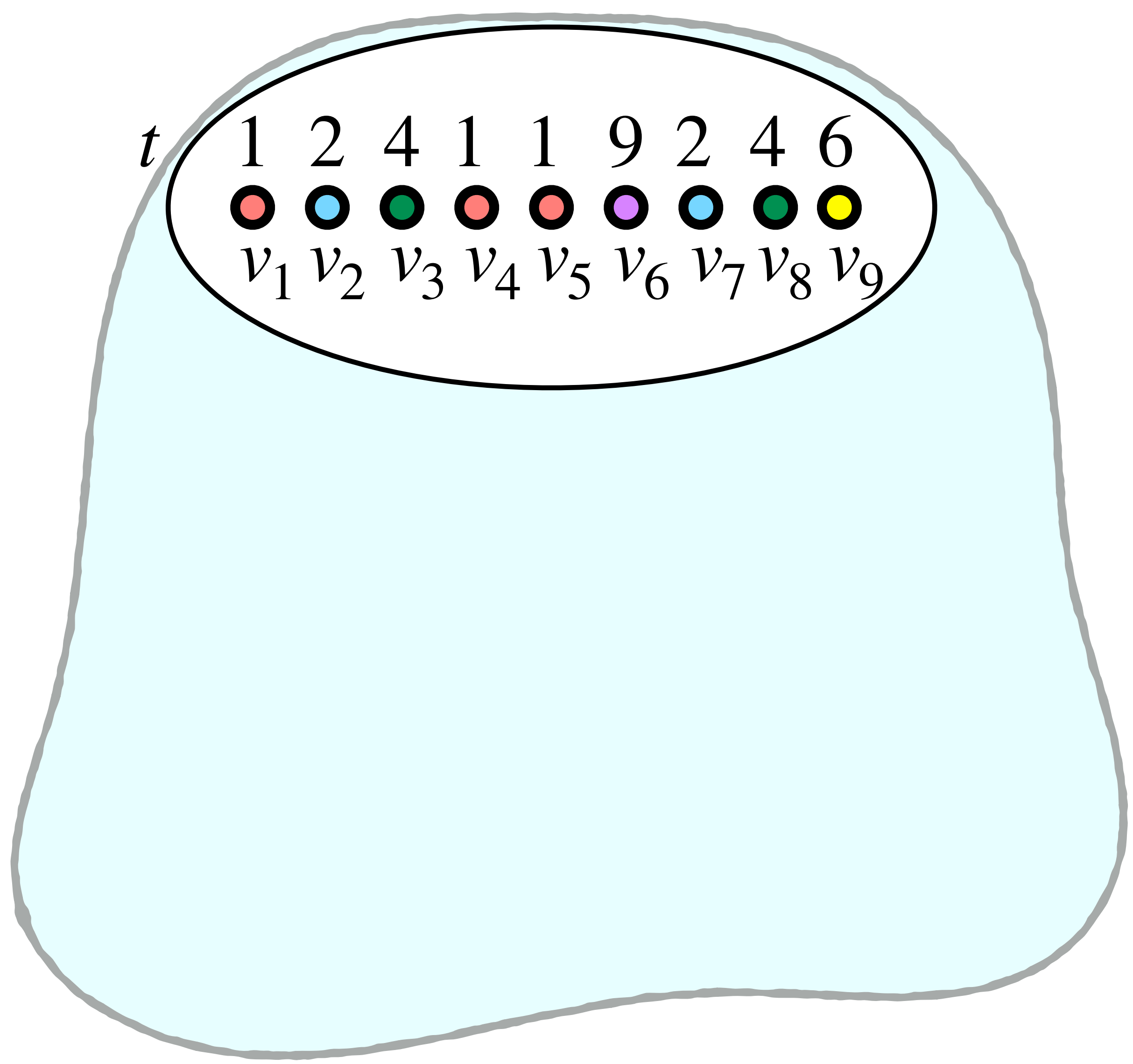
Key Insight

- Instead of remembering the color subsets in the neighborhood of vertices in $\beta(t)$, classify colors according to where they appear, and remember only the number of them!

Key Insight

* Instead of remembering the color subsets in the neighborhood of vertices in $\beta(t)$, classify colors according to where they appear, and remember only the number of them!

- Remember $\chi : \beta(t) \to \{1,2,\cdots,q\}$

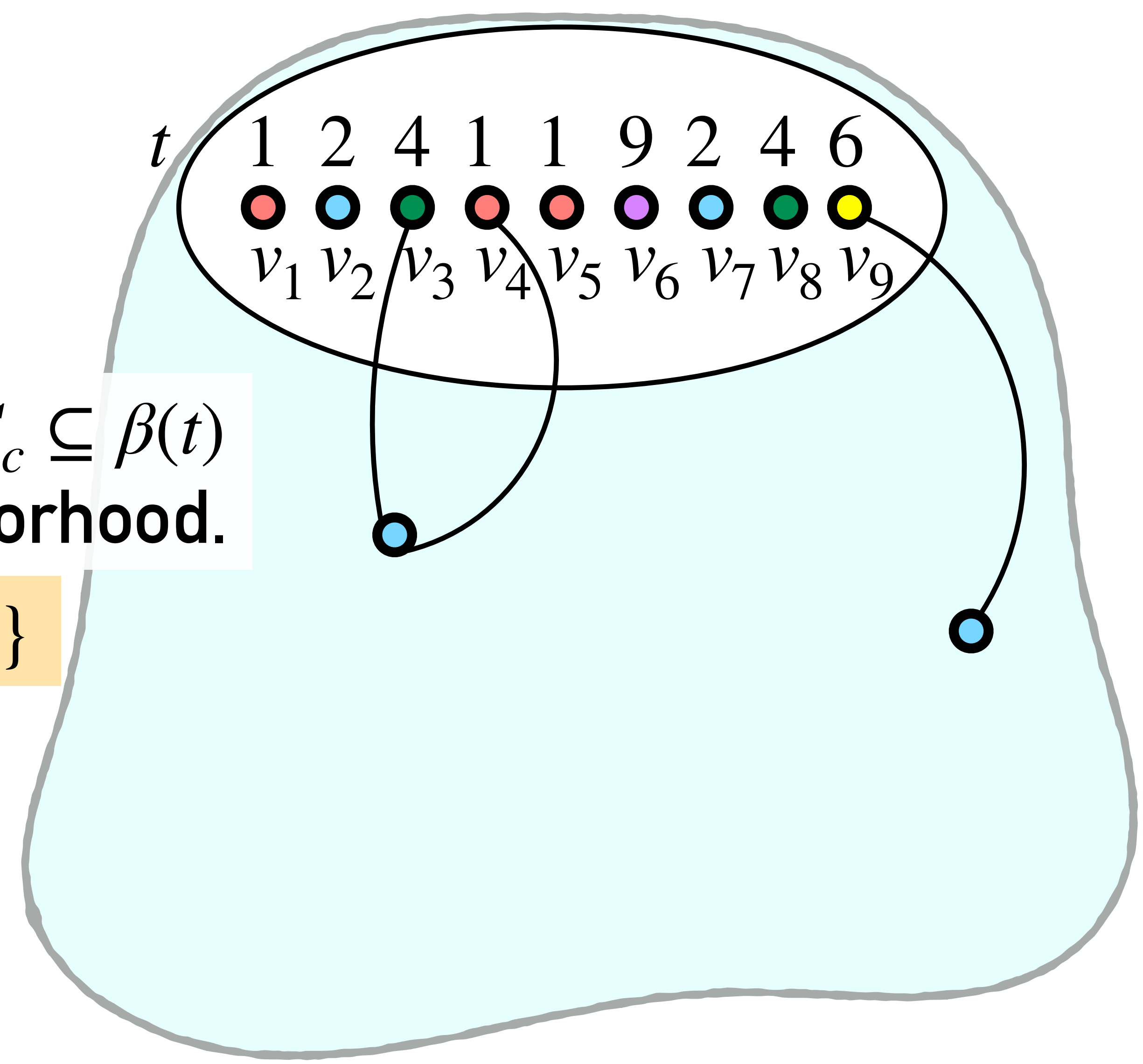


Key Insight

* Instead of remembering the color subsets in the neighborhood of vertices in $\beta(t)$, classify colors according to where they appear, and remember only the number of them!

- Remember $\chi : \beta(t) \to \{1,2,\cdots,q\}$
- For each color c used by χ, the vertices $S_c \subseteq \beta(t)$ that have a color c vertex in their neighborhood.

For color $c = 2$ (blue), $S_2 \subseteq \{v_3, v_4, v_9\}$

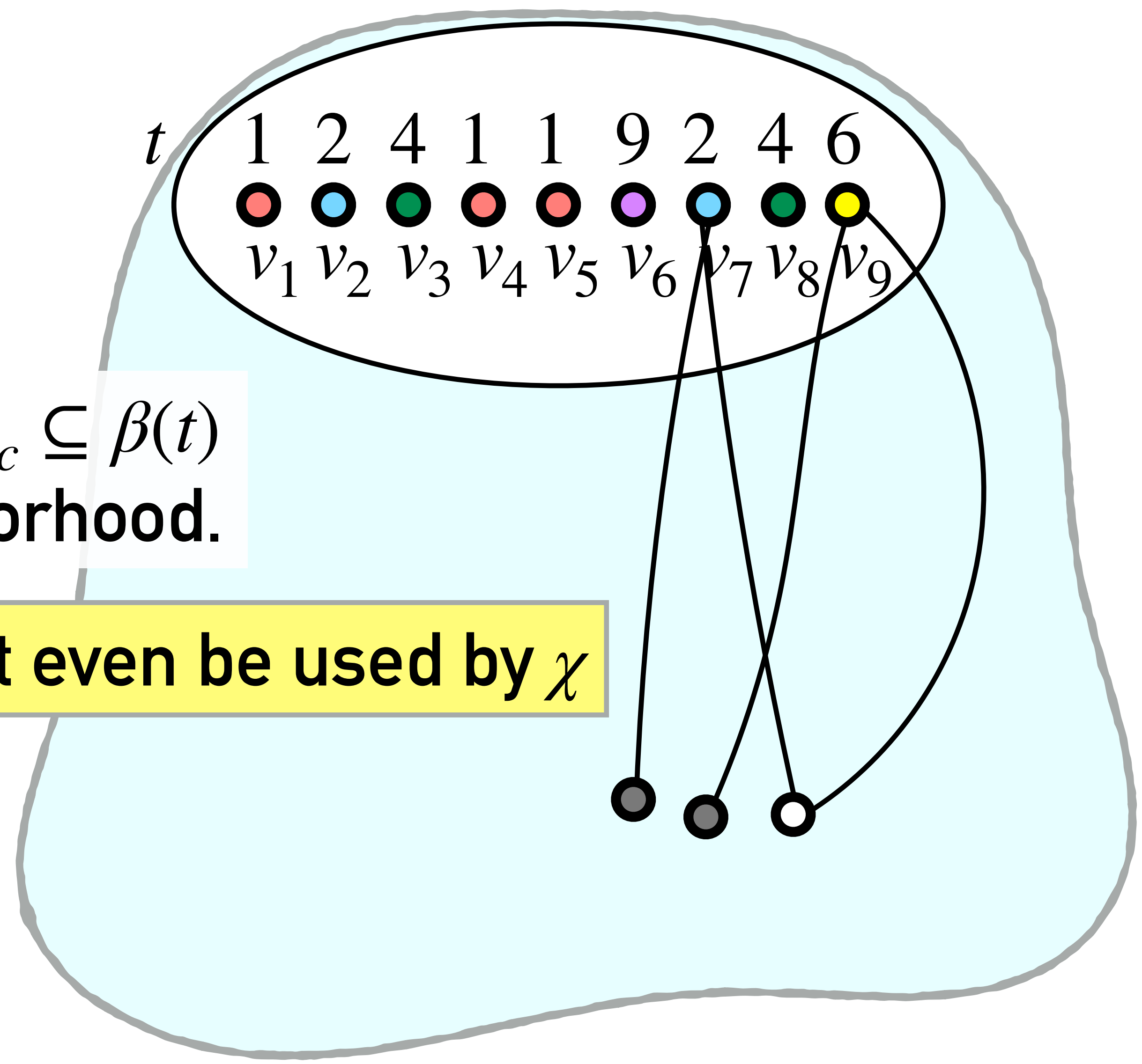


Key Insight

✱ Instead of remembering the color subsets in the neighborhood of vertices in $\beta(t)$, classify colors according to where they appear, and remember only the number of them!

- Remember $\chi : \beta(t) \to \{1,2,\cdots,q\}$

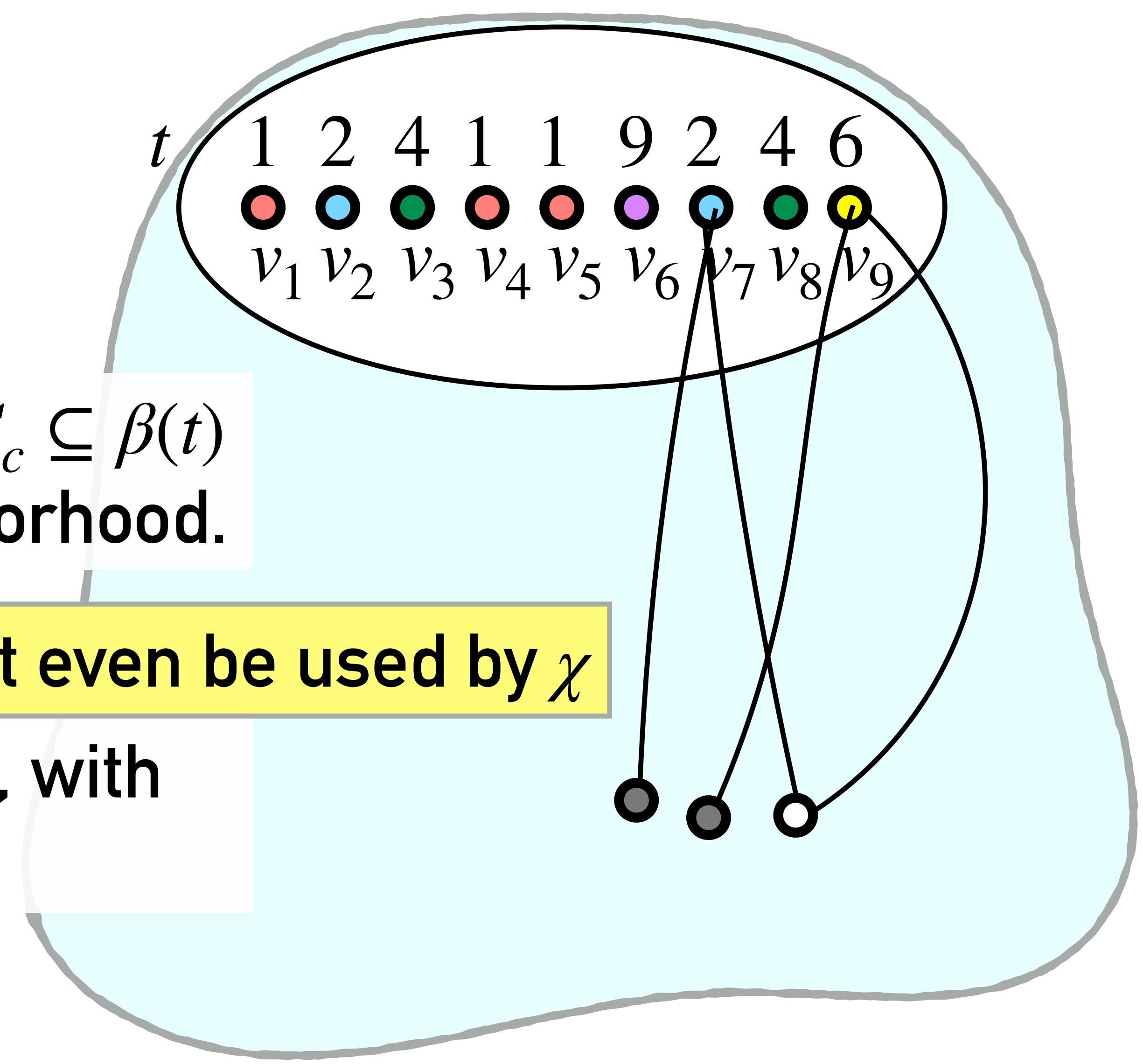
- For each color c used by χ, the vertices $S_c \subseteq \beta(t)$ that have a color c vertex in their neighborhood.



NOTE: Many colors from $\{1,2,\cdots,q\}$ may not even be used by χ

Key Insight

Instead of remembering the color subsets in the neighborhood of vertices in $\beta(t)$, classify colors according to where they appear, and remember only the number of them!



- Remember $\chi : \beta(t) \rightarrow \{1,2,\cdots,q\}$
- For each color c used by χ, the vertices $S_c \subseteq \beta(t)$ that have a color c vertex in their neighborhood.

NOTE: Many colors from $\{1,2,\cdots,q\}$ may not even be used by χ

- The number of colors, q_A, NOT used by χ, with neighborhood exactly $A \subseteq \beta(t)$.

Key Insight

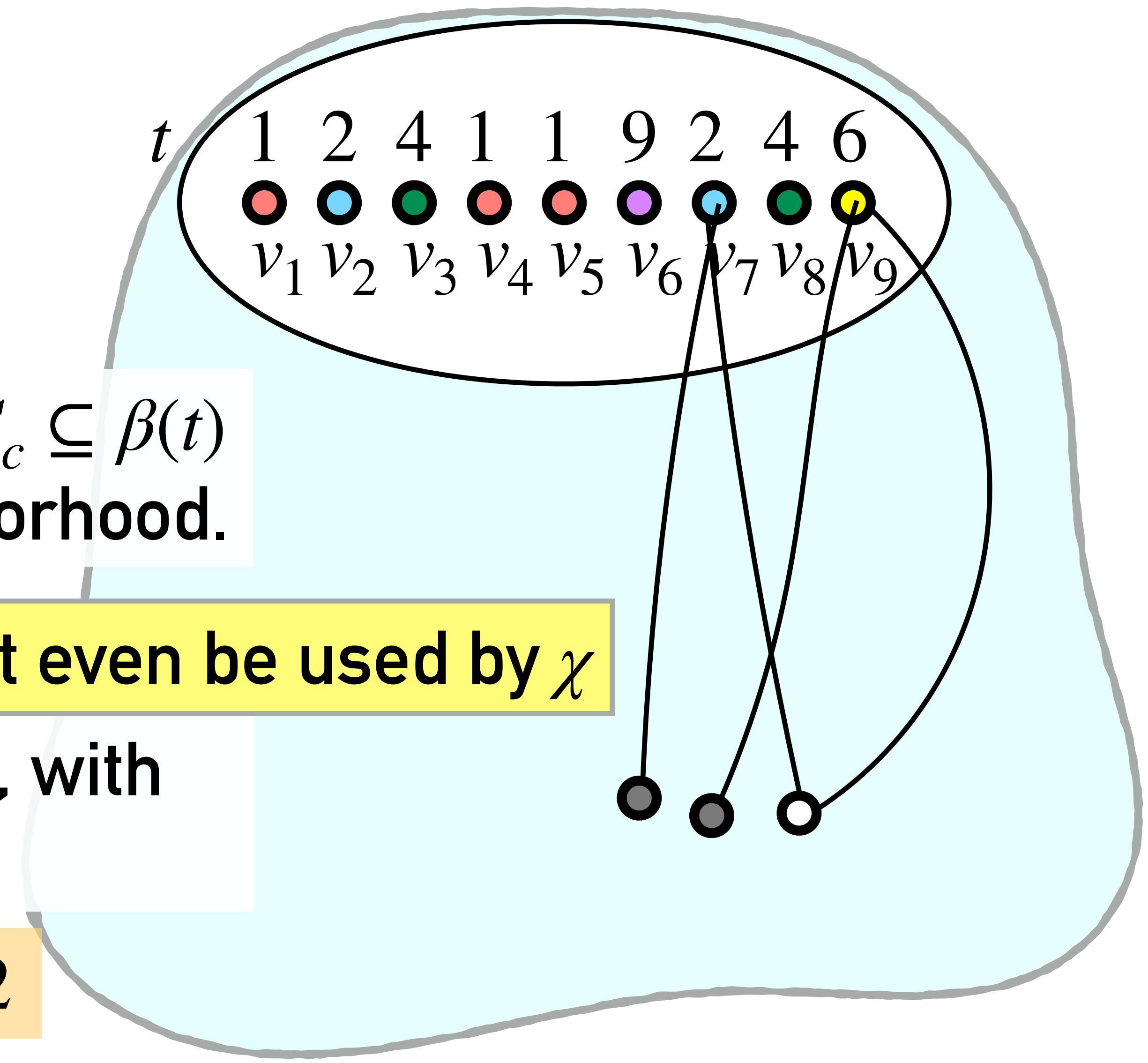
✱ Instead of remembering the color subsets in the neighborhood of vertices in $\beta(t)$, classify colors according to where they appear, and remember only the number of them!

- Remember $\chi : \beta(t) \to \{1,2,\cdots,q\}$
- For each color c used by χ, the vertices $S_c \subseteq \beta(t)$ that have a color c vertex in their neighborhood.

NOTE: Many colors from $\{1,2,\cdots,q\}$ may not even be used by χ

- The number of colors, q_A, NOT used by χ, with neighborhood exactly $A \subseteq \beta(t)$.

For $A = \{v_7, v_9\}$, $q_A = 2$



Key Insight

* Instead of remembering the color subsets in the neighborhood of vertices in $\beta(t)$, classify colors according to where they appear, and remember only the number of them!

- Remember $\chi : \beta(t) \to \{1,2,\cdots,q\}$

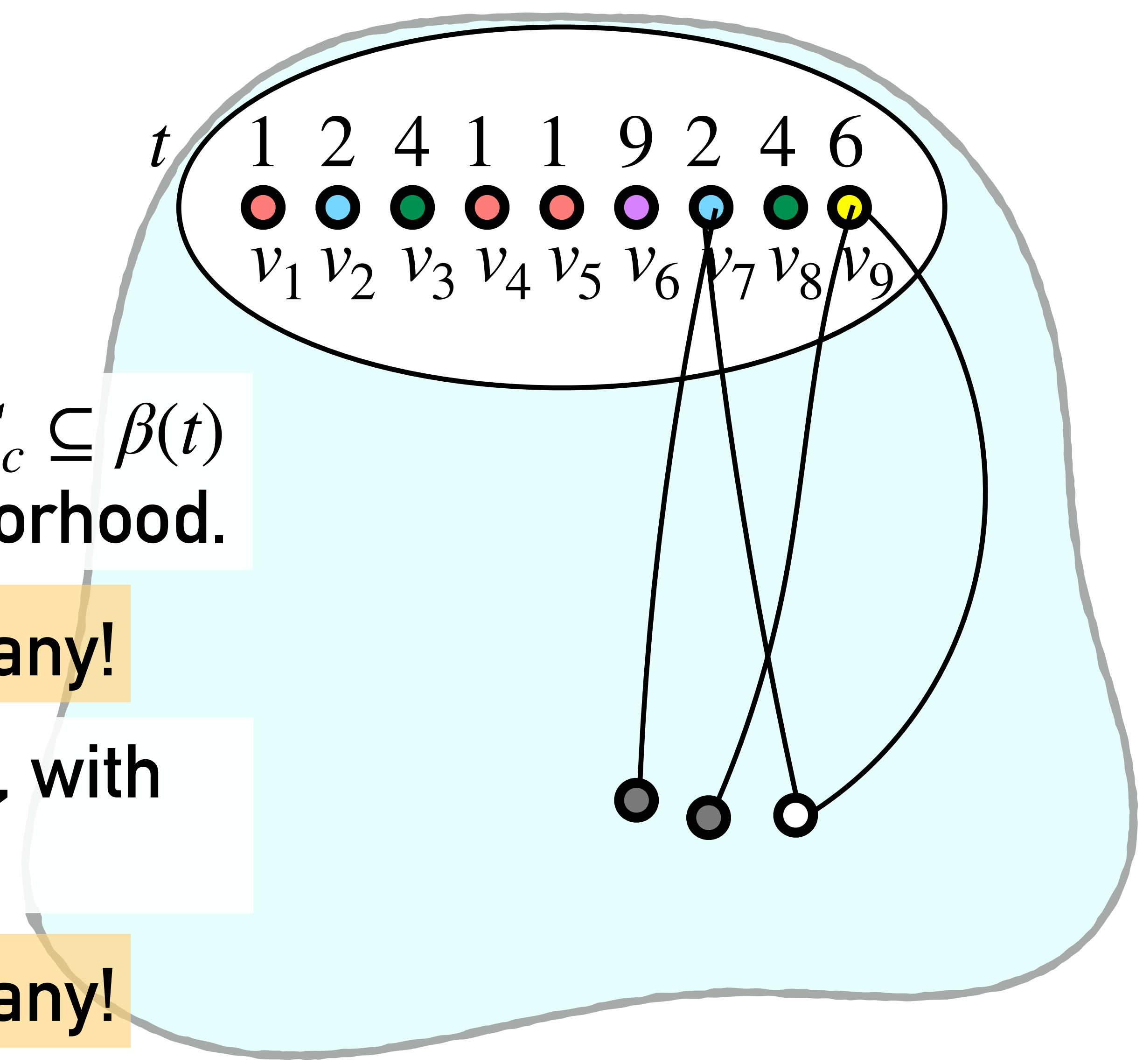
  At most $q^{\mathrm{tw}+1}$ many!

- For each color c used by χ, the vertices $S_c \subseteq \beta(t)$ that have a color c vertex in their neighborhood.

  At most $(2^{\mathrm{tw}+1})^{\mathrm{tw}+1}$ many!

- The number of colors, q_A, NOT used by χ, with neighborhood exactly $A \subseteq \beta(t)$.

  At most $(q+1)^{2^{\mathrm{tw}+1}}$ many!



Key Insight

No. of states are bounded by: $(q+1)^{2^{\mathrm{tw}+4}}$

- Remember $\chi : \beta(t) \to \{1,2,\cdots,q\}$

At most $q^{\mathrm{tw}+1}$ many!

- For each color c used by χ, the vertices $S_c \subseteq \beta(t)$ that have a color c vertex in their neighborhood.

At most $(2^{\mathrm{tw}+1})^{\mathrm{tw}+1}$ many!

- The number of colors, q_A, NOT used by χ, with neighborhood exactly $A \subseteq \beta(t)$.

At most $(q+1)^{2^{\mathrm{tw}+1}}$ many!

Key Insight

No. of states are bounded by: $(q+1)^{2^{\text{tw}+4}}$

But the trouble doesn't end!

- Remember $\chi : \beta(t) \to \{1,2,\cdots,q\}$

At most $q^{\text{tw}+1}$ many!

- For each color c used by χ, the vertices $S_c \subseteq \beta(t)$ that have a color c vertex in their neighborhood.

At most $(2^{\text{tw}+1})^{\text{tw}+1}$ many!

- The number of colors, q_A, NOT used by χ, with neighborhood exactly $A \subseteq \beta(t)$.

At most $(q+1)^{2^{\text{tw}+1}}$ many!

This makes computing solutions harder from the descendant:

FIX: Another layer of Integer Linear Programming based dynamic programming!

Planar Square Coloring

Planar Square Coloring has $2^{O(n^{2/3} \log n)}$-time algorithm.

Weird Looking Running Time?

Interplay between two algorithms

Planar Square Coloring

ALGORITHM

If $q \leq n^{1/3}$

ALGO 1:

Planar Square Coloring has an $q^{O(\sqrt{qn})}$-time algorithm

If $q \geq n^{1/3}$

ALGO 2:

Planar Square Coloring has an $2^{O(\frac{n \log n}{q})}$-time algorithm

$2^{O(n^{2/3} \log n)}$-time algorithm!

Planar Square Coloring

ALGO 1:

Planar Square Coloring has an $q^{O(\sqrt{qn})}$-time algorithm

Planar Square Coloring

ALGO 1:

Planar Square Coloring has an $q^{O(\sqrt{qn})}$-time algorithm

- We show that the treewidth of G^2 is bounded by $O(\sqrt{n\Delta})$

Δ = the maximum degree of a vertex in G

Planar Square Coloring

ALGO 1:

Planar Square Coloring has an $q^{O(\sqrt{qn})}$-time algorithm

- We show that the treewidth of G^2 is bounded by $O(\sqrt{n\Delta})$

Δ = the maximum degree of a vertex in G

RECALL:

- Coloring has a $q^{O(\mathrm{tw})} \cdot n^{O(1)}$-time algorithm
- $q \geq \Delta + 1$, for a yes-instance of Planar Square Coloring

Planar Square Coloring

ALGO 1:

Planar Square Coloring has an $q^{O(\sqrt{qn})}$-time algorithm

- We show that the treewidth of G^2 is bounded by $O(\sqrt{n\Delta})$

  Δ = the maximum degree of a vertex in G

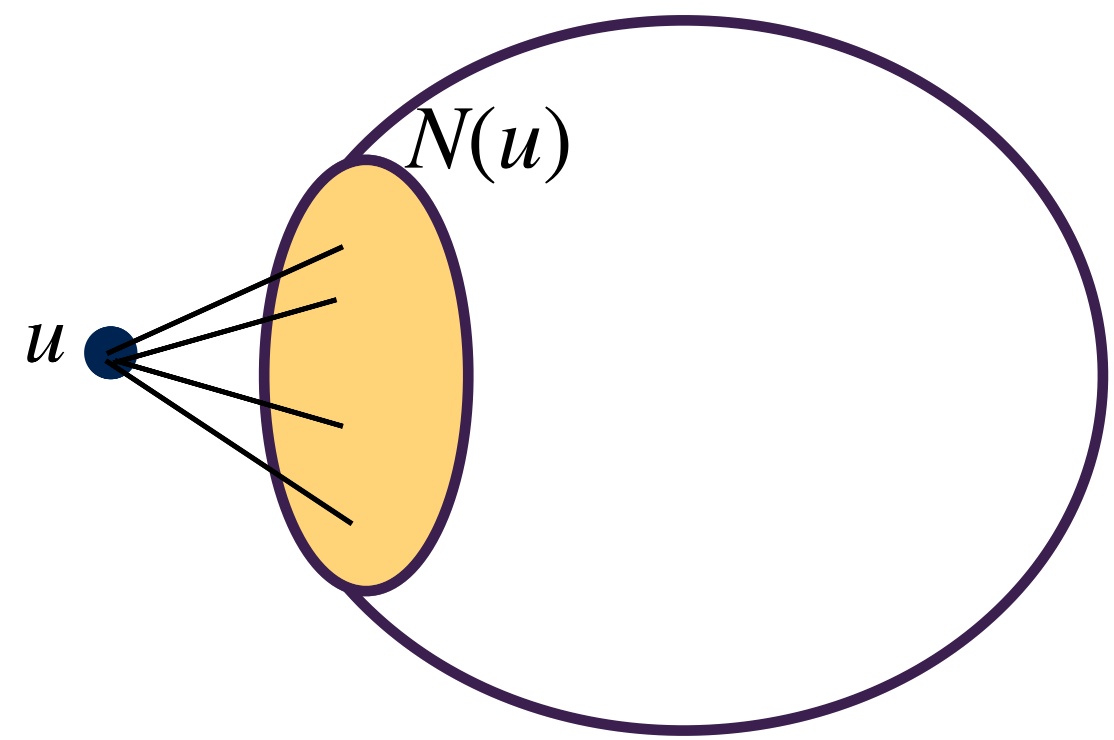
RECALL:

- Coloring has a $q^{O(\text{tw})} \cdot n^{O(1)}$-time algorithm
- $q \geq \Delta + 1$, for a yes-instance of Planar Square Coloring

Planar Square Coloring

ALGO 2:

Planar Square Coloring has an $2^{O(\frac{n \log n}{q})}$-time algorithm



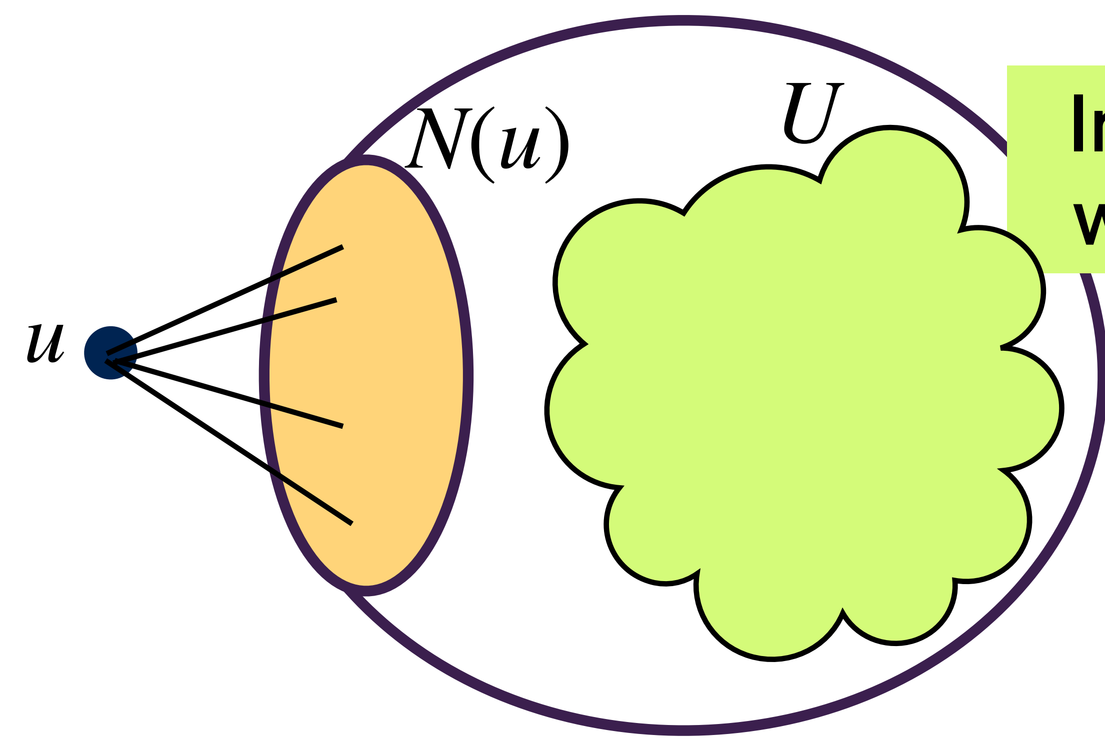
Graph G

If $|N(u)| \leq q-1$, sufficiently many colors available to color u, if the other vertices are already colored.

Planar Square Coloring

ALGO 2:

Planar Square Coloring has an $2^{O(\frac{n \log n}{q})}$-time algorithm



Important vertices: those with at least q neighbors

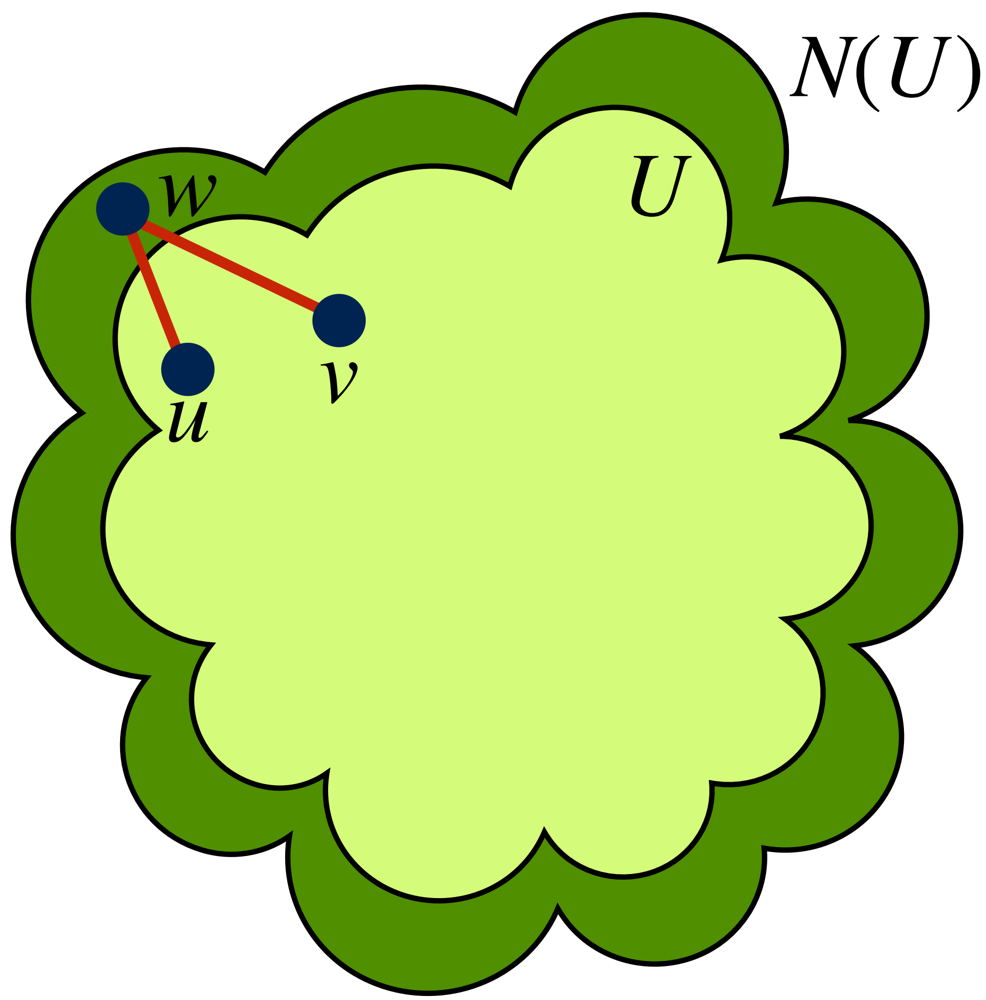
Graph G

If $|N(u)| \leq q-1$, sufficiently many colors available to color u, if the other vertices are already colored.

Planar Square Coloring

ALGO 2:

Planar Square Coloring has an $2^{O(\frac{n \log n}{q})}$-time algorithm

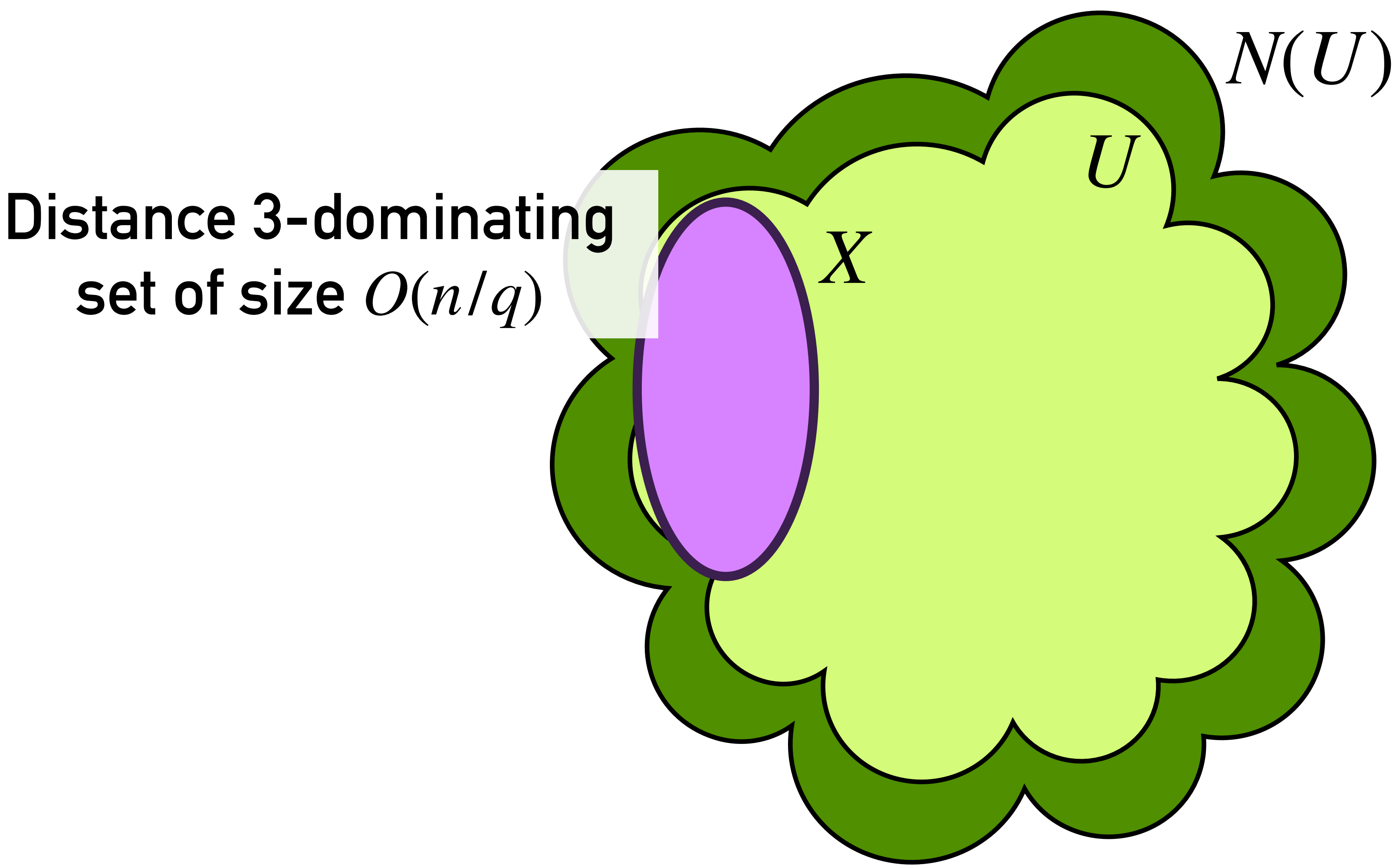


Graph $G[N[U]]$

Planar Square Coloring

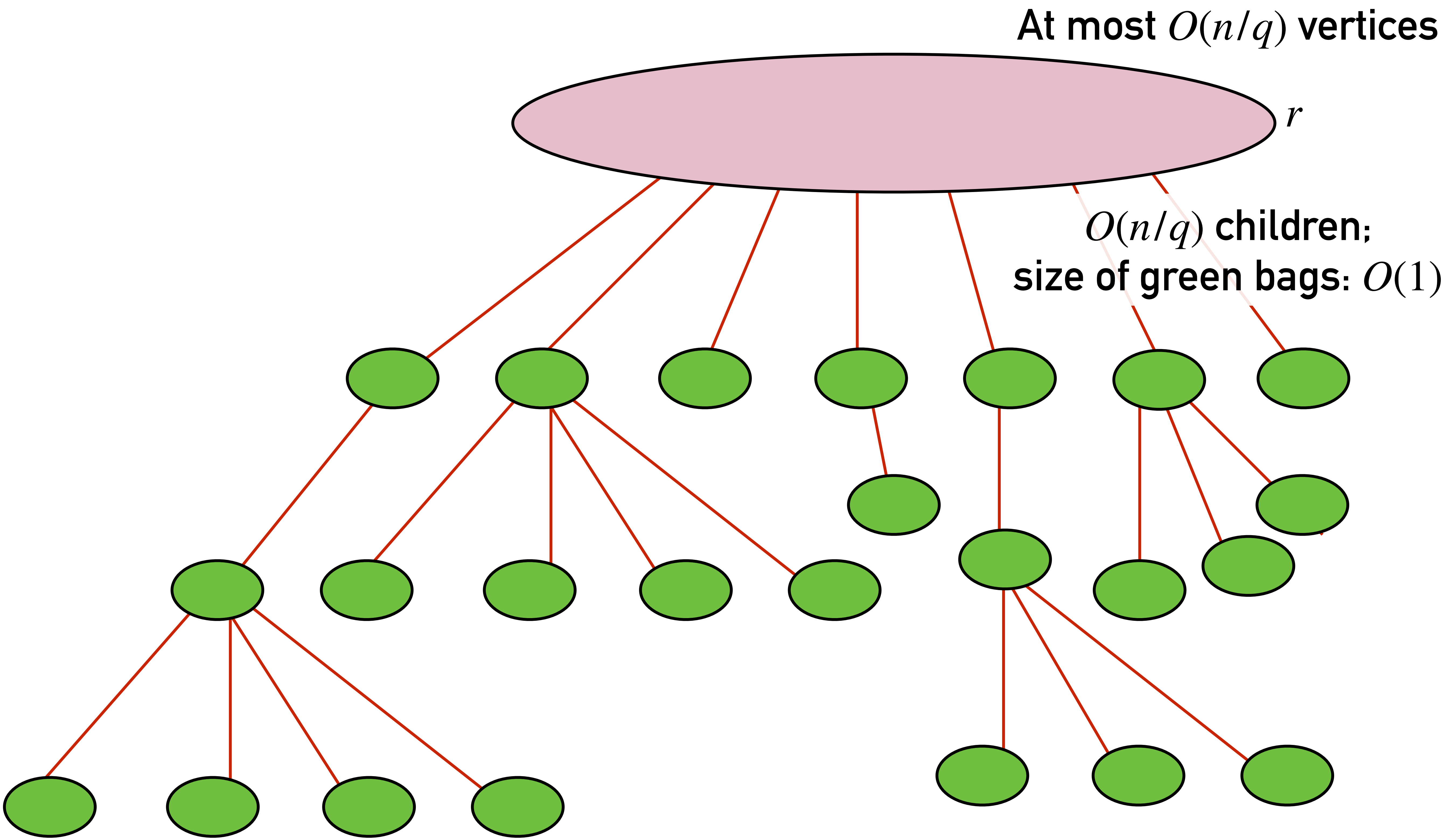
ALGO 2:

Planar Square Coloring has an $2^{O(\frac{n \log n}{q})}$-time algorithm



Graph $G[N[U]]$

Planar Square Coloring



Special tree decomposition for $G[N[U]]$

Key Insight

No. of states are bounded by: $(q+1)^{2^{\mathrm{tw}+4}} \cdot n^{O(1)}$

But the trouble doesn't end!

- Remember $\chi : \beta(t) \to \{1,2,\cdots,q\}$

  At most $q^{\mathrm{tw}+1}$ many!

- For each color c used by χ, the vertices $S_c \subseteq \beta(t)$ that have a color c vertex in their neighborhood.

  At most $(2^{\mathrm{tw}+1})^{\mathrm{tw}+1}$ many!

- The number of colors, q_A, NOT used by χ, with neighborhood exactly $A \subseteq \beta(t)$.

  At most $(q+1)^{2^{\mathrm{tw}+1}}$ many!

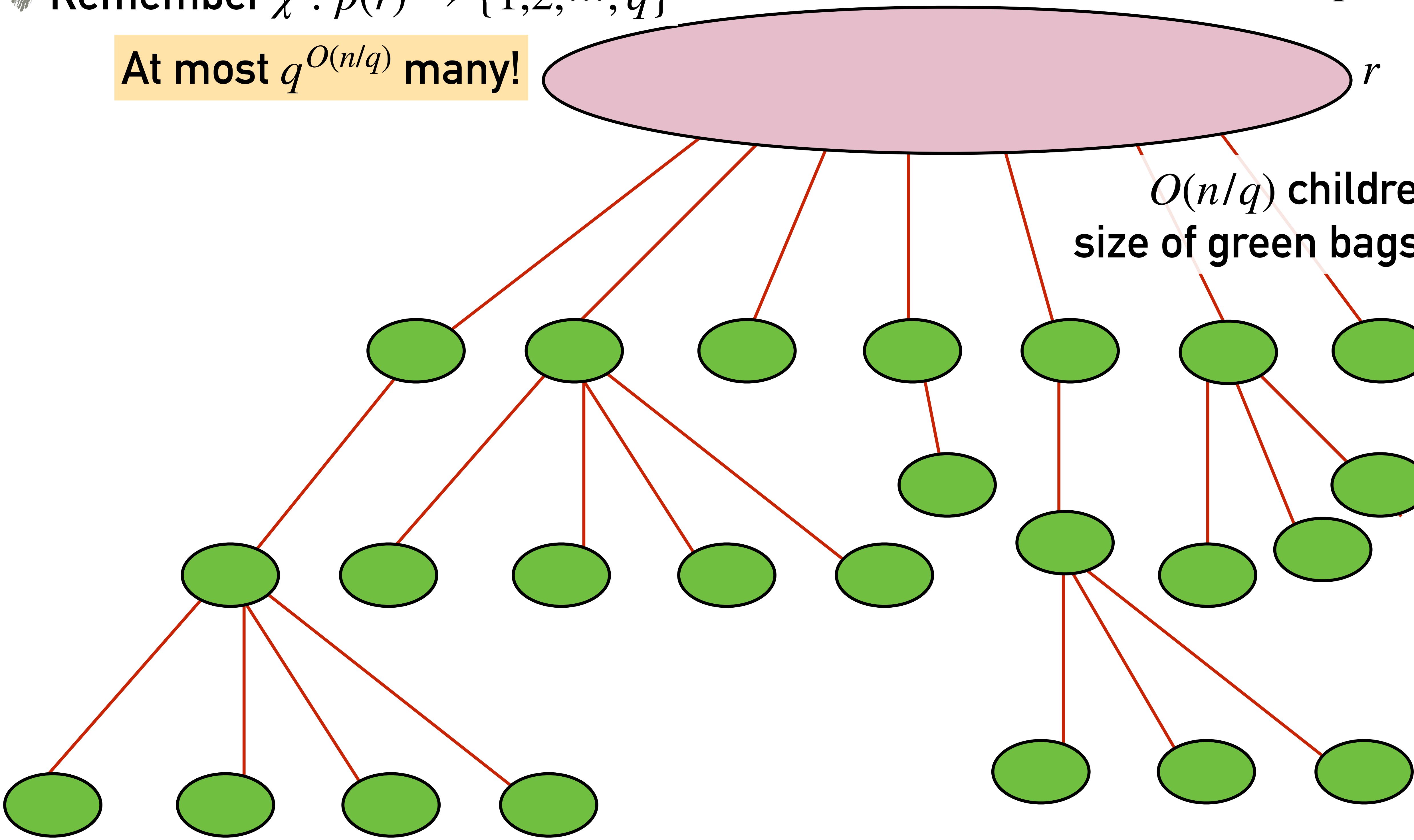
This makes computing solutions harder from the descendant:

FIX: Another layer of Integer Linear Programming based dynamic programming!

Planar Square Coloring

- Remember $\chi : \beta(r) \to \{1,2,\cdots,q\}$

At most $q^{O(n/q)}$ many!



Special tree decomposition for $G[N[U]]$

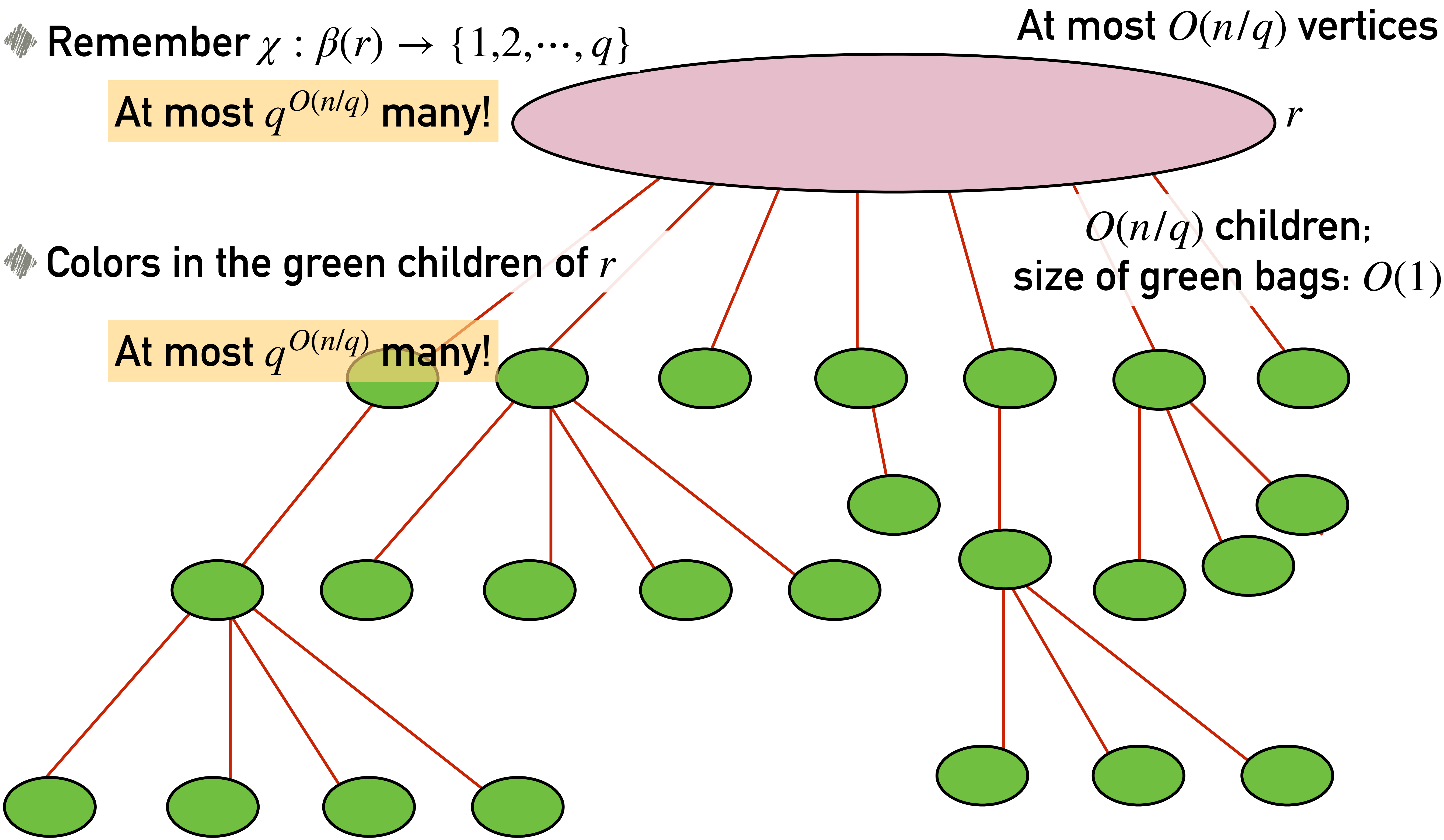
Planar Square Coloring

- Remember $\chi : \beta(r) \to \{1,2,\cdots,q\}$

  At most $q^{O(n/q)}$ many!

- Colors in the green children of r

  At most $q^{O(n/q)}$ many!



Special tree decomposition for $G[N[U]]$

Planar Square Coloring

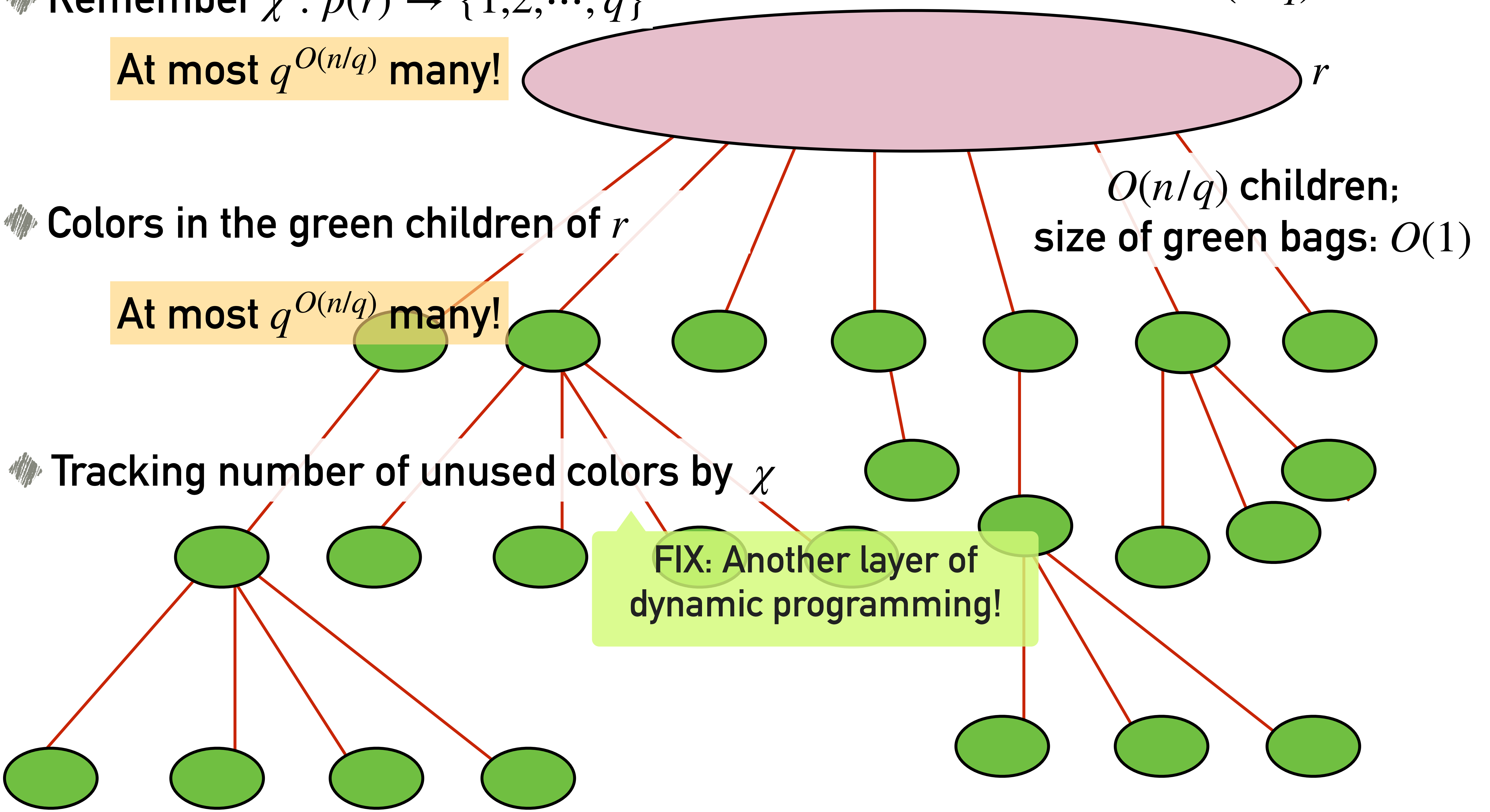
- Remember $\chi : \beta(r) \to \{1,2,\cdots,q\}$

  At most $q^{O(n/q)}$ many!

- Colors in the green children of r

  At most $q^{O(n/q)}$ many!

- Tracking number of unused colors by χ



Special tree decomposition for $G[N[U]]$

Planar Square Coloring

ALGO 2:

Planar Square Coloring has an $2^{O(\frac{n \log n}{q})}$-time algorithm

Conclusion & Open Problems

- We saw an ETH-tight algorithm for Square Coloring with an unusual running time.

Conclusion & Open Problems

- We saw an ETH-tight algorithm for Square Coloring with an unusual running time.

- For Planar Square Coloring we obtained a sub-exponential algorithm.

Conclusion & Open Problems

- We saw an ETH-tight algorithm for Square Coloring with an unusual running time.
- For Planar Square Coloring we obtained a sub-exponential algorithm.
- Can we obtain a sub-exponential time algorithm for Square Coloring on H-minor free graphs?

Conclusion & Open Problems

- We saw an ETH-tight algorithm for Square Coloring with an unusual running time.
- For Planar Square Coloring we obtained a sub-exponential algorithm.
- Can we obtain a sub-exponential time algorithm for Square Coloring on H-minor free graphs?
- How do the algorithmic complexity vary when we look at distance d-colorings?

Conclusion & Open Problems

* We saw an ETH-tight algorithm for Square Coloring with an unusual running time.
* For Planar Square Coloring we obtained a sub-exponential algorithm.
* Can we obtain a sub-exponential time algorithm for Square Coloring on H-minor free graphs?
* How do the algorithmic complexity vary when we look at distance d-colorings?

Thank You!