Dynamic Edge Coloring with Improved Approximation

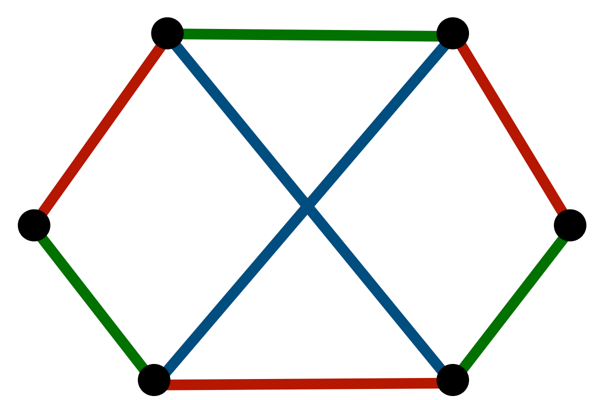
Ran Duan, Haoqing He, **Tianyi Zhang**

Tsinghua University

Definition: Edge Coloring

Undirected **simple** graph, max vertex degree $= \Delta$

- **Edge coloring**: any coloring of edges, s.t. any two edges incident on the same vertex have different colors



- **Number of colors**: NP-hard to decide if Δ-colorable, but $\Delta + 1$ coloring can be computed efficiently **[Viz'64]**

Definition: Dynamic Edge Coloring

Data structure

- Maintain an **edge coloring** using a "small" number of colors

Update operation

- **Input: insertion / deletion** of an edge
- **Output: reassignment** of colors

A short history

Assume is a fixed value

Reference	Number of colors	Update time
[Viz'64]	$\Delta + 1$	$\tilde{O}(n)$
[BM'17]	$O(\Delta)$	$\tilde{O}(\sqrt{\Delta})$
[BCHN'18]	$2\Delta - 1$	$O(\log \Delta)$
[CHLPU'18]	$\Delta + c$ $c \leq \Delta/3$	$\Omega(\frac{\Delta}{c} \log n)$
New	$(1+\epsilon)\Delta$ $\Delta \geq \Omega(\log^2 n/\epsilon^2)$	$O(\log^8 n/\epsilon^4)$ rand. & amortized

A short history

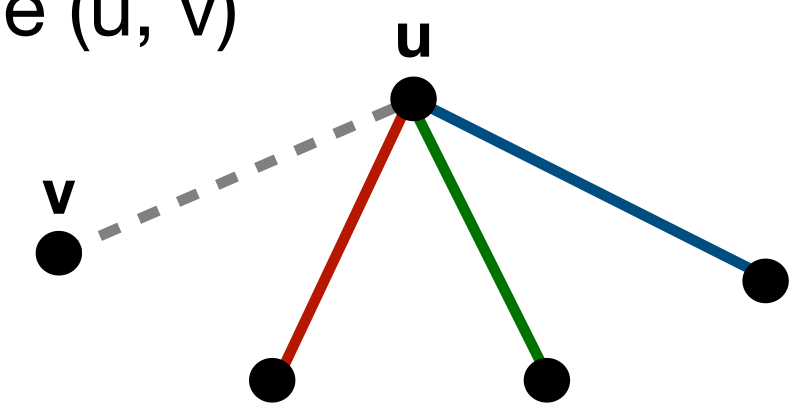
Assume is a fixed value

Reference	Number of colors	Update time
[Viz'64]	$\Delta + 1$	$\tilde{O}(n)$
[BM'17]	$O(\Delta)$	$\tilde{O}(\sqrt{\Delta})$
[BCHN'18]	$2\Delta - 1$	$O(\log \Delta)$
[CHLPU'18]	$\Delta + c$ $c \leq \Delta/3$	$\Omega(\frac{\Delta}{c} \log n)$
New	$(1+\epsilon)\Delta$ $\Delta \geq \Omega(\log^2 n/\epsilon^2)$	$O(\log^8 n/\epsilon^4)$ rand. & amortized

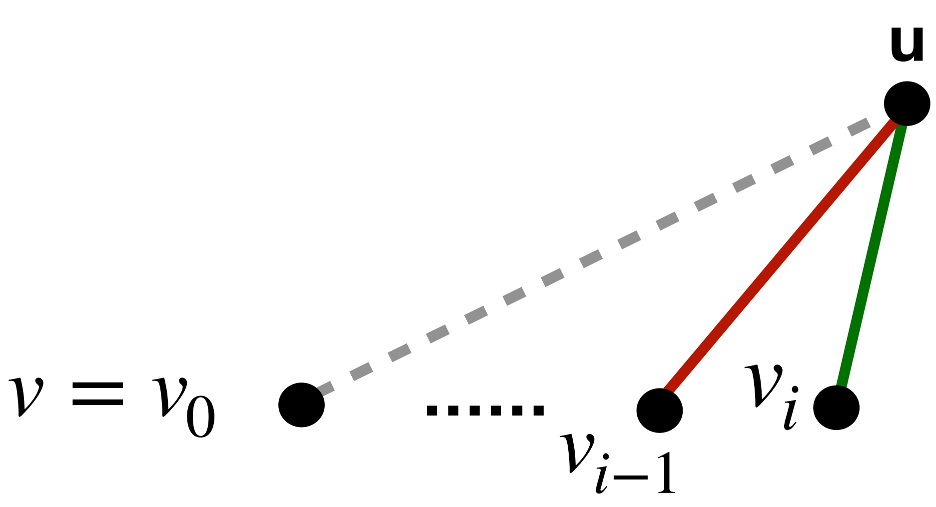
This is a **worst-case "lower bound"**

A review of [Viz'64]

A newly inserted edge (u, v)

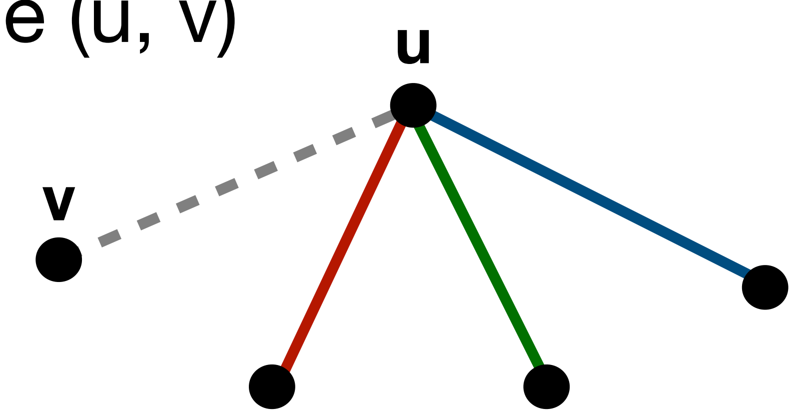


Idea: Find a maximal chain of neighbors

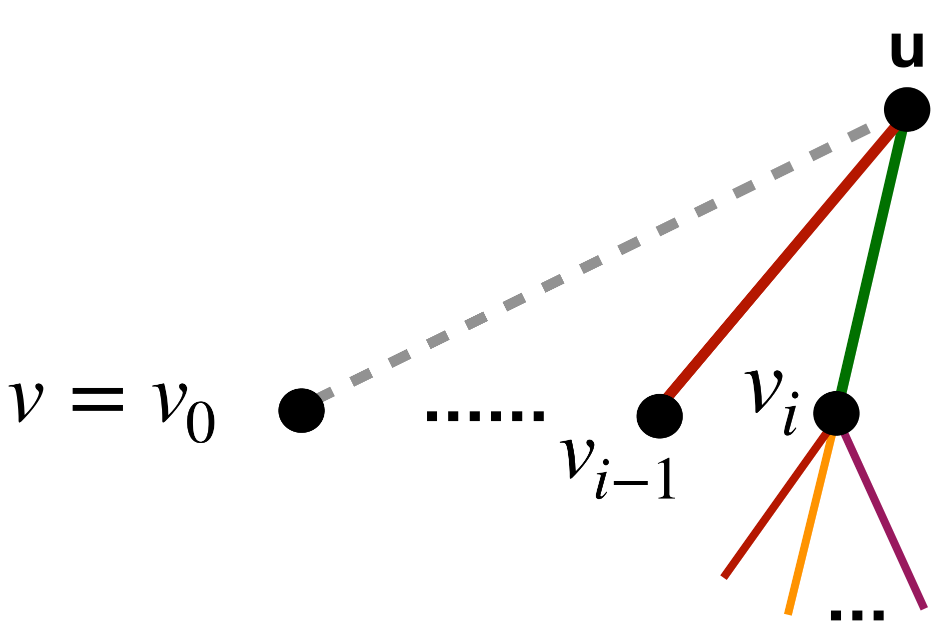


A review of [Viz'64]

A newly inserted edge (u, v)

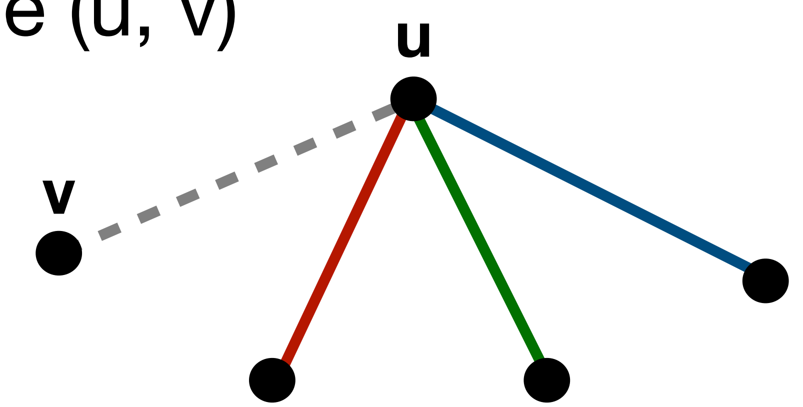


Idea: Find a maximal chain of neighbors

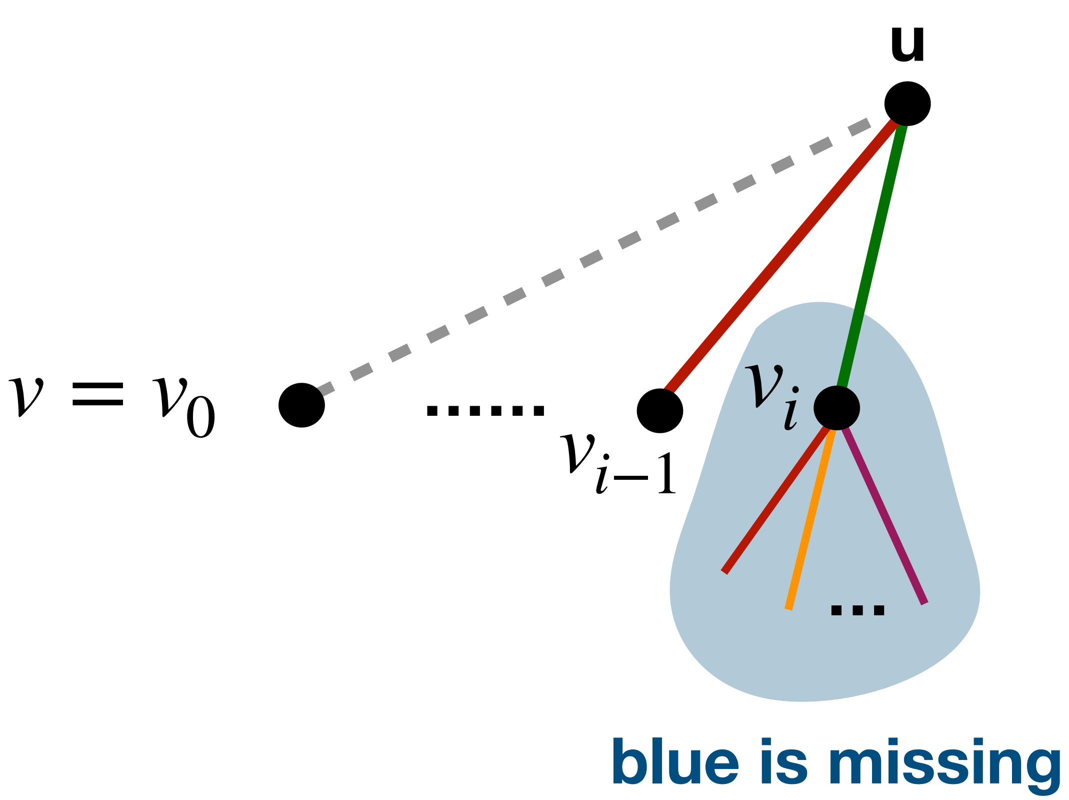


A review of [Viz'64]

A newly inserted edge (u, v)

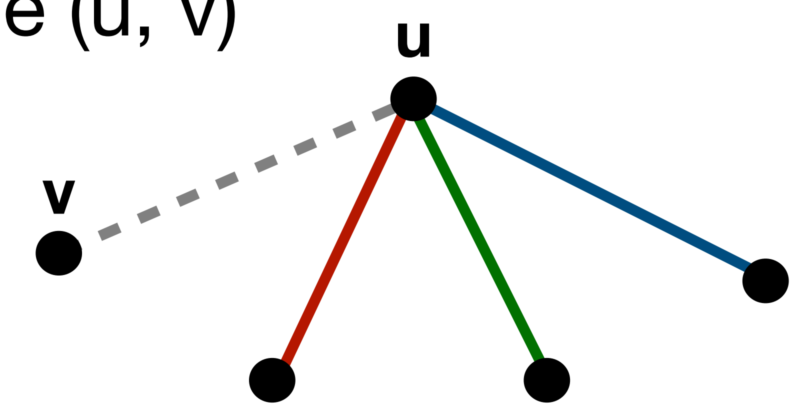


Idea: Find a maximal chain of neighbors

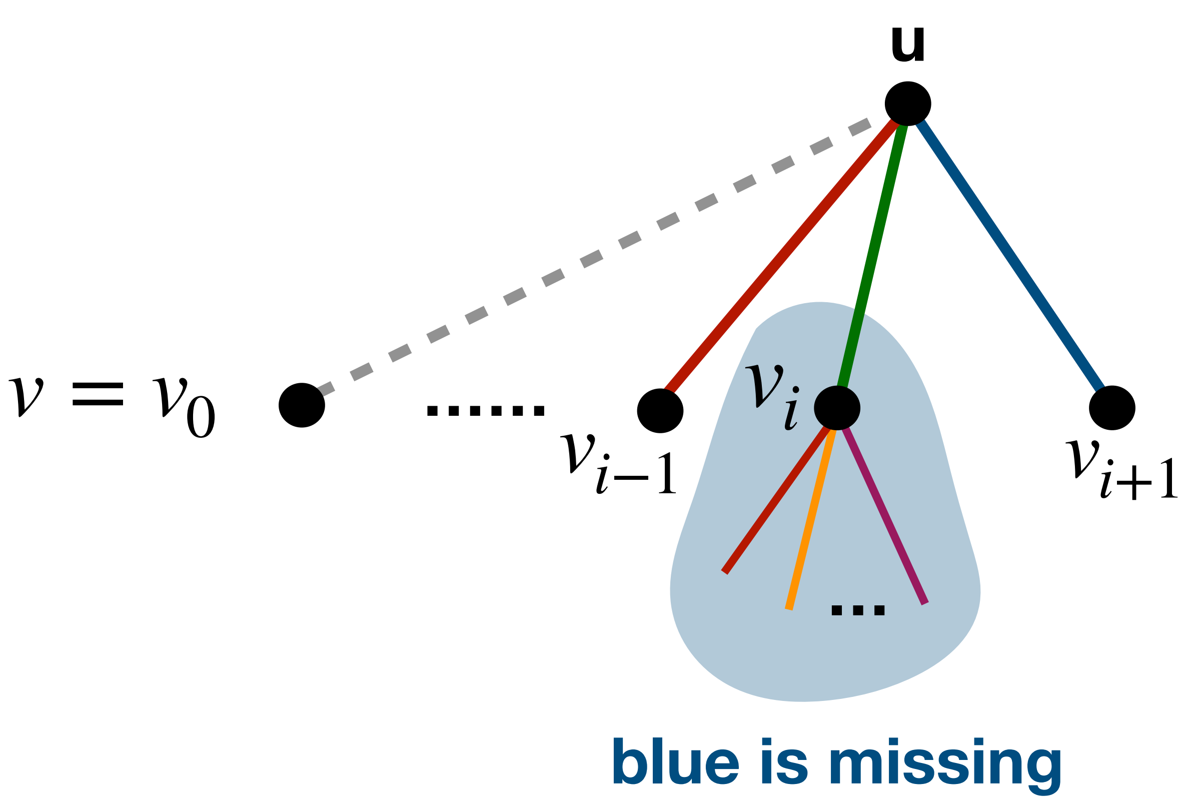


A review of [Viz'64]

A newly inserted edge (u, v)

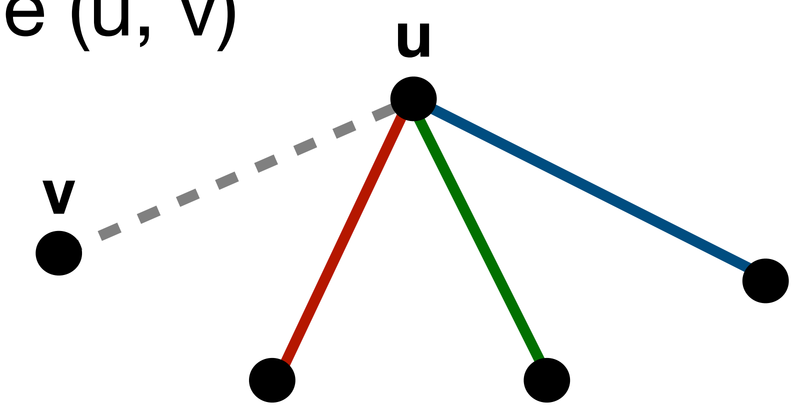


Idea: Find a maximal chain of neighbors

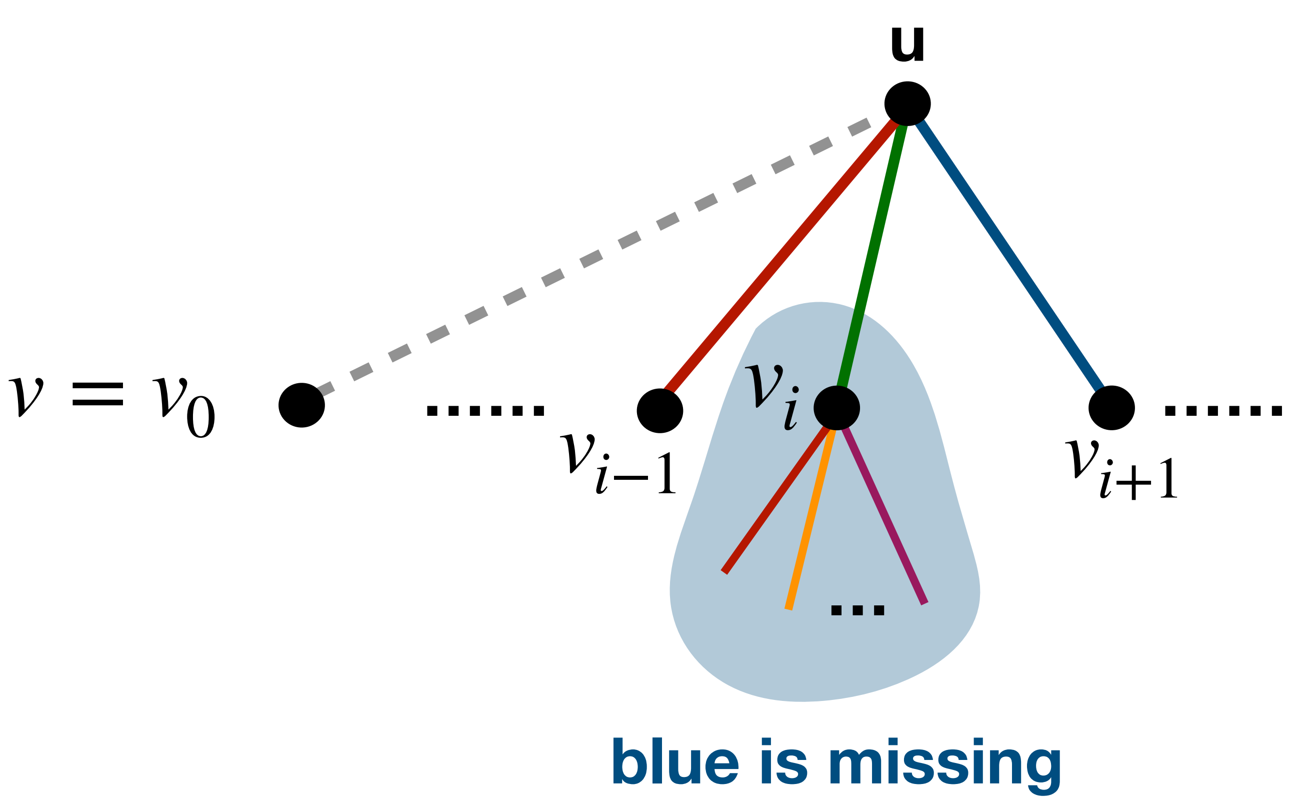


A review of [Viz'64]

A newly inserted edge (u, v)

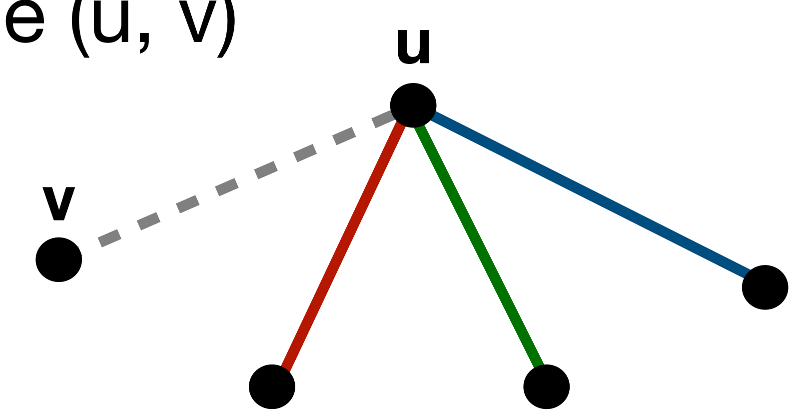


Idea: Find a maximal chain of neighbors

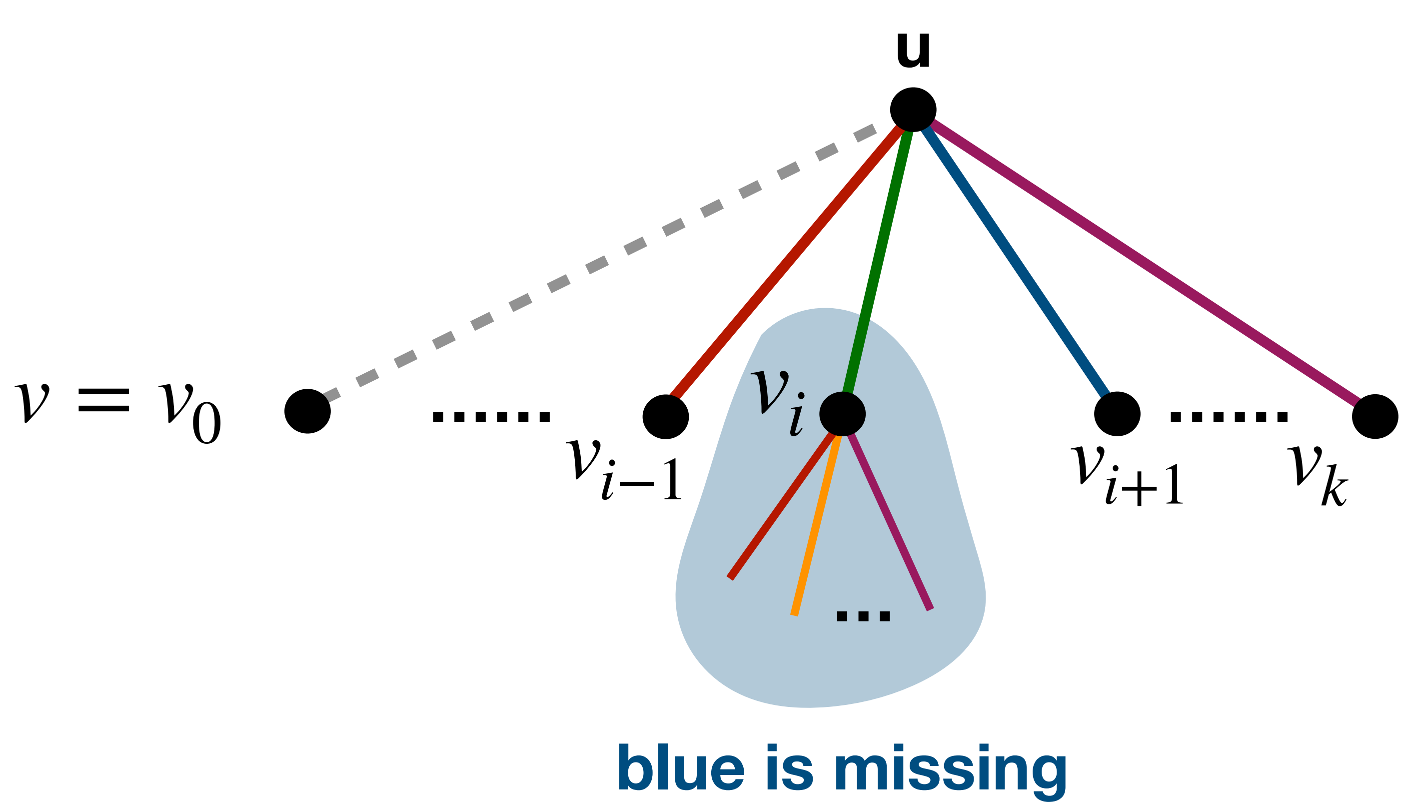


A review of [Viz'64]

A newly inserted edge (u, v)

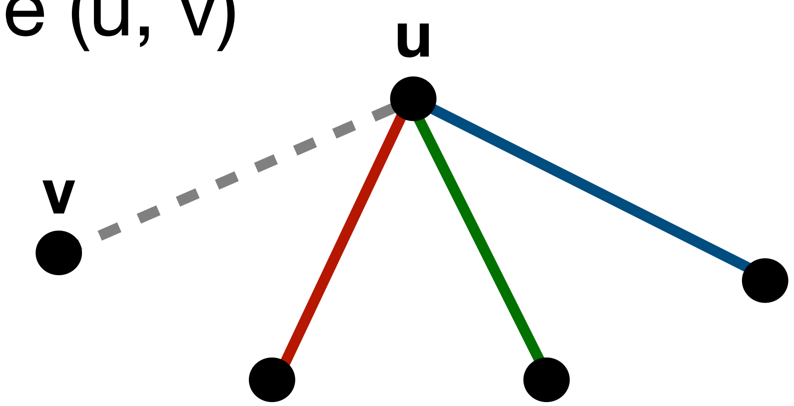


Idea: Find a maximal chain of neighbors

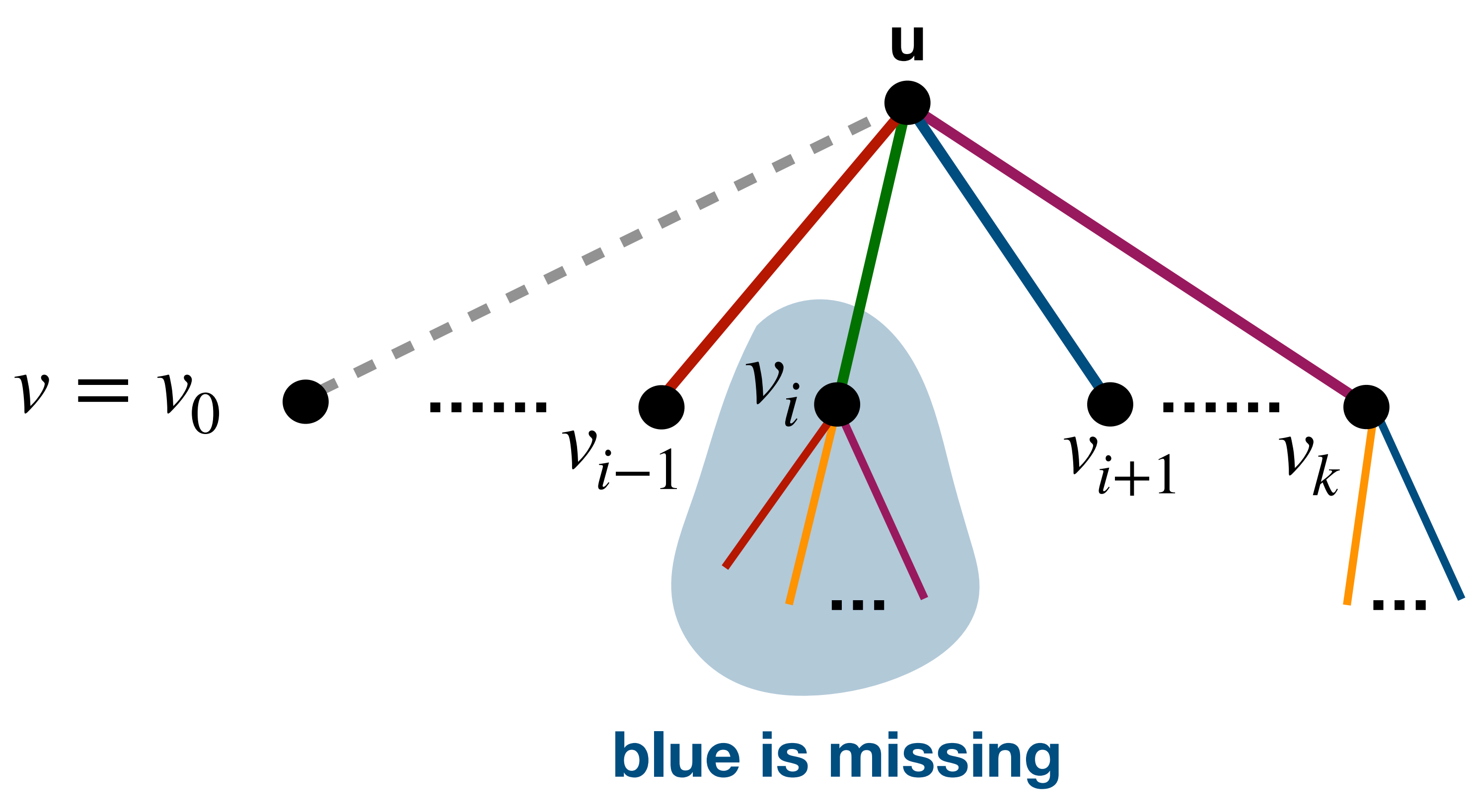


A review of [Viz'64]

A newly inserted edge (u, v)

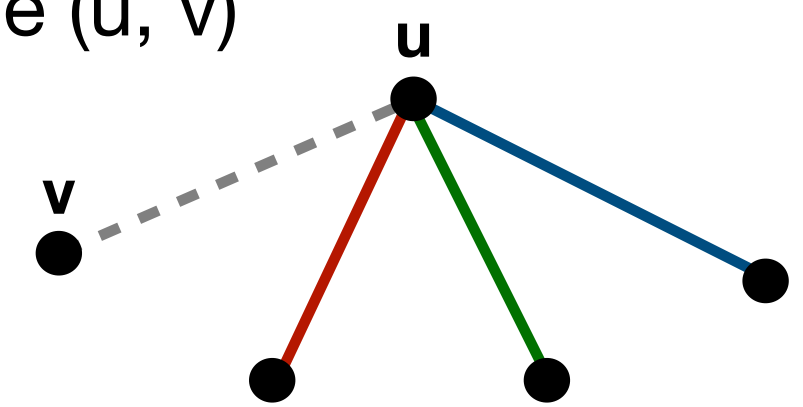


Idea: Find a maximal chain of neighbors

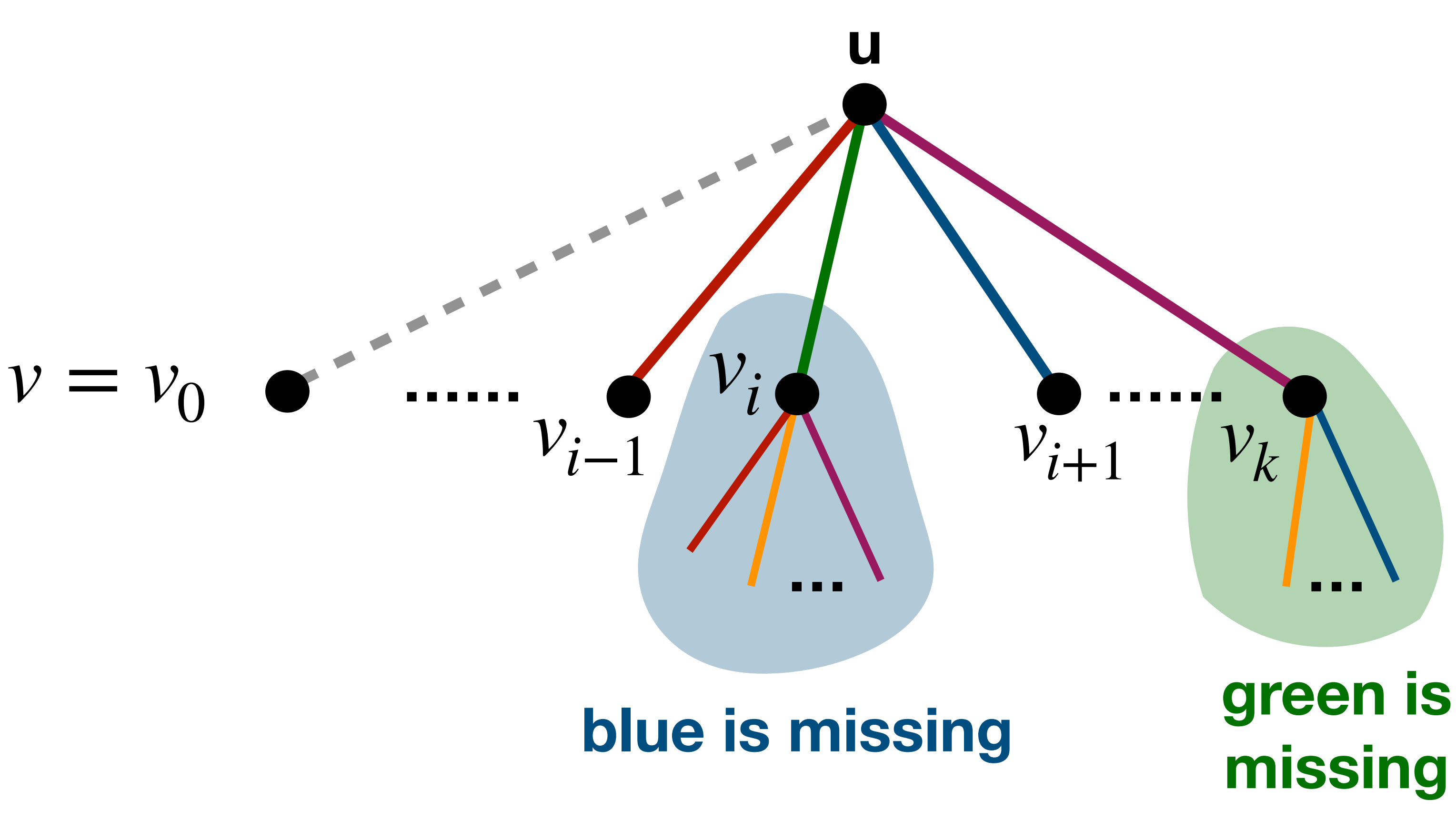


A review of [Viz'64]

A newly inserted edge (u, v)

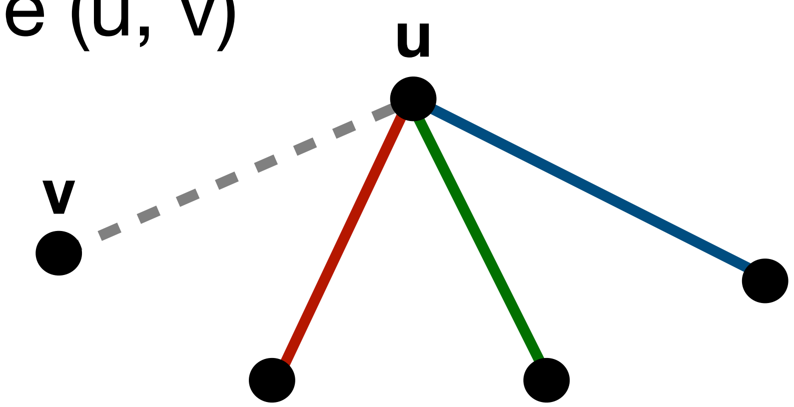


Idea: Find a maximal chain of neighbors

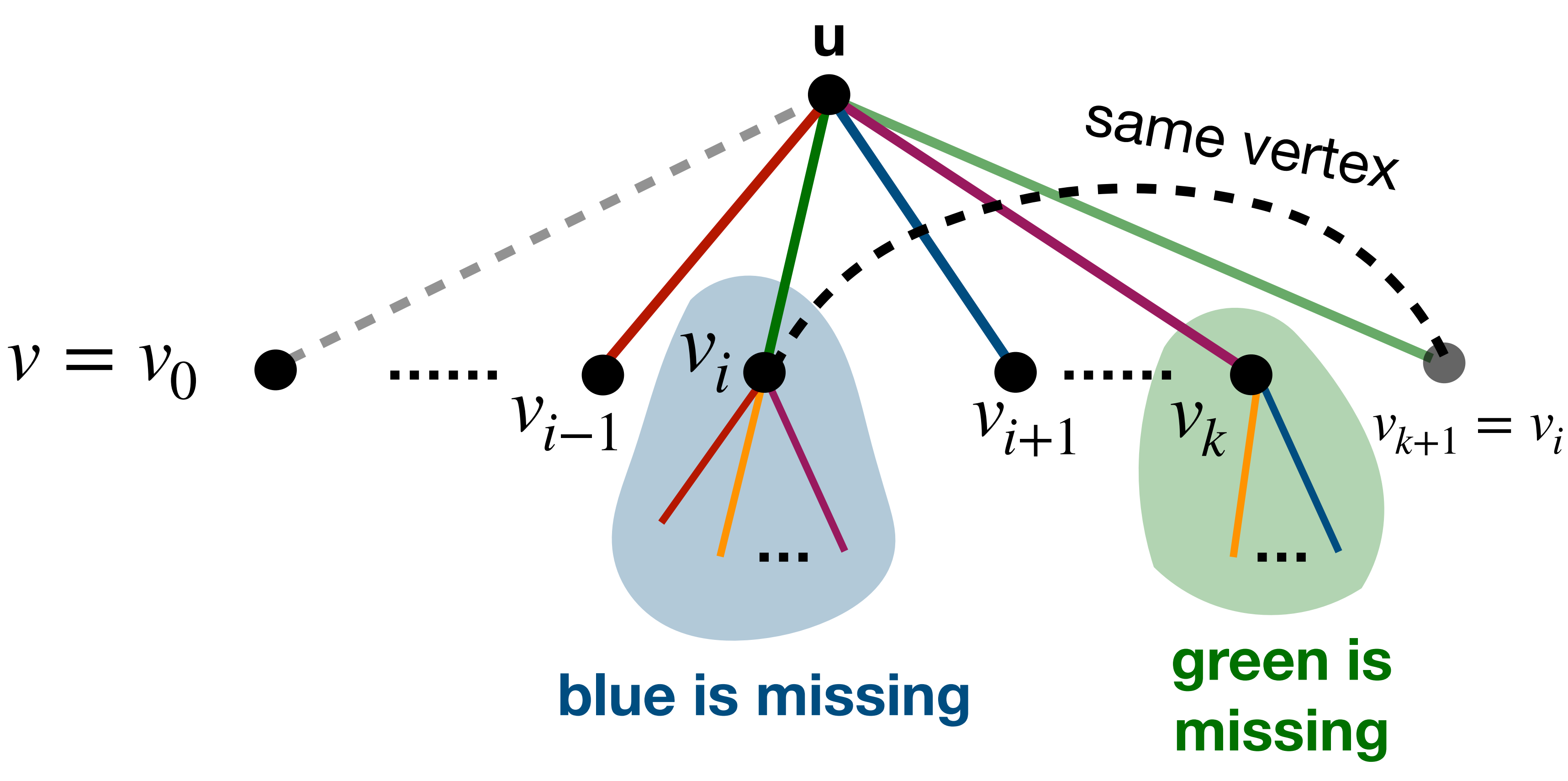


A review of [Viz'64]

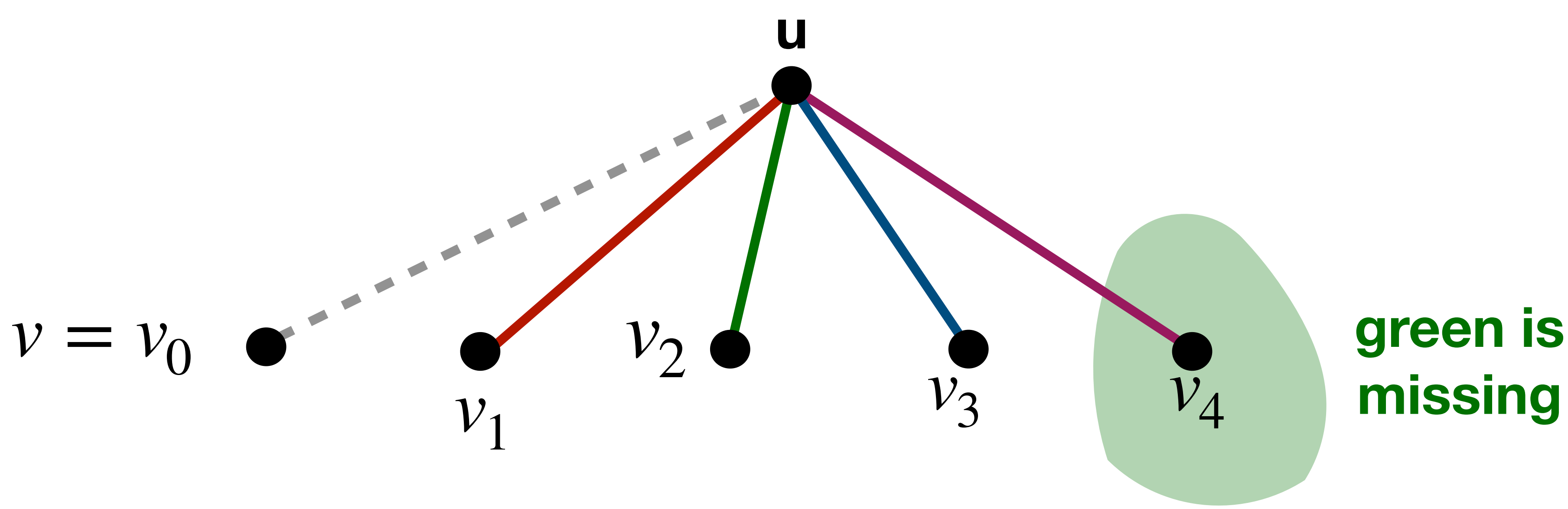
A newly inserted edge (u, v)



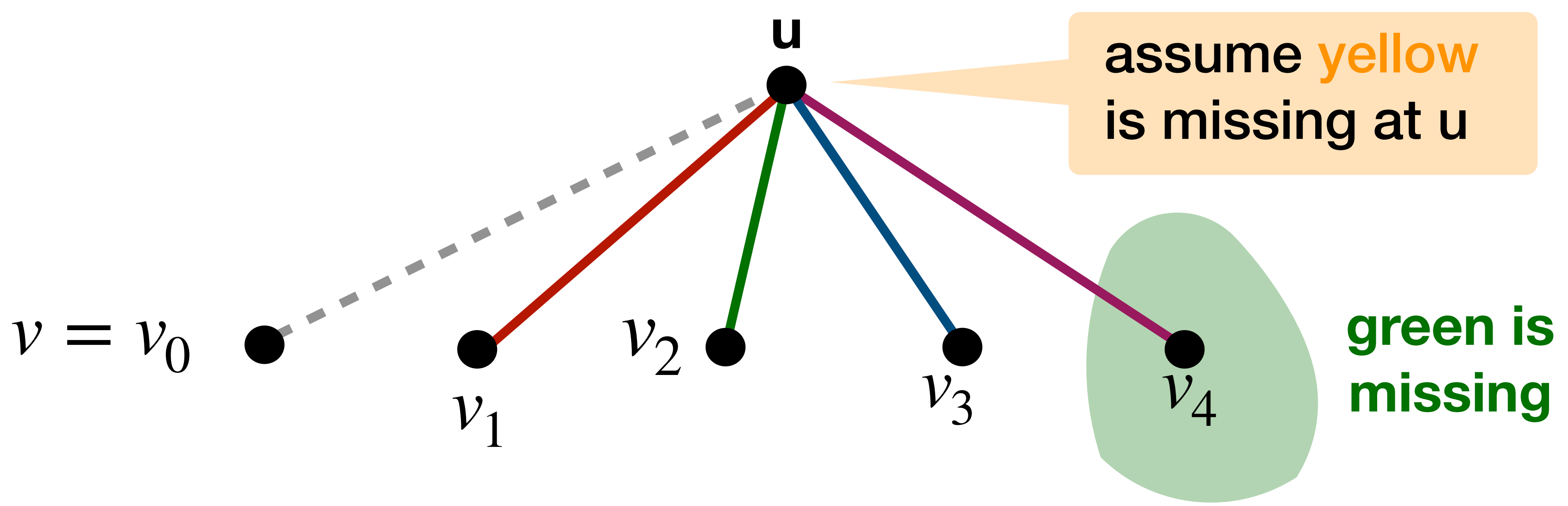
Idea: Find a maximal chain of neighbors



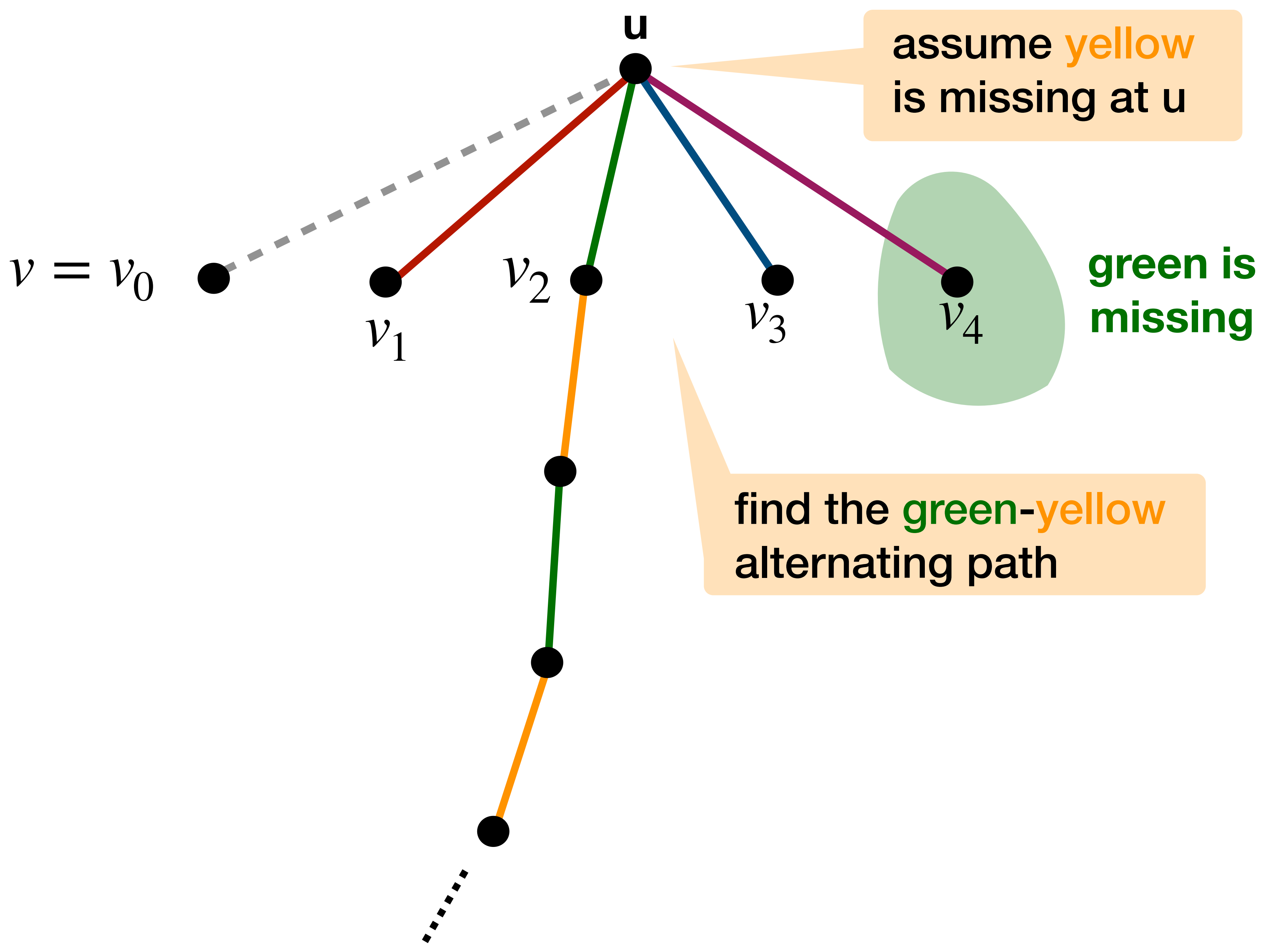
A review of [Viz'64]



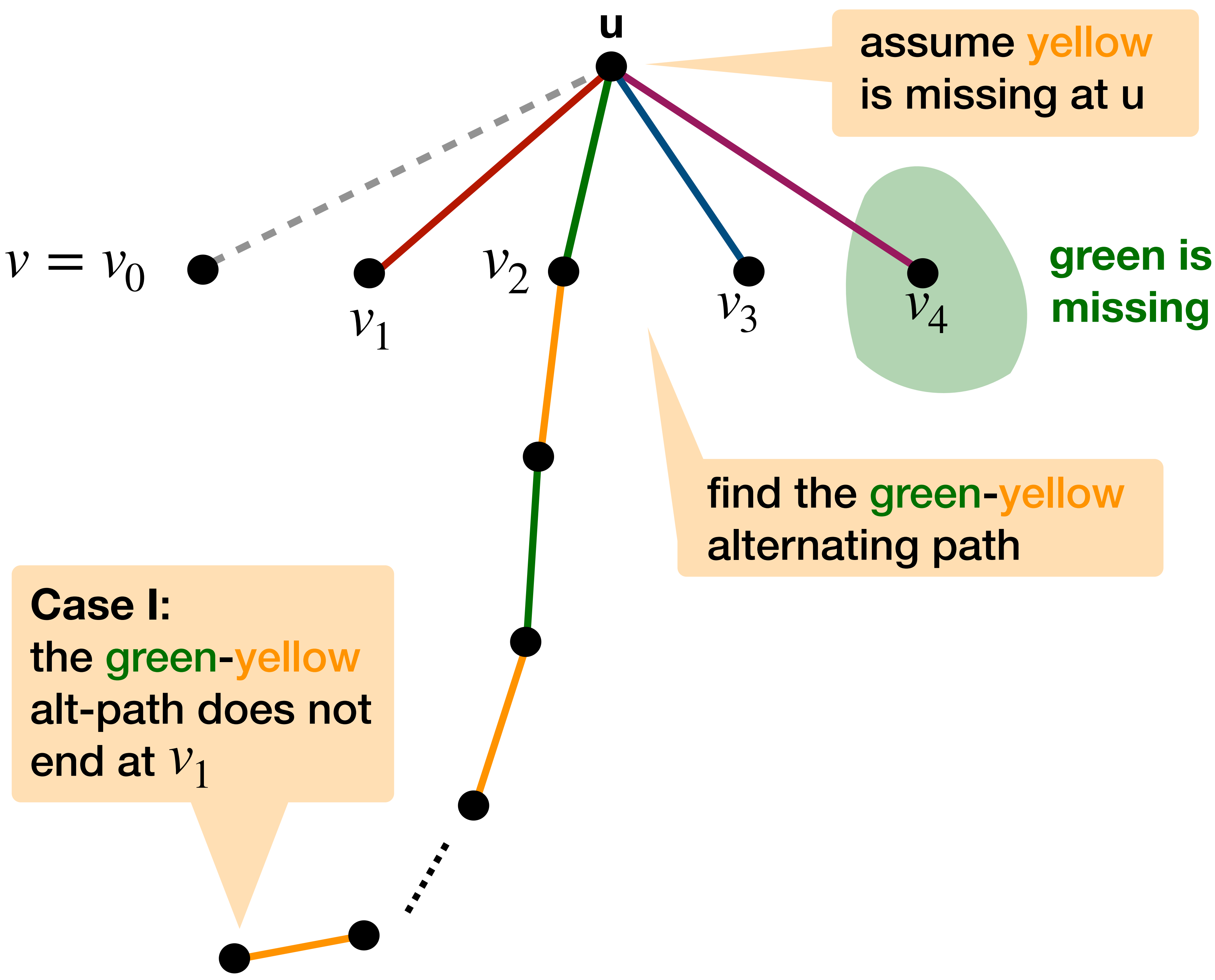
A review of [Viz'64]



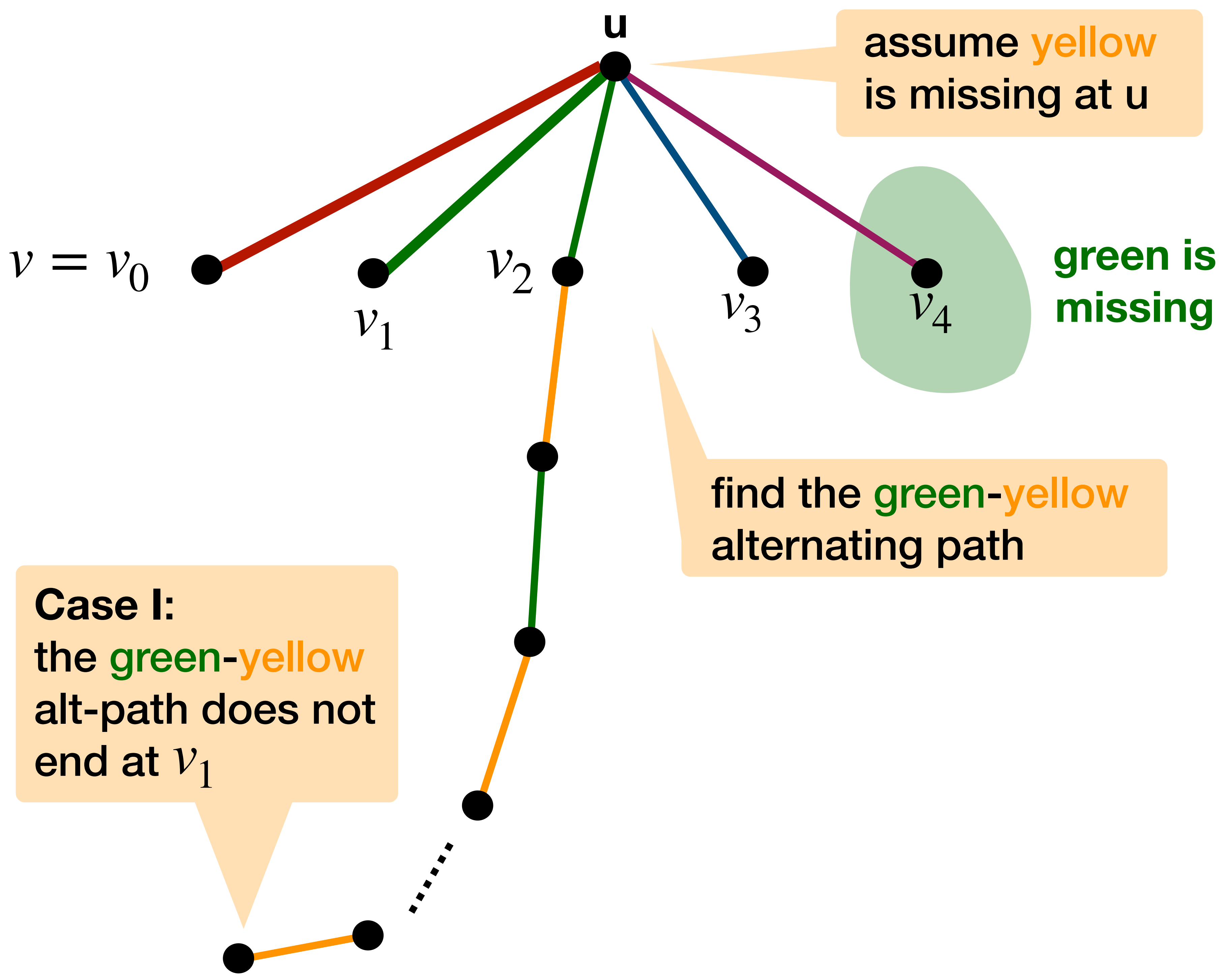
A review of [Viz'64]



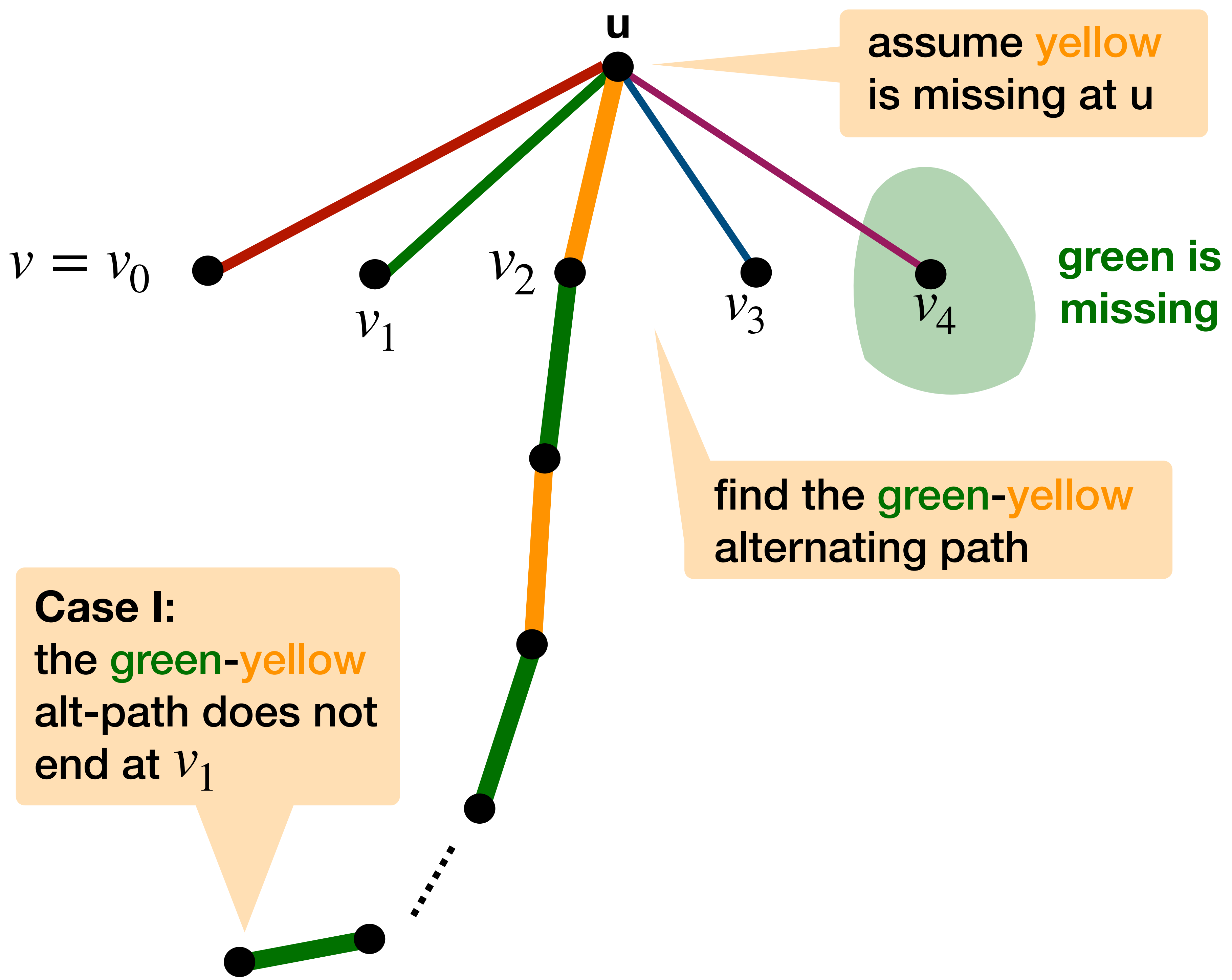
A review of [Viz'64]



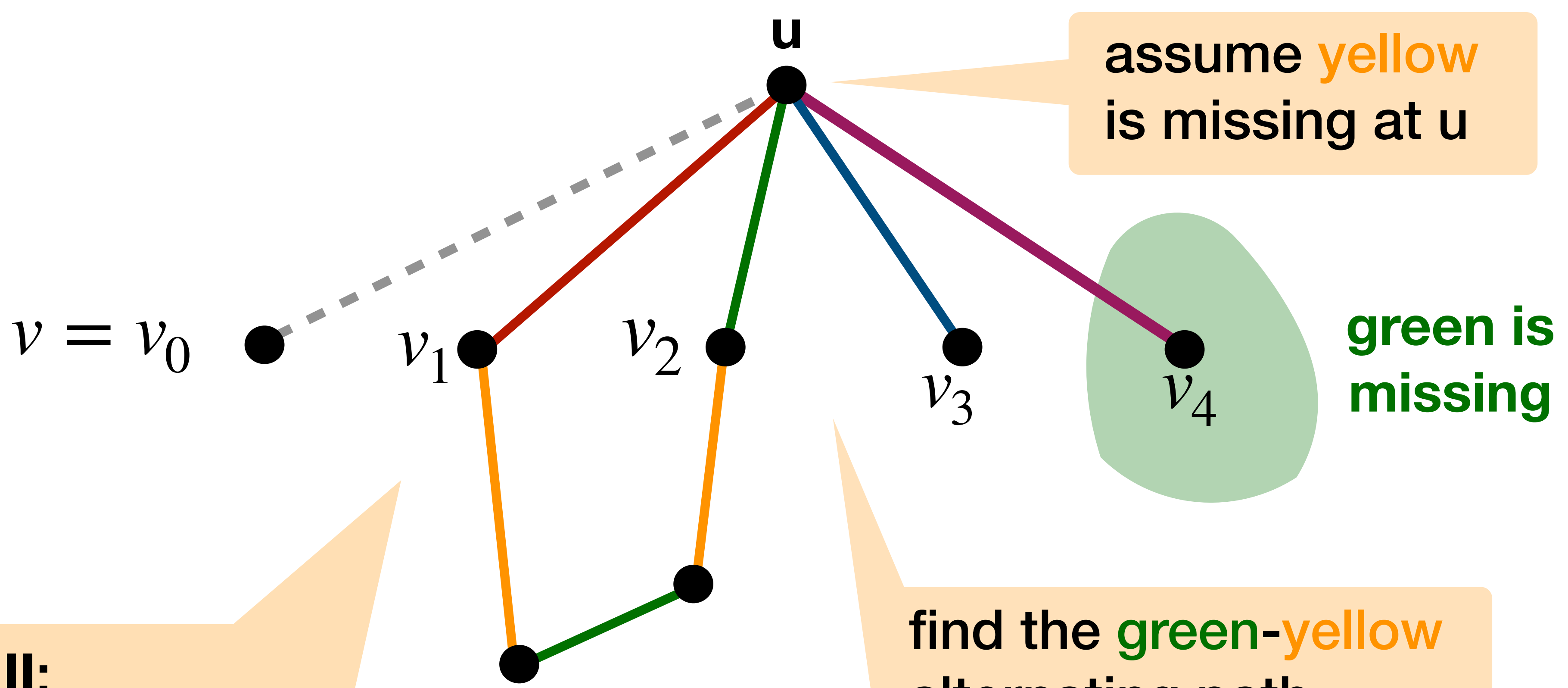
A review of [Viz'64]



A review of [Viz'64]



A review of [Viz'64]

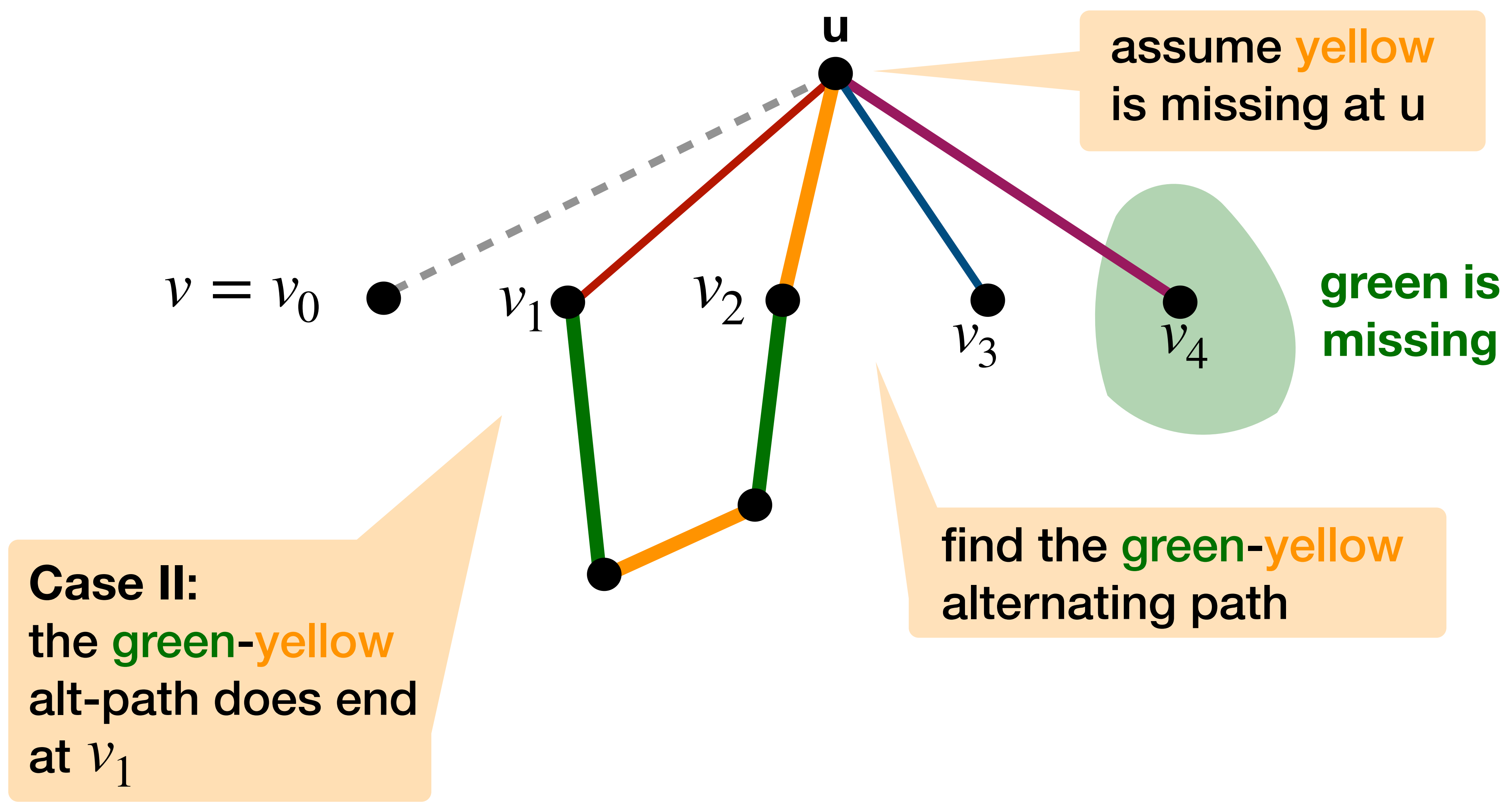


assume yellow is missing at u

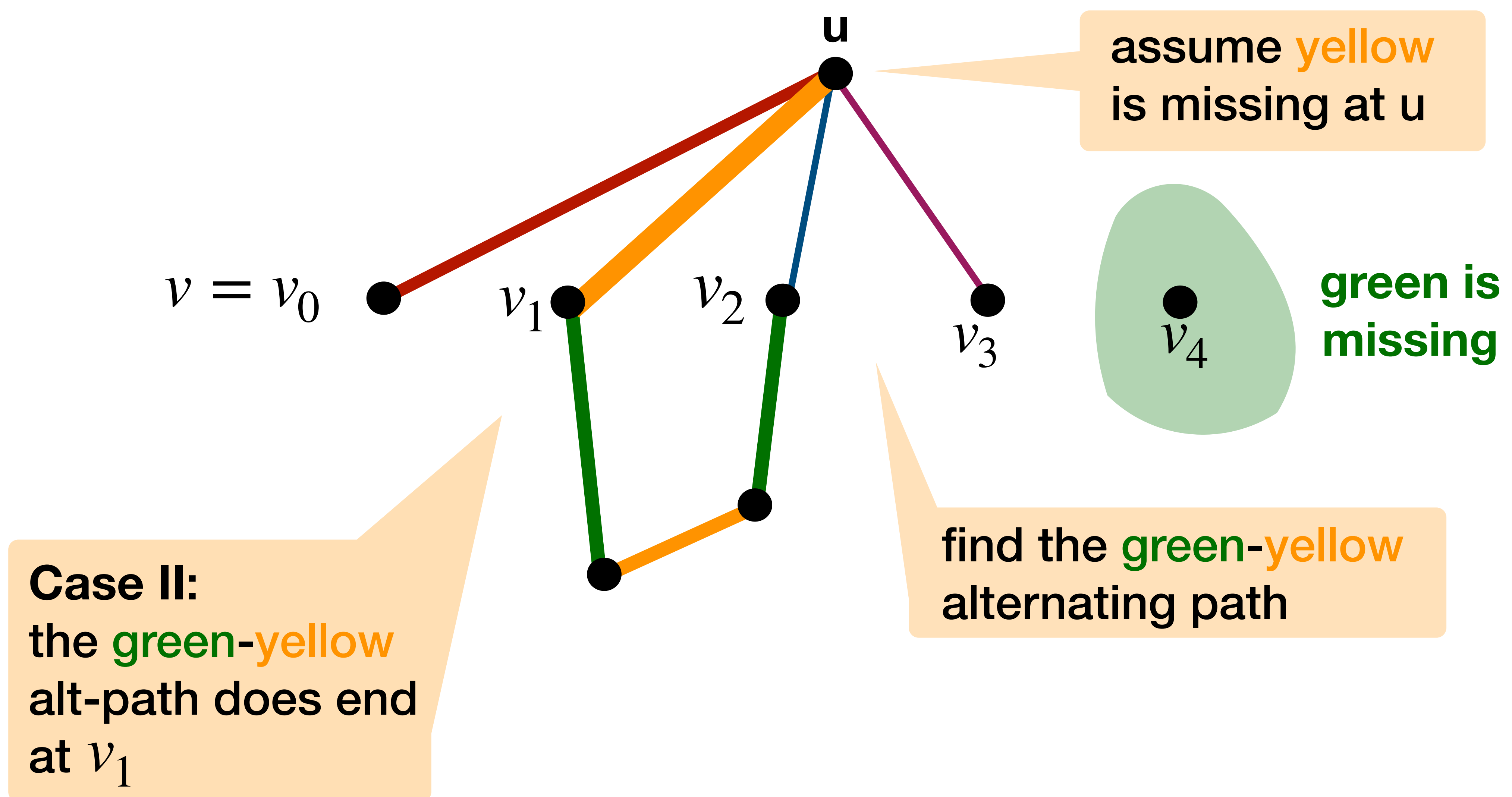
find the green-yellow alternating path

Case II:
the green-yellow alt-path does end at v_1

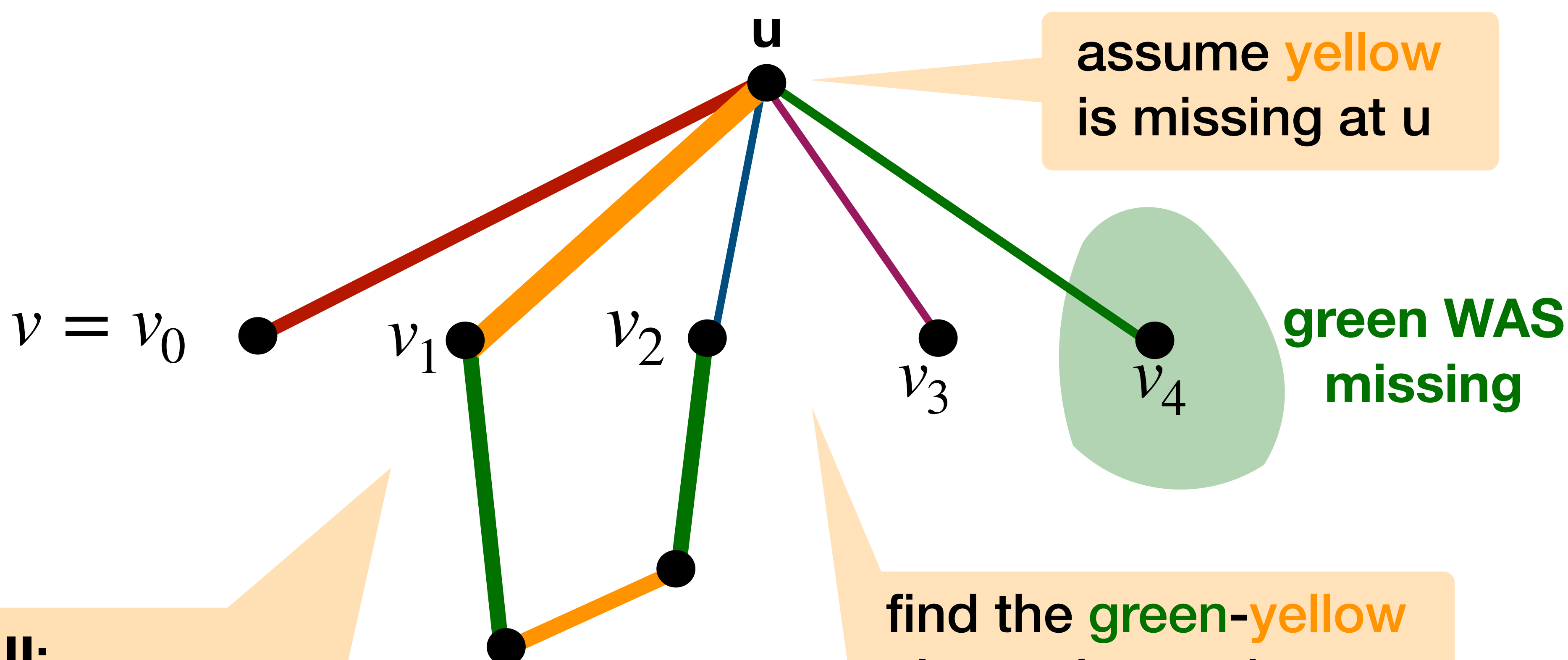
A review of [Viz'64]



A review of [Viz'64]



A review of [Viz'64]



assume yellow is missing at u

find the green-yellow alternating path

Case II:
the green-yellow alt-path does end at v_1

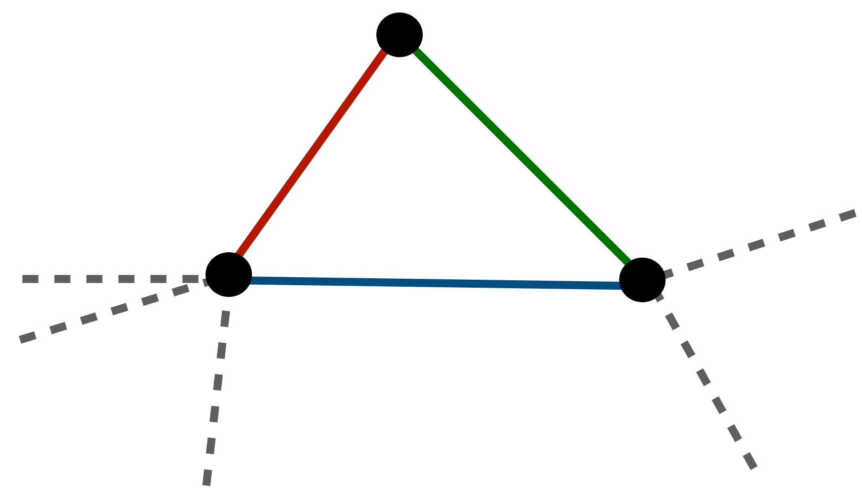
A review of [Viz'64]

Two bottlenecks in maintaining a $\Delta + 1$ coloring

- **Maximal chain**: $\tilde{O}(\Delta)$
- **Alternating path**: $\tilde{O}(L)$, L being the length of alt-path
- Total time: $\tilde{O}(\Delta + L) = \tilde{O}(n)$
- How to improve these two terms using $(1+\epsilon)\Delta$ colors?

Definition: A color subset is called a **palette**, if no vertex contains the entire palette in its neighborhood

Subset {red, green, blue} makes a palette for this partially colored graph



Lemma: For any partially $(1+\epsilon)\Delta$ colored graph, a random color subset of size $O(\log n/\epsilon)$ makes a **palette**, w.h.p.

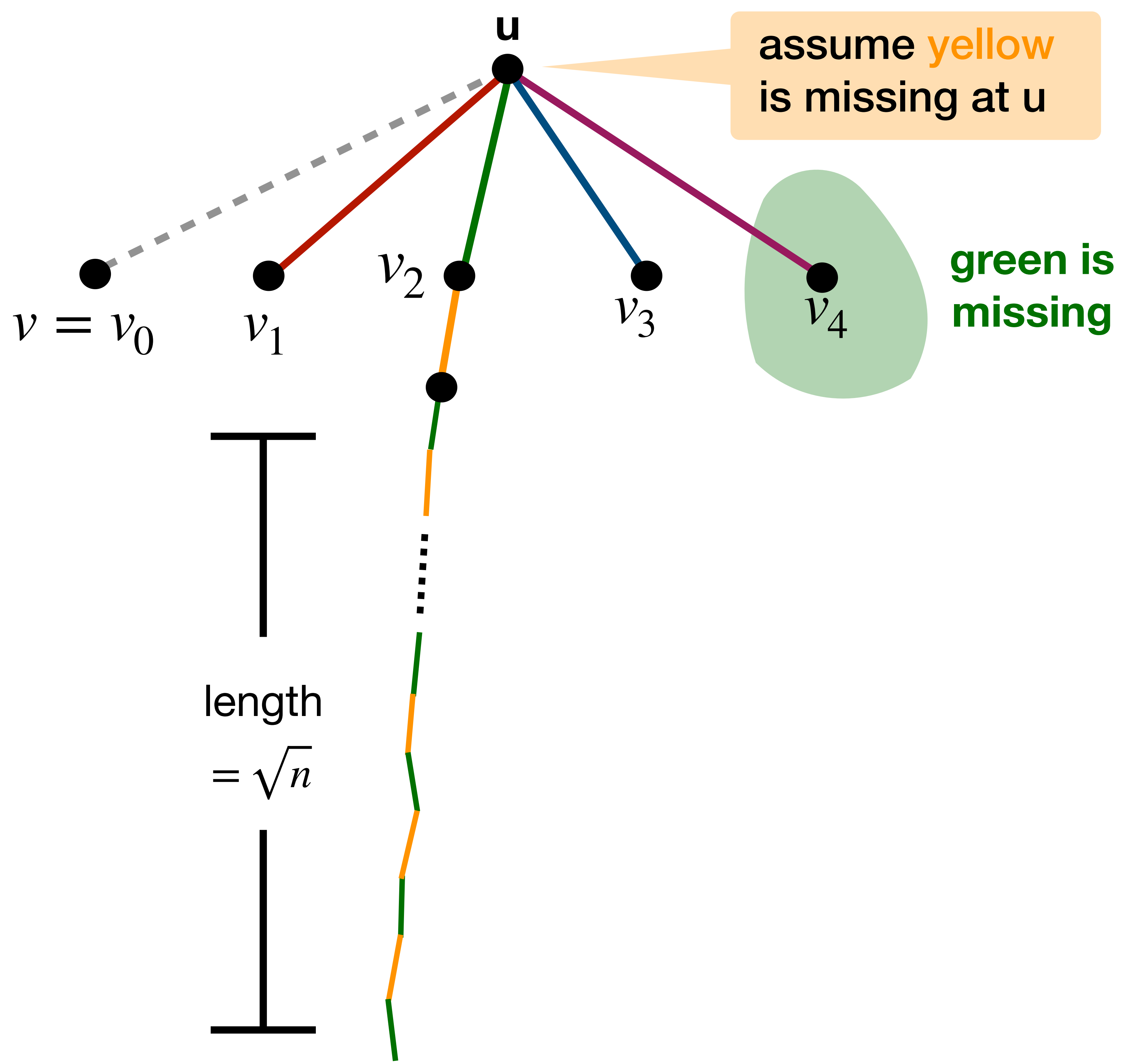
- Run Vizing's algorithm only using colors from a **palette**
- Maximal chains have length at most $O(\log n/\epsilon)$
- How about alternating paths?

An $\tilde{O}(\sqrt{n})$ update time algorithm

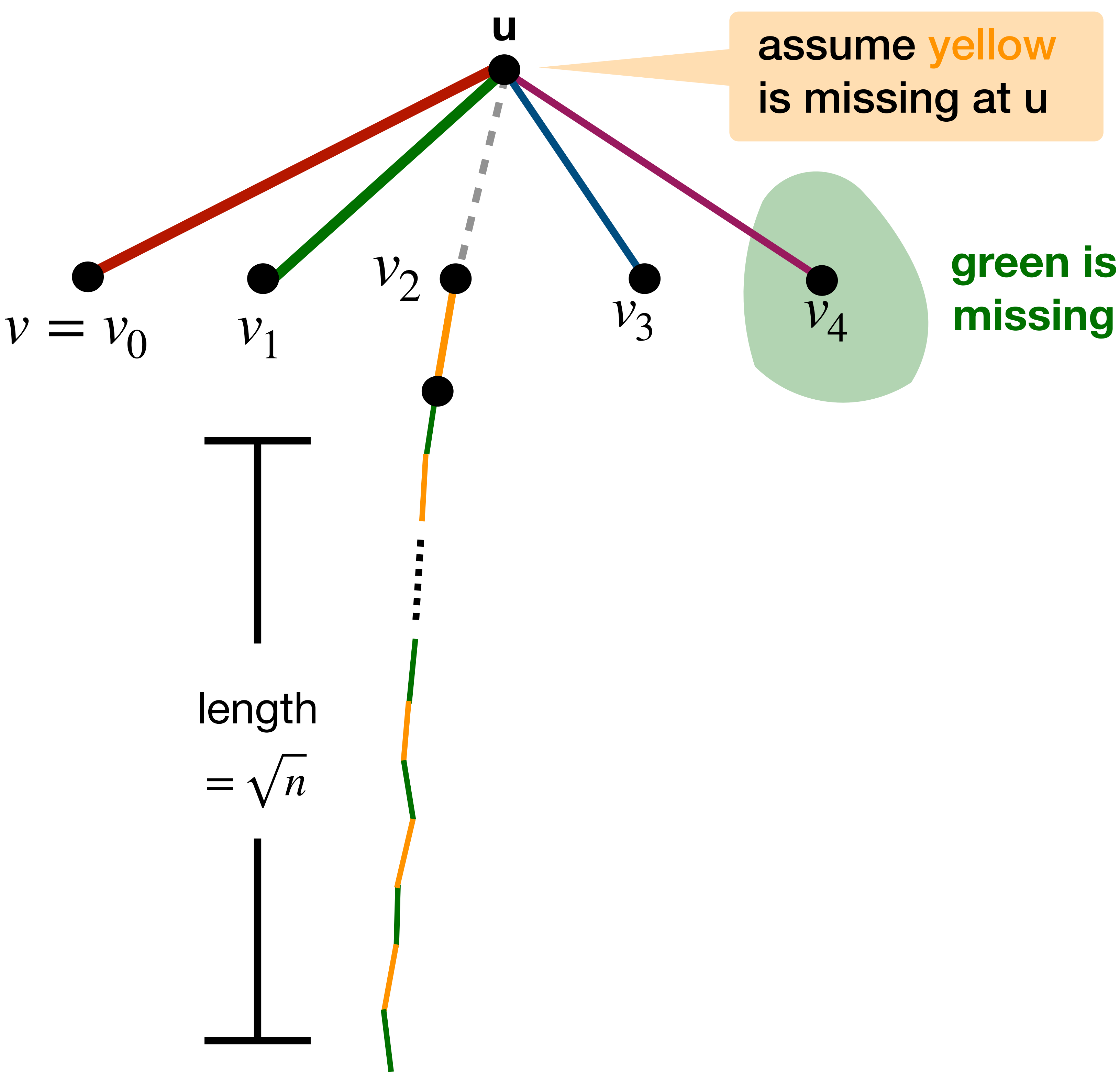
Algorithm: If the alternating path is long, then translate the uncolored edge to a **random position** on the path, and reapply Vizing's algorithm

Observation: **Most positions** on this alternating path are good for applying Vizing's algorithm

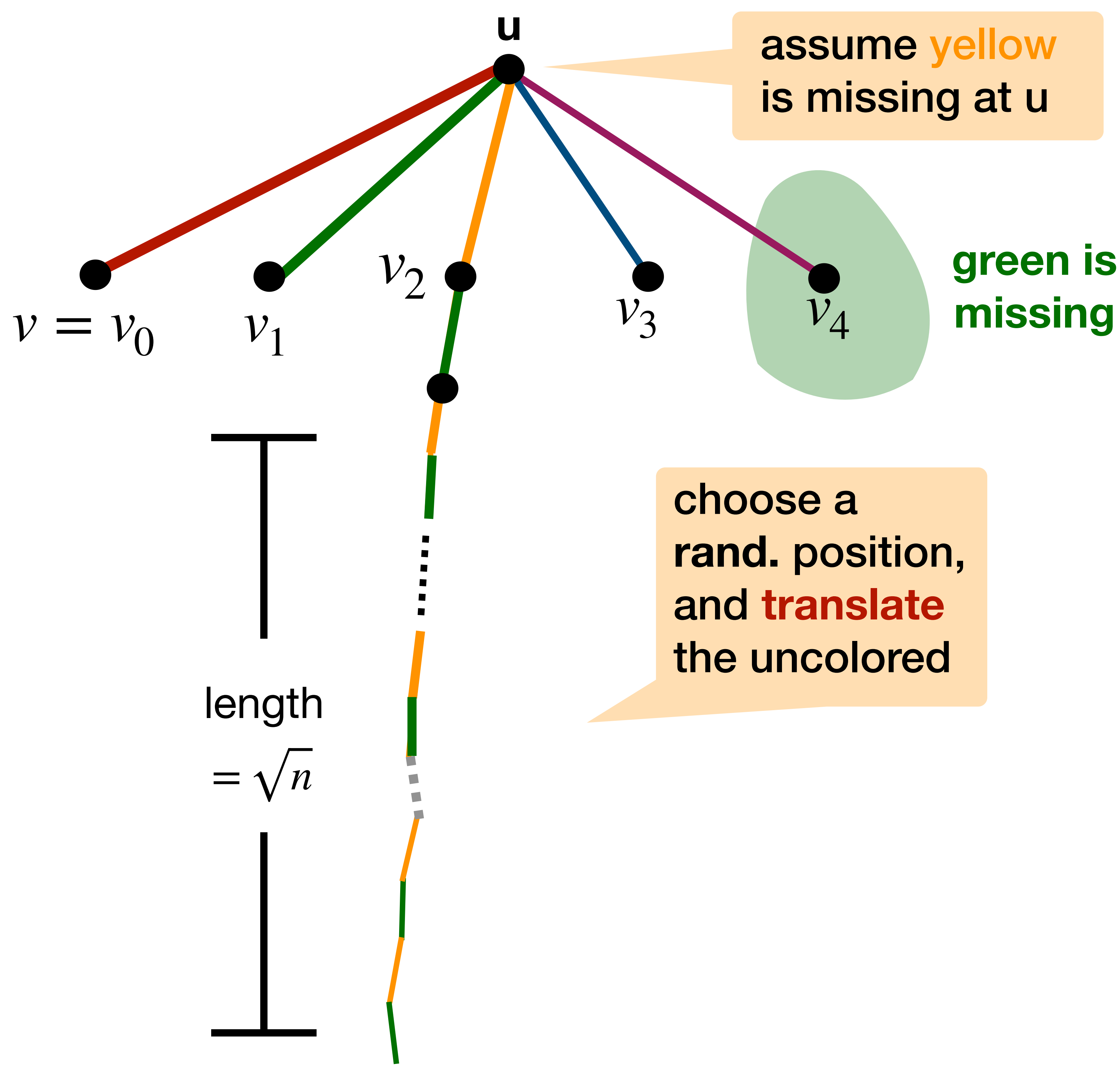
An $\tilde{O}(\sqrt{n})$ update time algorithm



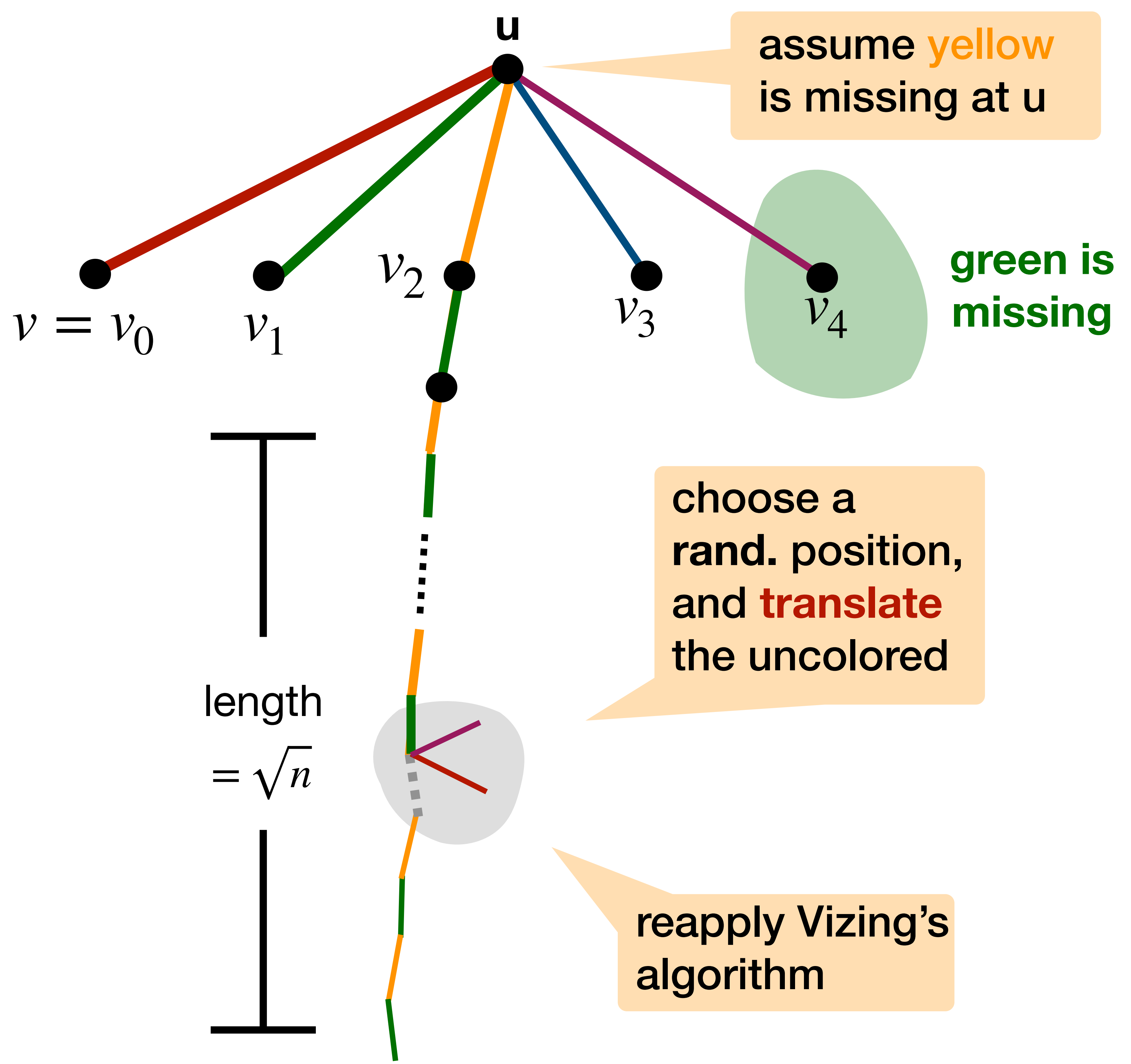
An $\tilde{O}(\sqrt{n})$ update time algorithm



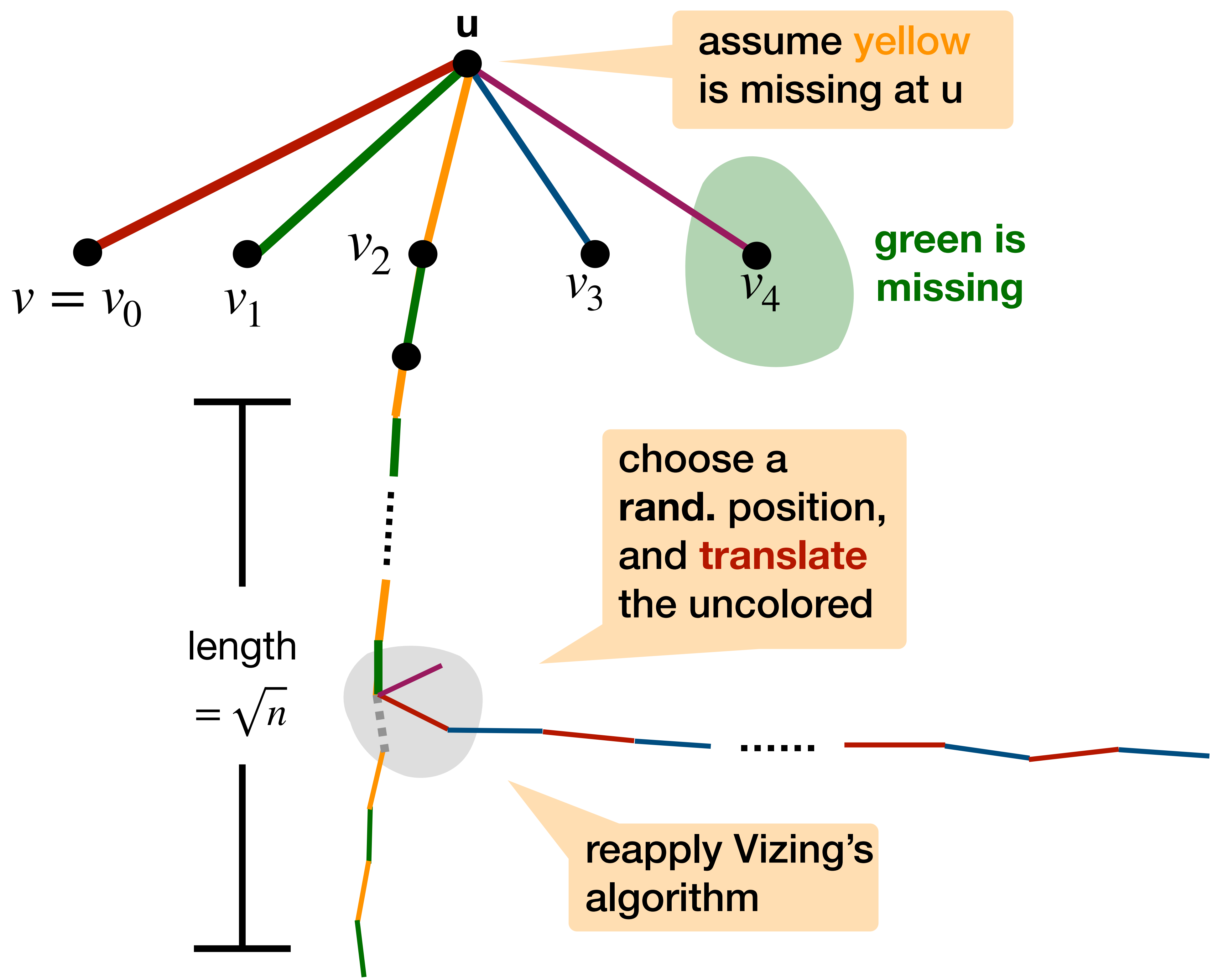
An $\tilde{O}(\sqrt{n})$ update time algorithm



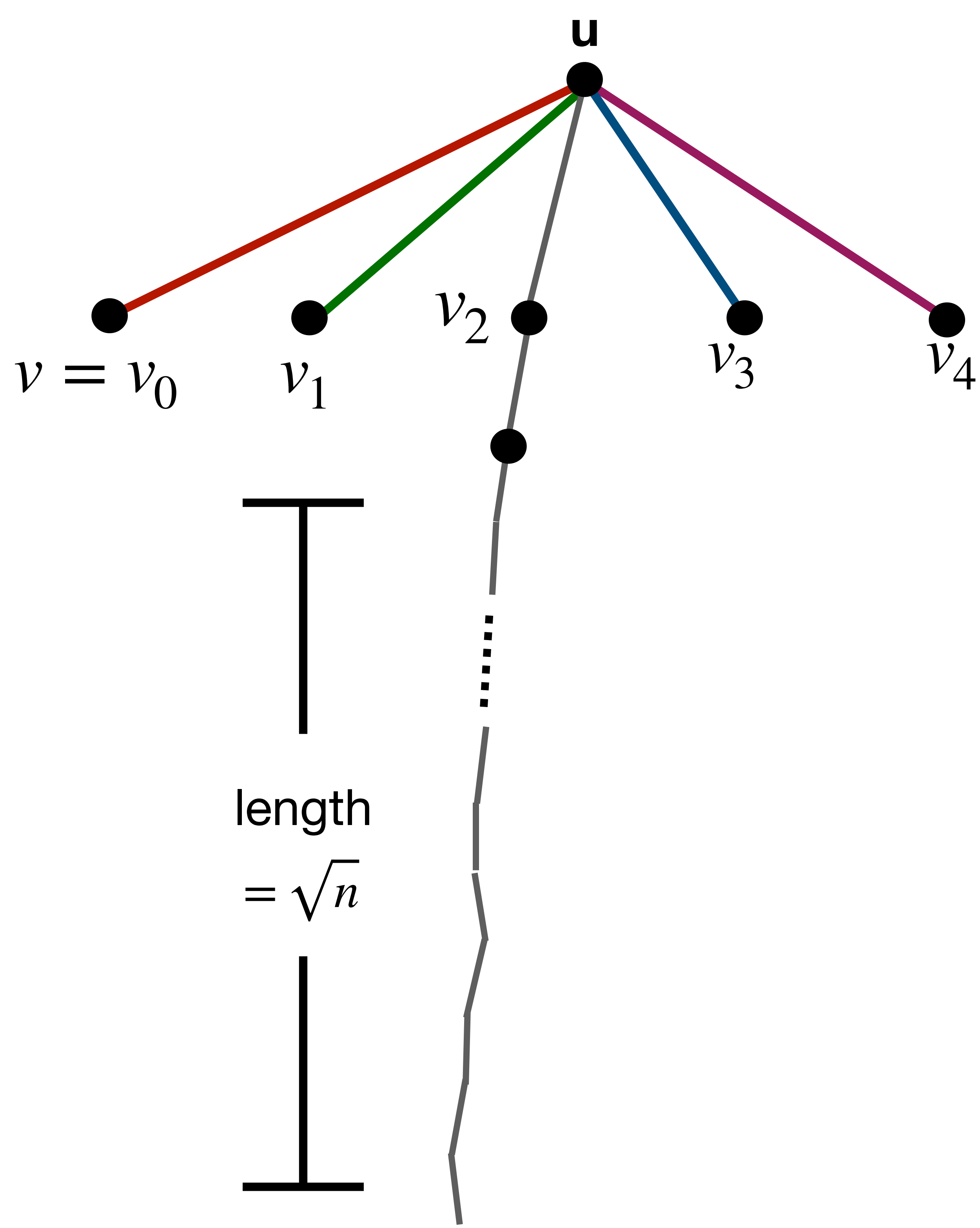
An $\tilde{O}(\sqrt{n})$ update time algorithm



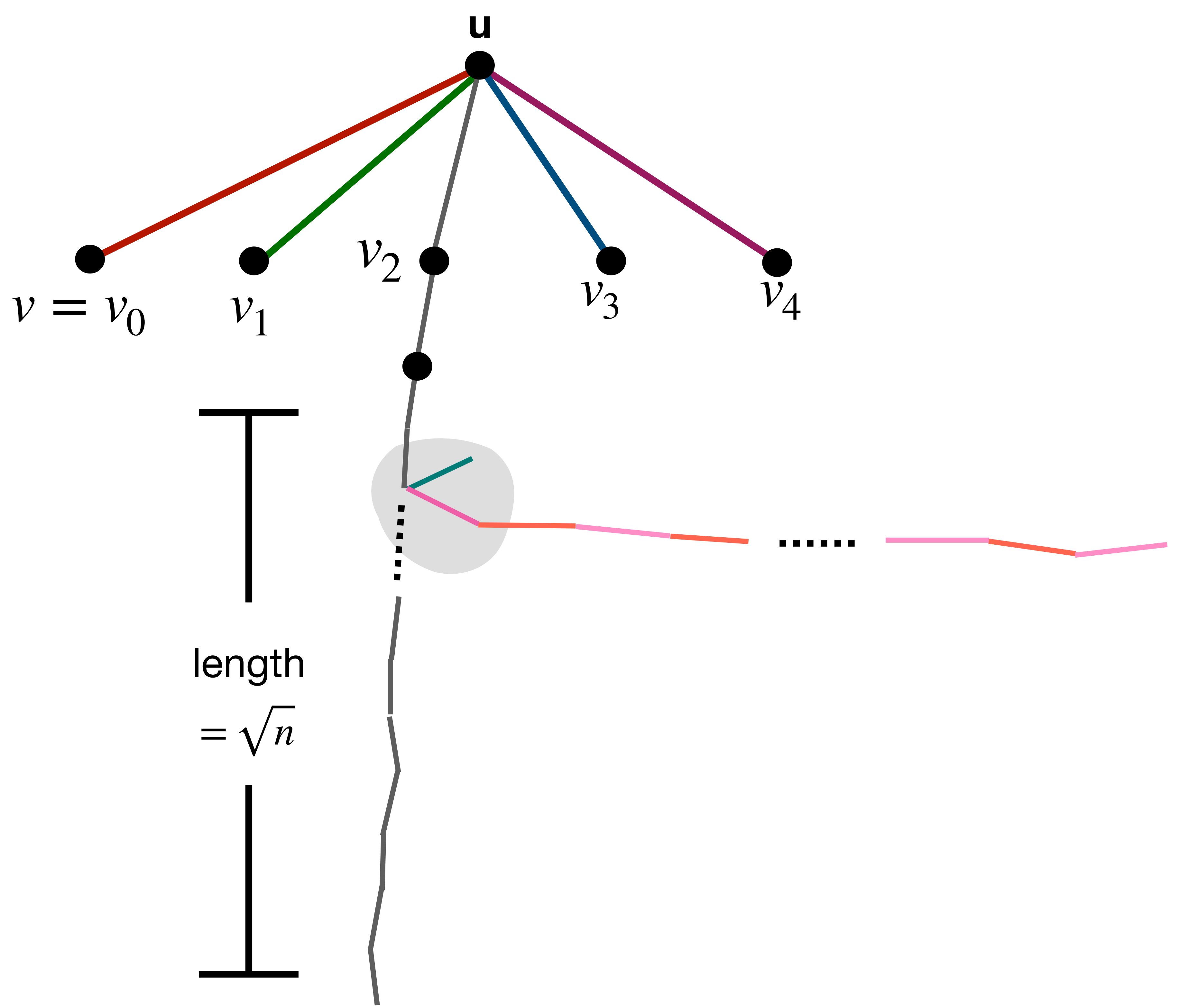
An $\tilde{O}(\sqrt{n})$ update time algorithm



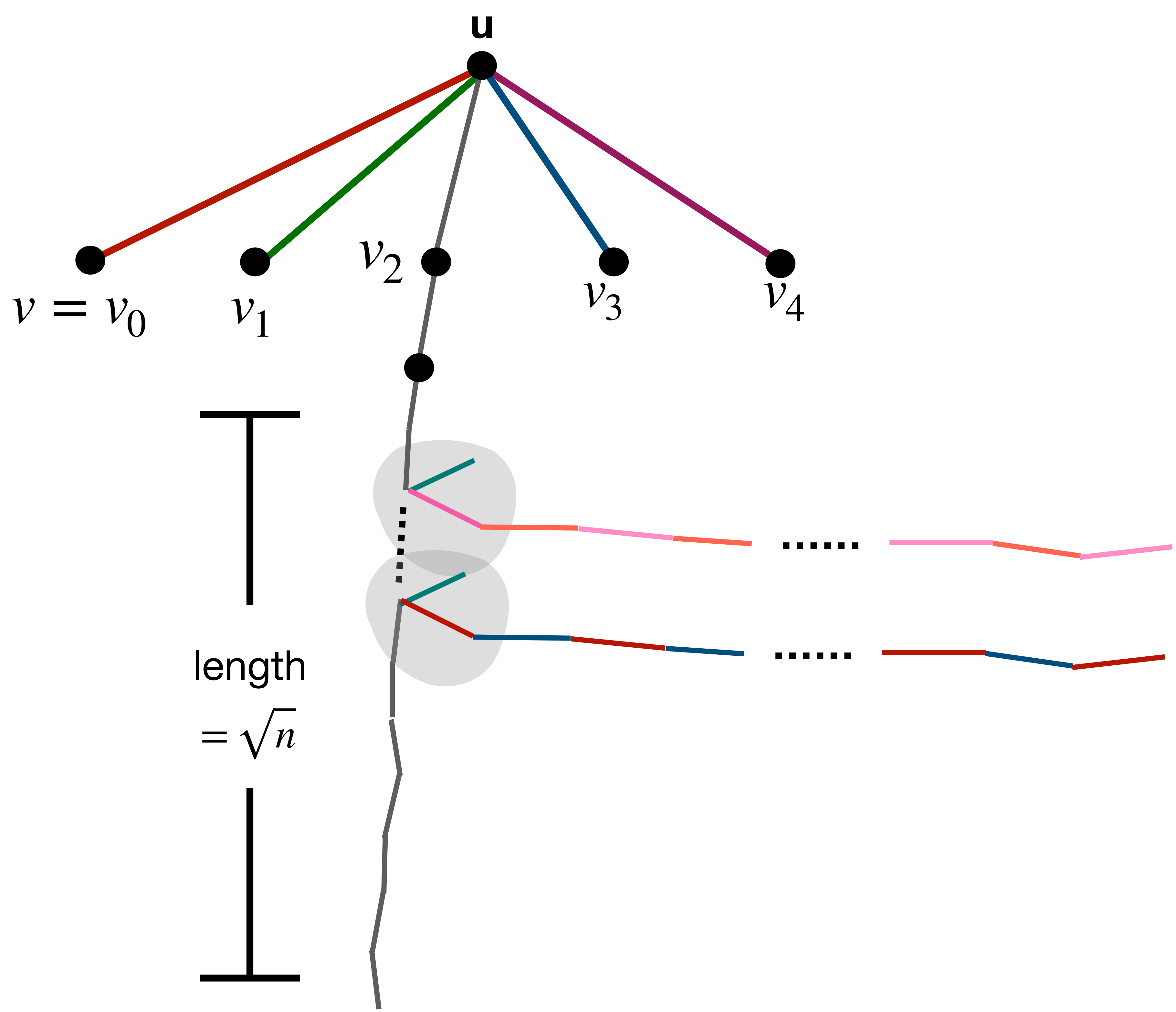
An $\tilde{O}(\sqrt{n})$ update time algorithm



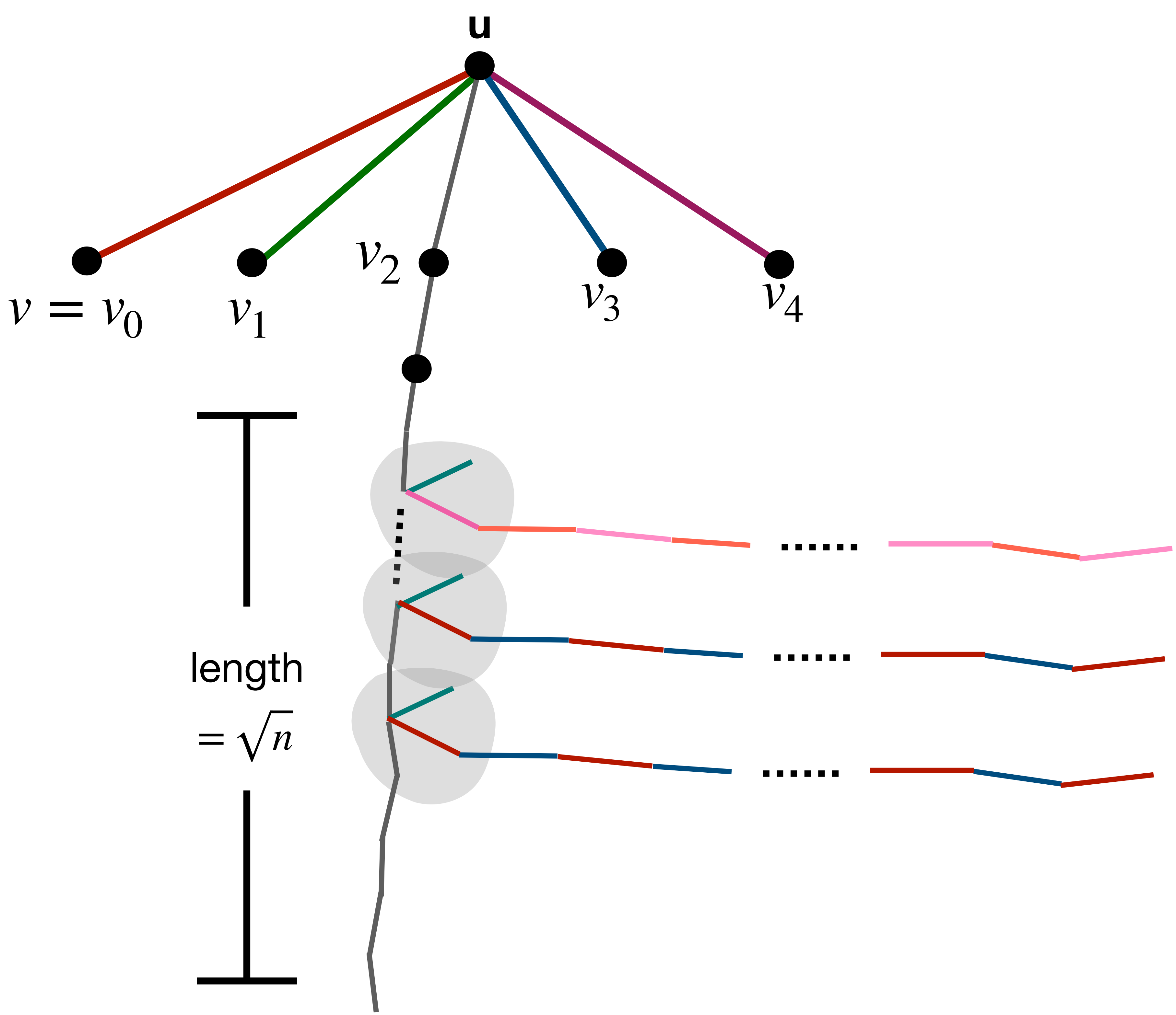
An $\tilde{O}(\sqrt{n})$ update time algorithm



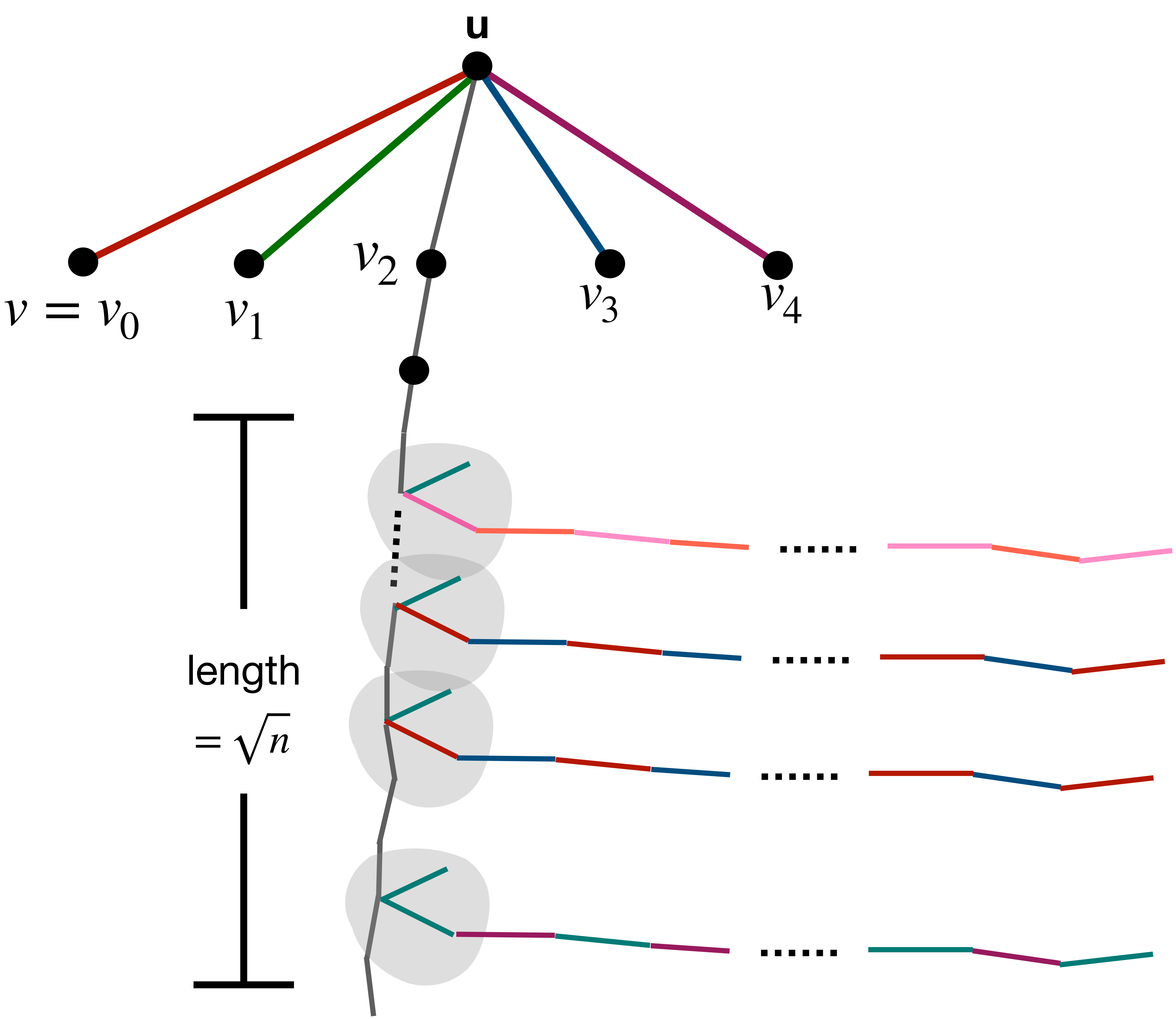
An $\tilde{O}(\sqrt{n})$ update time algorithm



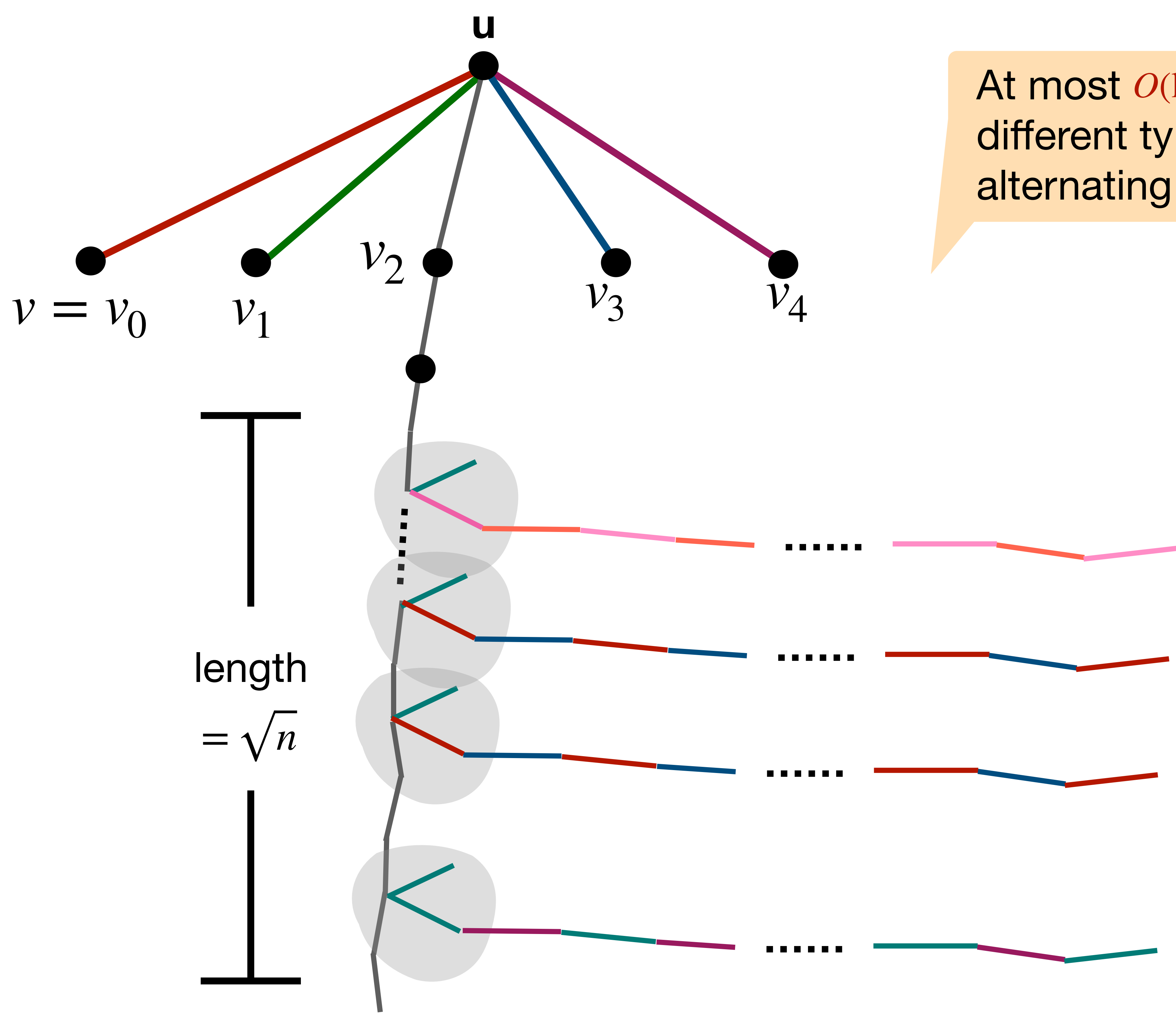
An $\tilde{O}(\sqrt{n})$ update time algorithm



An $\tilde{O}(\sqrt{n})$ update time algorithm

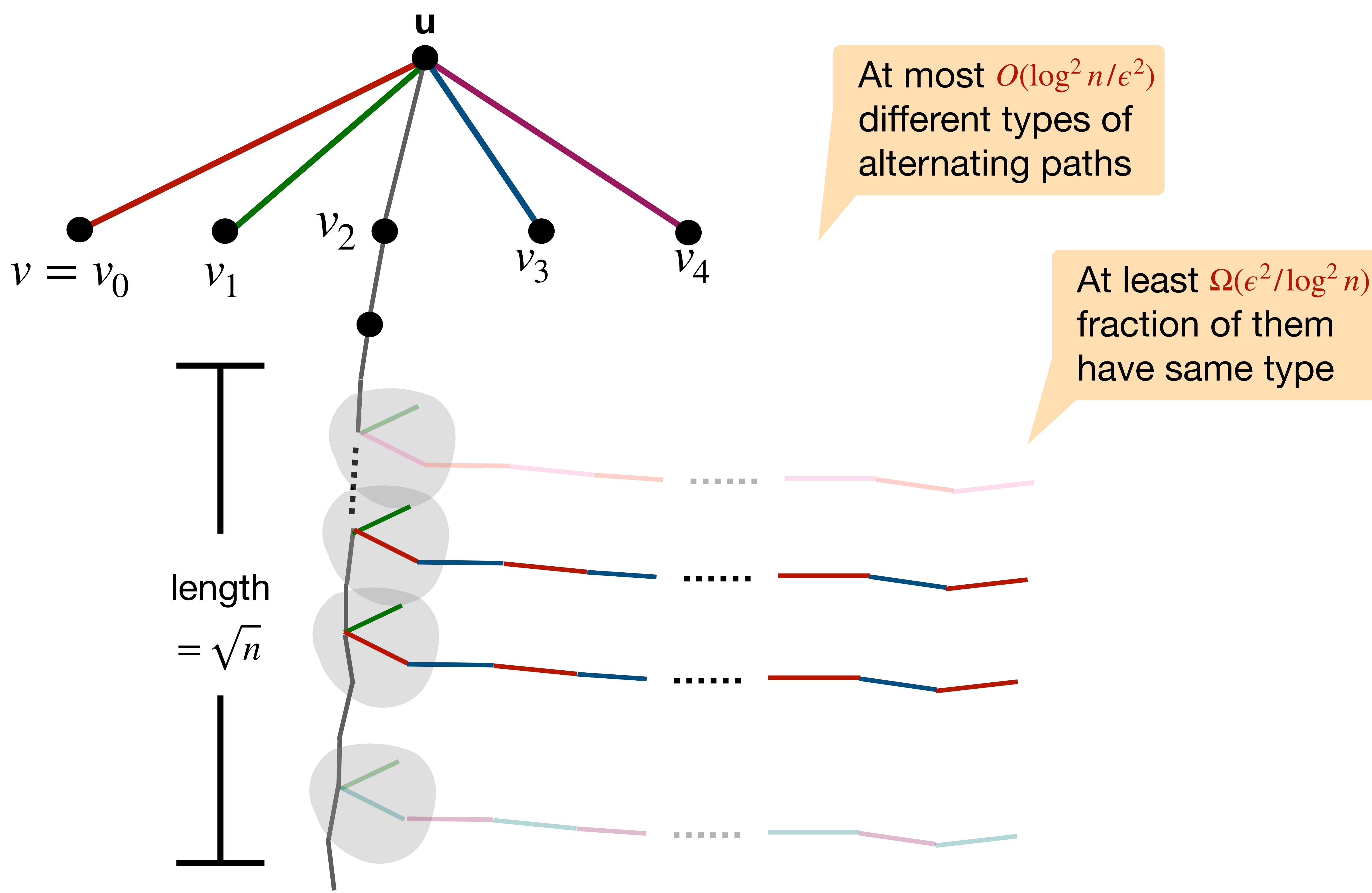


An $\tilde{O}(\sqrt{n})$ update time algorithm

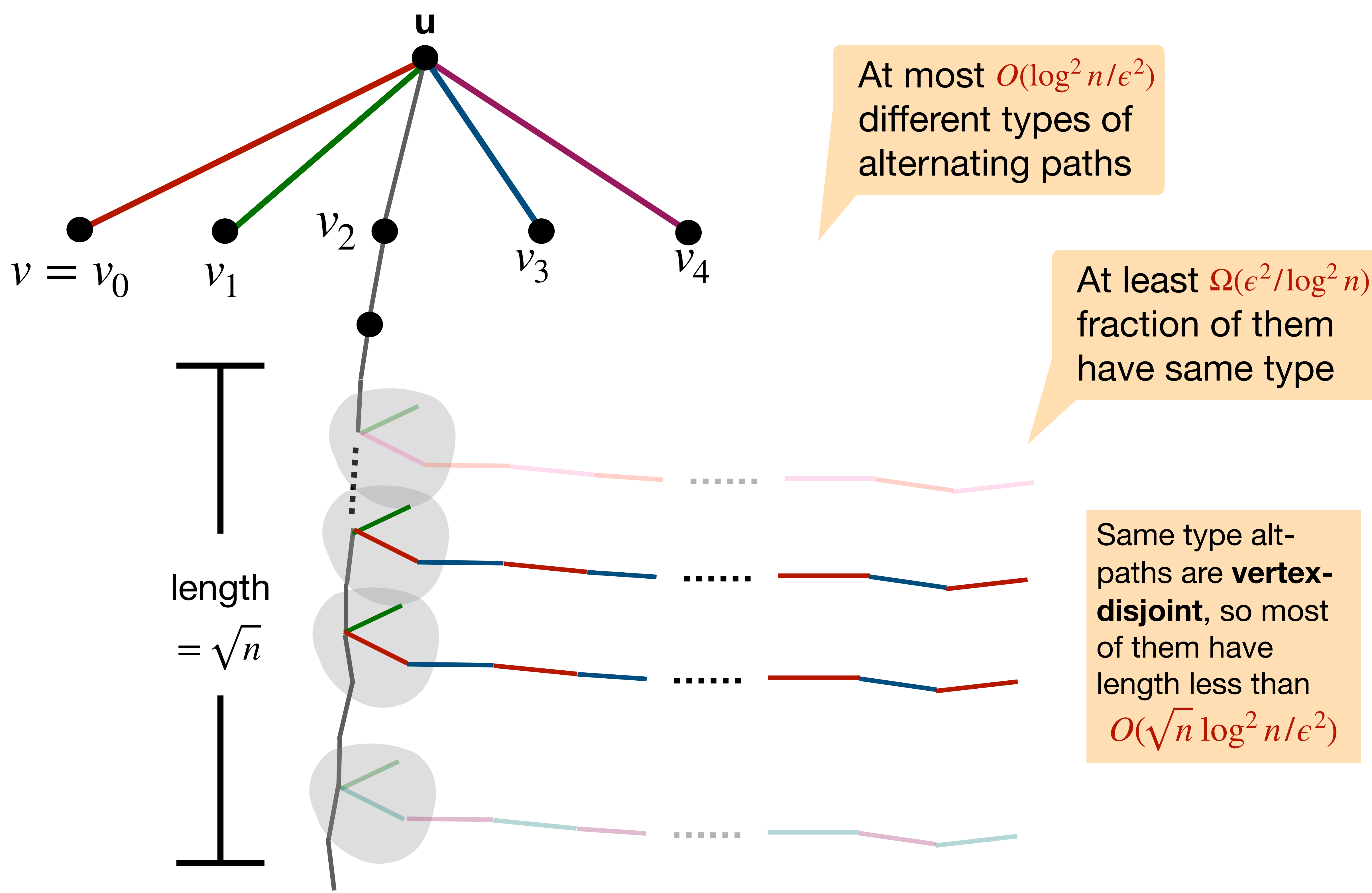


At most $O(\log^2 n / \epsilon^2)$ different types of alternating paths

An $\tilde{O}(\sqrt{n})$ update time algorithm



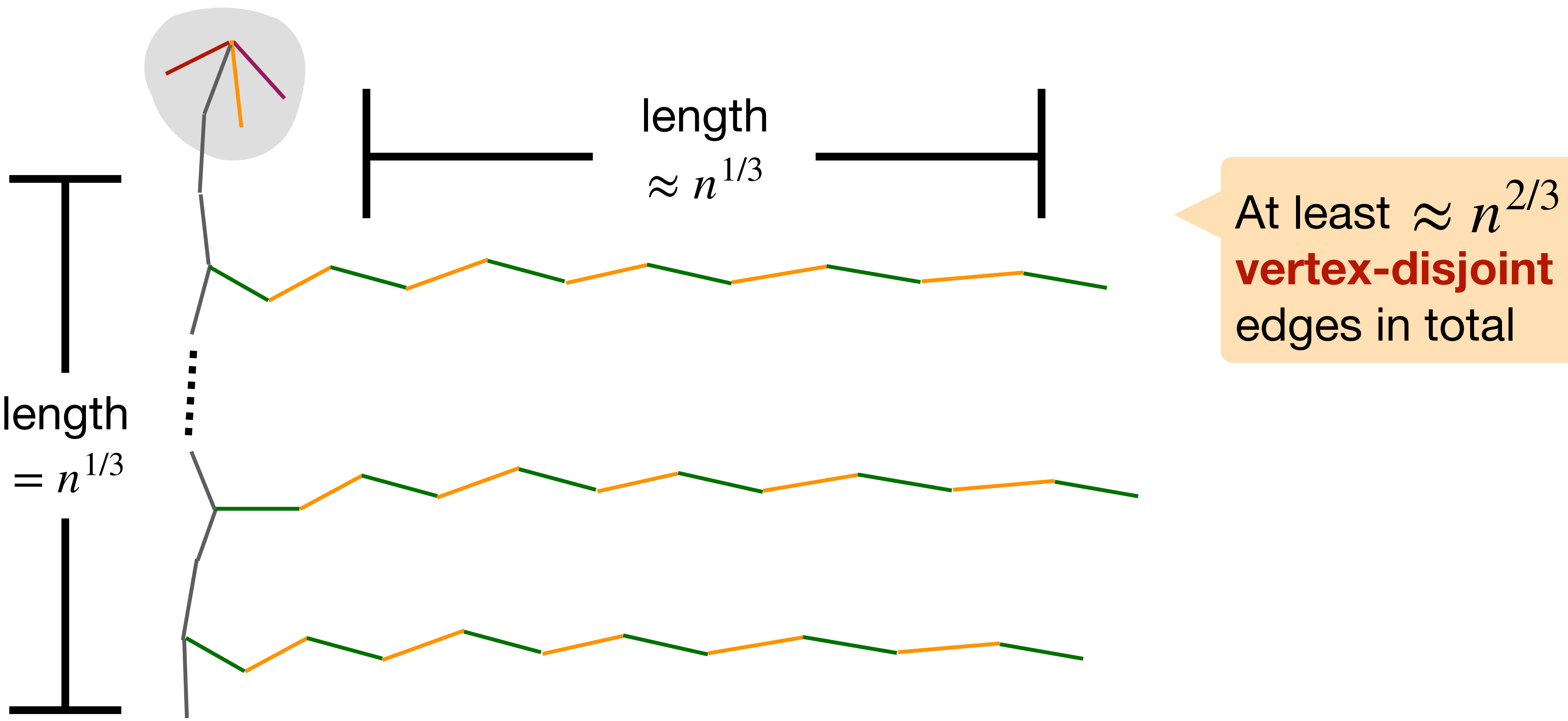
An $\tilde{O}(\sqrt{n})$ update time algorithm



Improving to sub-poly update time

Idea: Translate **multiple** times, until alt-path length is $\leq h$

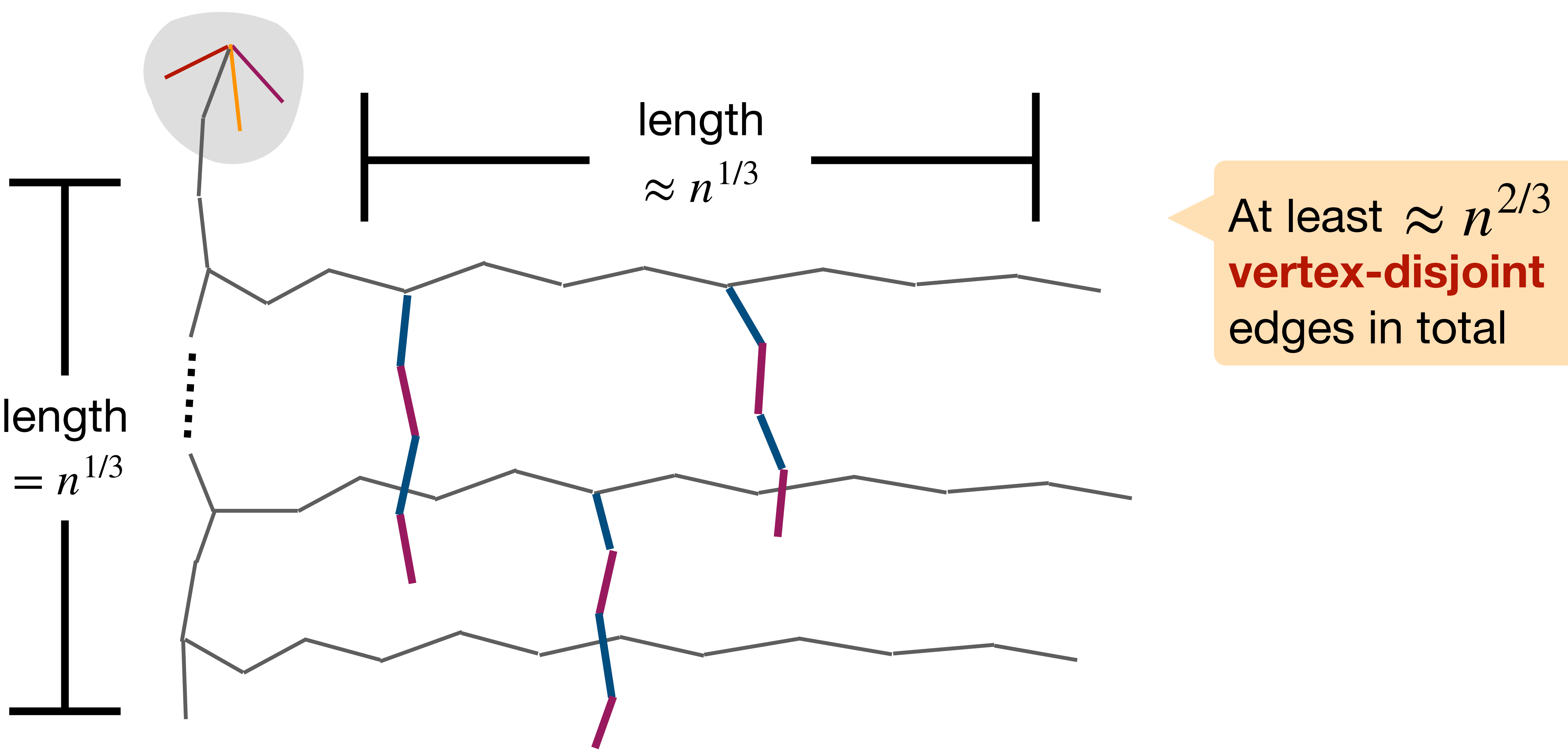
Example: Translate **twice** and we have $\tilde{O}(n^{1/3})$ update time



Improving to sub-poly update time

Idea: Translate **multiple** times, until alt-path length is $\leq h$

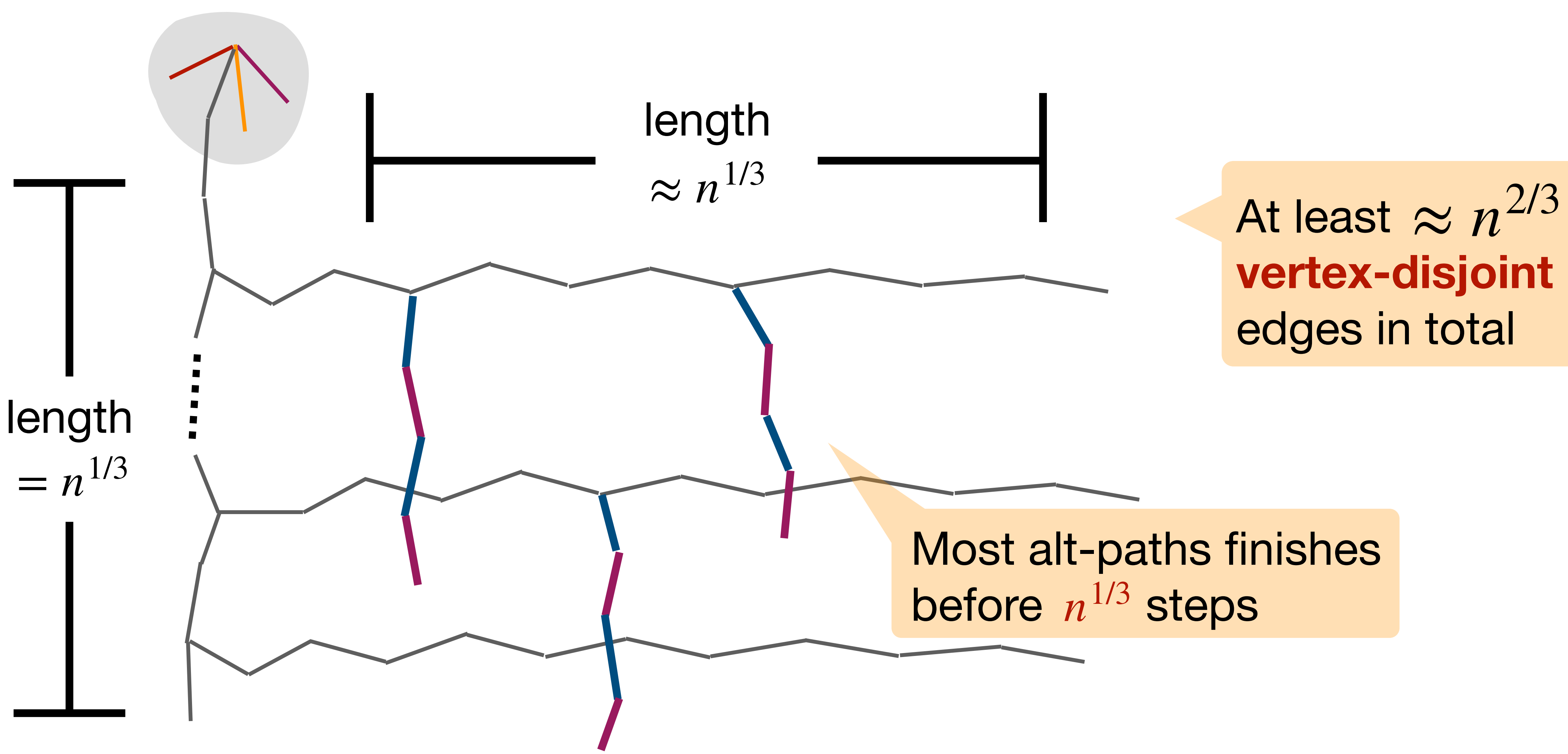
Example: Translate **twice** and we have $\tilde{O}(n^{1/3})$ update time



Improving to sub-poly update time

Idea: Translate **multiple** times, until alt-path length is $\leq h$

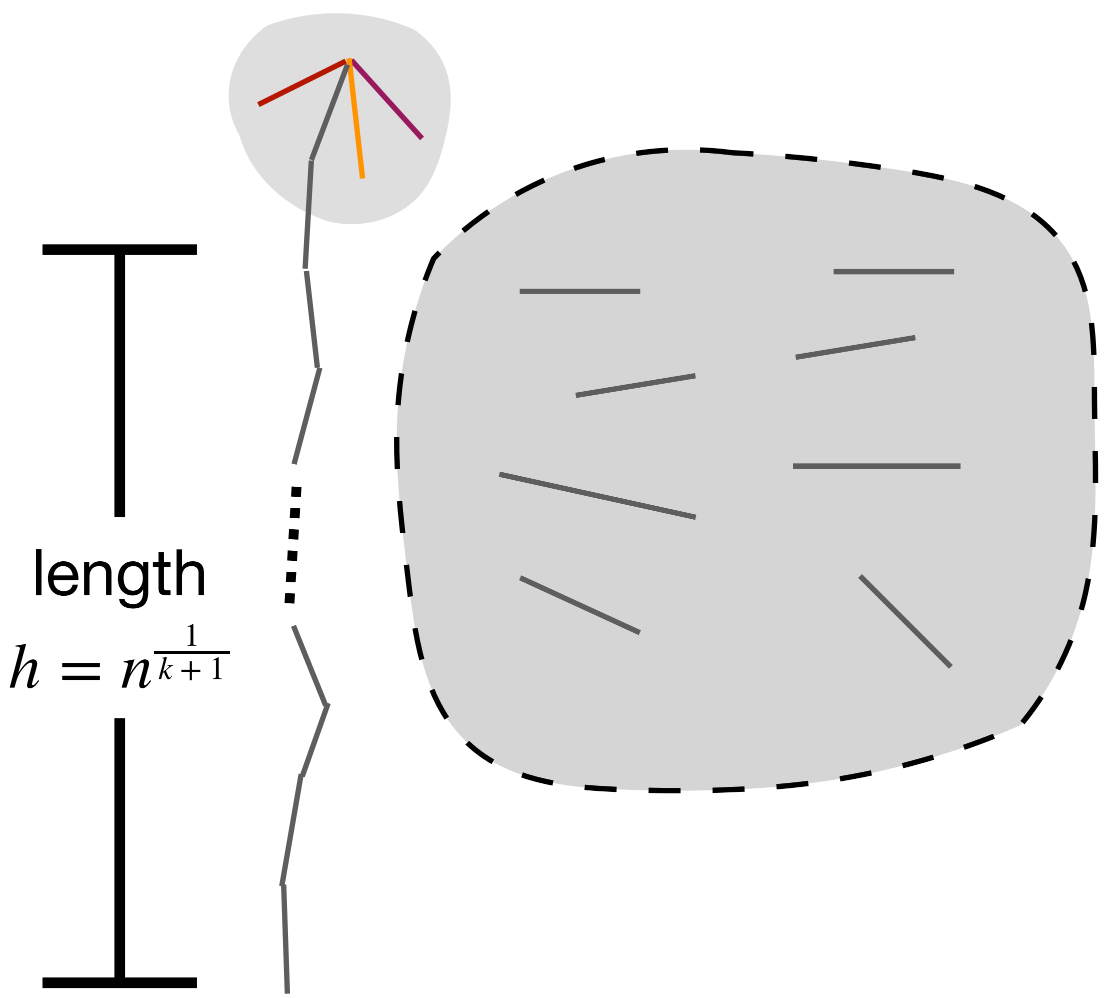
Example: Translate **twice** and we have $\tilde{O}(n^{1/3})$ update time



Improving to sub-poly update time

Idea: Translate **multiple** times, until alt-path length is $\leq h$

Example: Translate **k-times** and we have $\tilde{O}(n^{\frac{1}{k+1}})$ update time

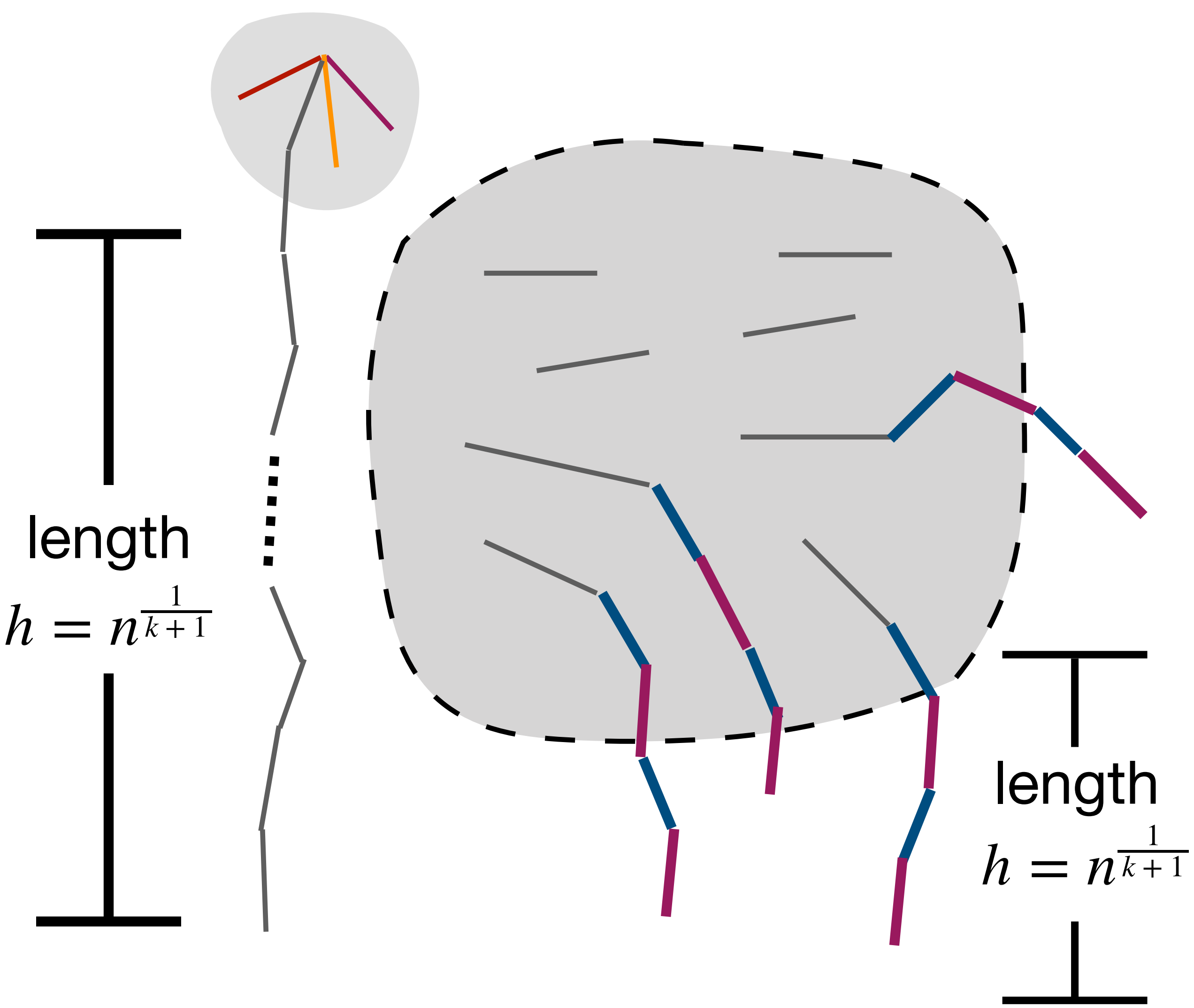


Assume after **i-th** translation, the uncolored edge is uniformly distributed among an edge set of size at least $\left(\epsilon^2/\log^2 n\right)^{i-1} \cdot n^{\frac{i}{k+1}}$

Improving to sub-poly update time

<u>Idea</u>: Translate **multiple** times, until alt-path length is $\leq h$

<u>Example</u>: Translate **k-times** and we have $\tilde{O}(n^{\frac{1}{k+1}})$ update time



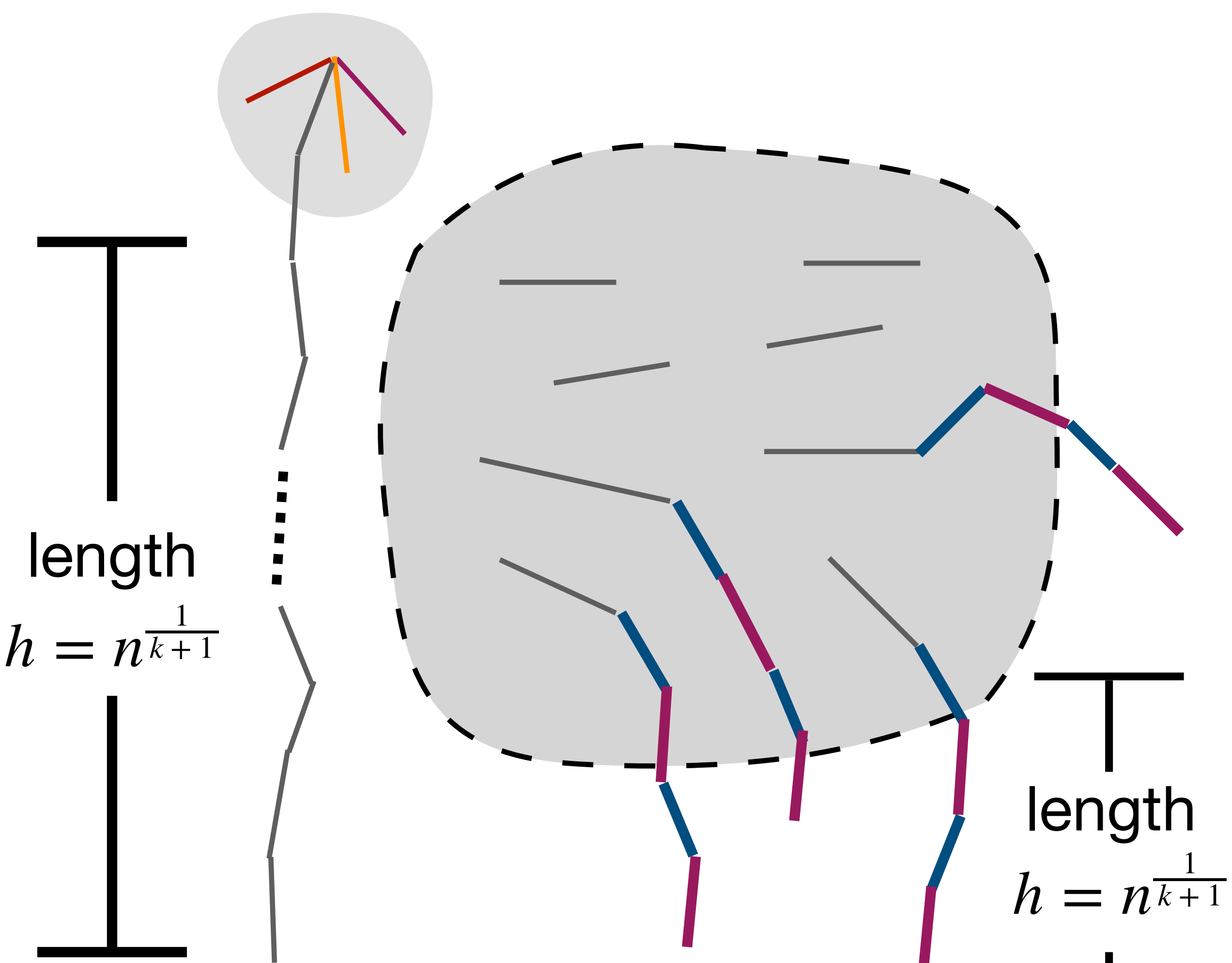
Assume after **i-th** translation, the uncolored edge is uniformly distributed among an edge set of size at least $\left(\epsilon^2/\log^2 n\right)^{i-1} \cdot n^{\frac{i}{k+1}}$

At least $\Omega(\epsilon^2/\log^2 n)$ fraction of alt-paths in the next round have the **same type**

Improving to sub-poly update time

<u>Idea</u>: Translate **multiple** times, until alt-path length is $\leq h$

<u>Example</u>: Translate **k-times** and we have $\tilde{O}(n^{\frac{1}{k+1}})$ update time



Assume after **i-th** translation, the uncolored edge is uniformly distributed among an edge set of size at least $\left(\epsilon^2/\log^2 n\right)^{i-1} \cdot n^{\frac{i}{k+1}}$

At least $\Omega(\epsilon^2/\log^2 n)$ fraction of alt-paths in the next round have the **same type**

Then after **(i+1)-th** translation, the uncolored edge is uniformly distributed among an edge set of size at least $\left(\epsilon^2/\log^2 n\right)^{i} \cdot n^{\frac{i+1}{k+1}}$

Refining the update time analysis

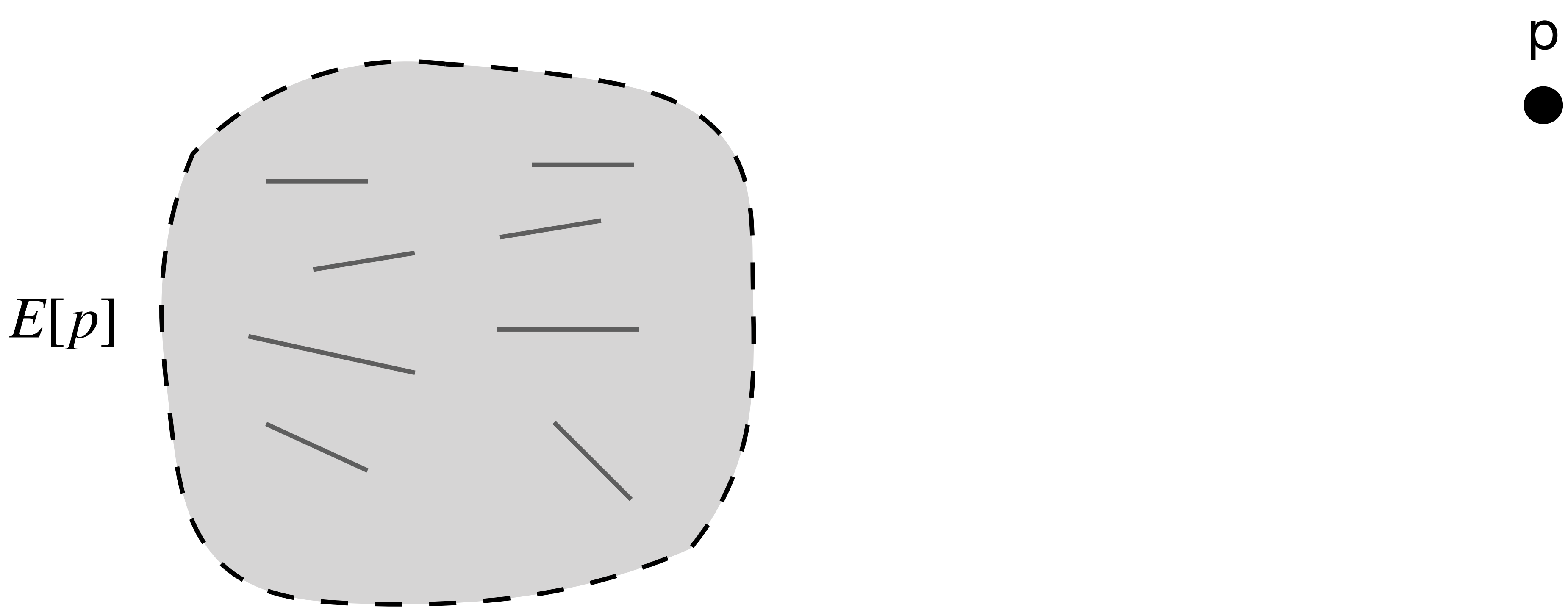
Bottleneck: Only take one type of alt-path that accounts for a fraction of $\Omega(\epsilon^2/\log^2 n)$

Refinement: Consider every type of alt-paths that accounts for a fraction of $\Omega(\epsilon^2/\log^3 n)$

Model the algorithm as a tree $\mathscr{T}$

Each node $p \in \mathscr{T}$ is associated with two fields:

- a probability mass $\mu[p] \in [0,1]$
- a set $E[p]$ of vertex-disjoint edges

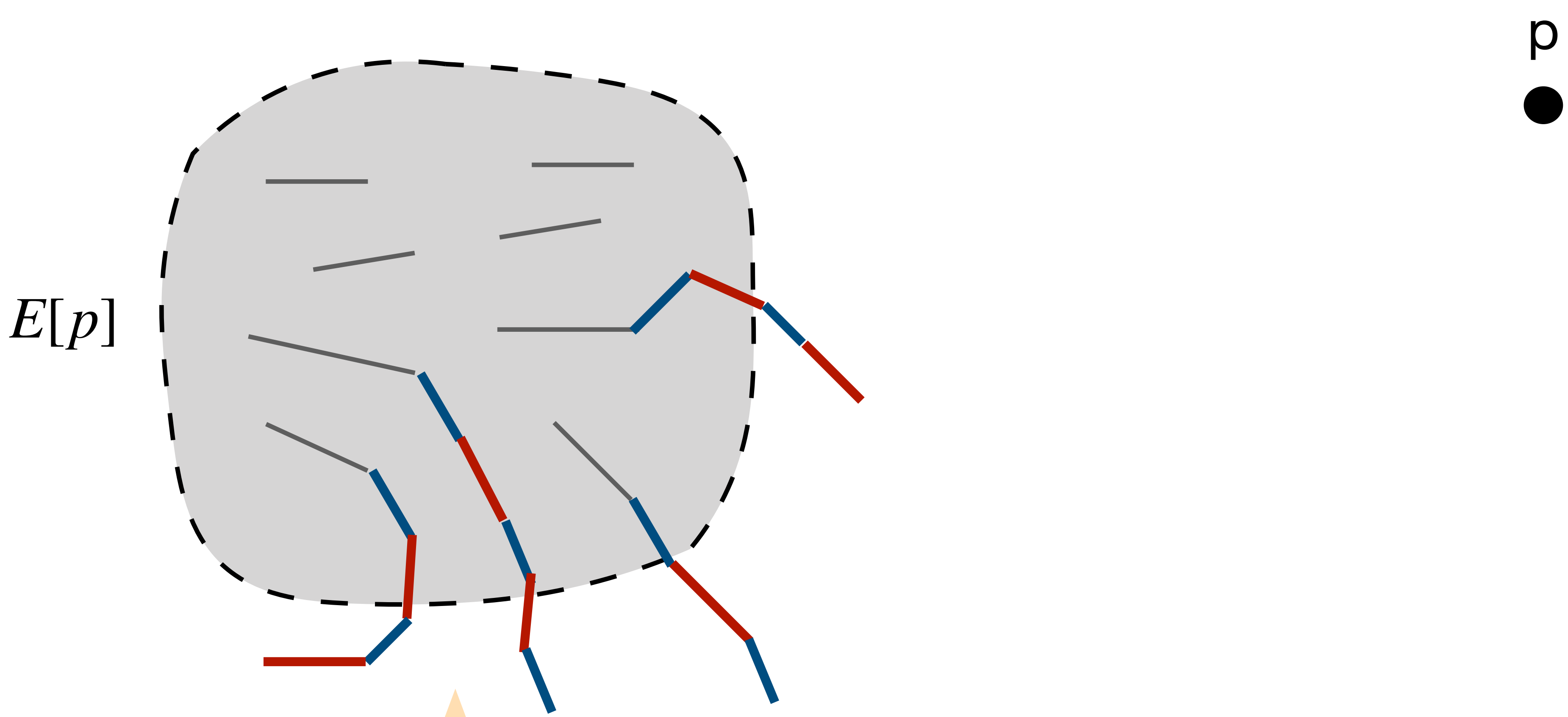


Assume with prob. $\mu[p]$, uncolored edge is **uniformly distributed** among $E[p]$

Model the algorithm as a tree $\mathscr{T}$

Each node $p \in \mathscr{T}$ is associated with two fields:

- a probability mass $\mu[p] \in [0,1]$
- a set $E[p]$ of vertex-disjoint edges

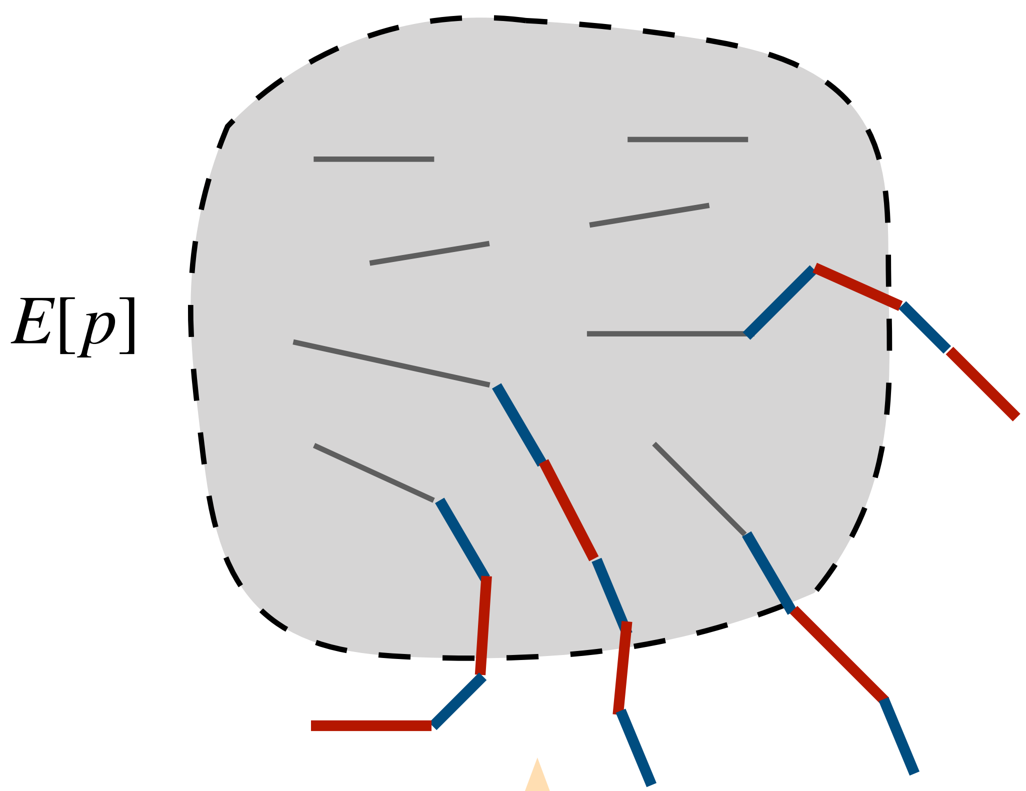


Assume with prob. $\mu[p]$, uncolored edge is **uniformly distributed** among $E[p]$

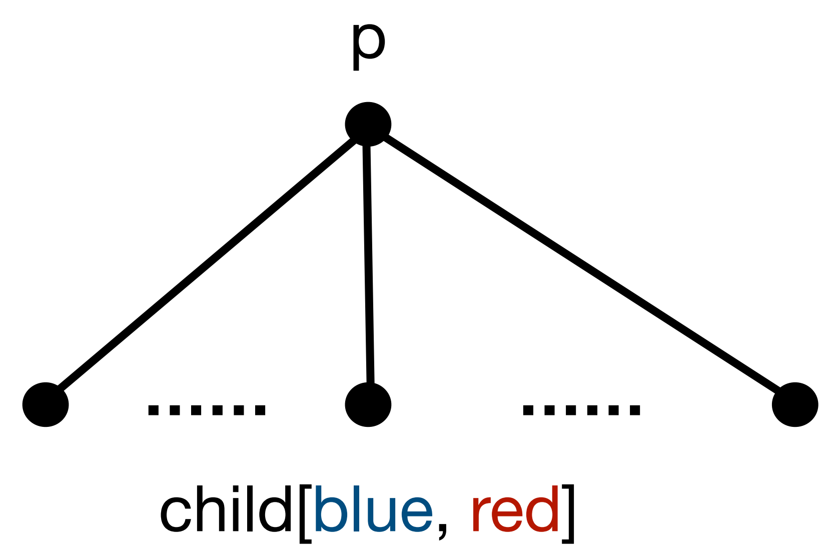
Model the algorithm as a tree $\mathscr{T}$

Each node $p \in \mathscr{T}$ is associated with two fields:

- a probability mass $\mu[p] \in [0,1]$
- a set $E[p]$ of vertex-disjoint edges



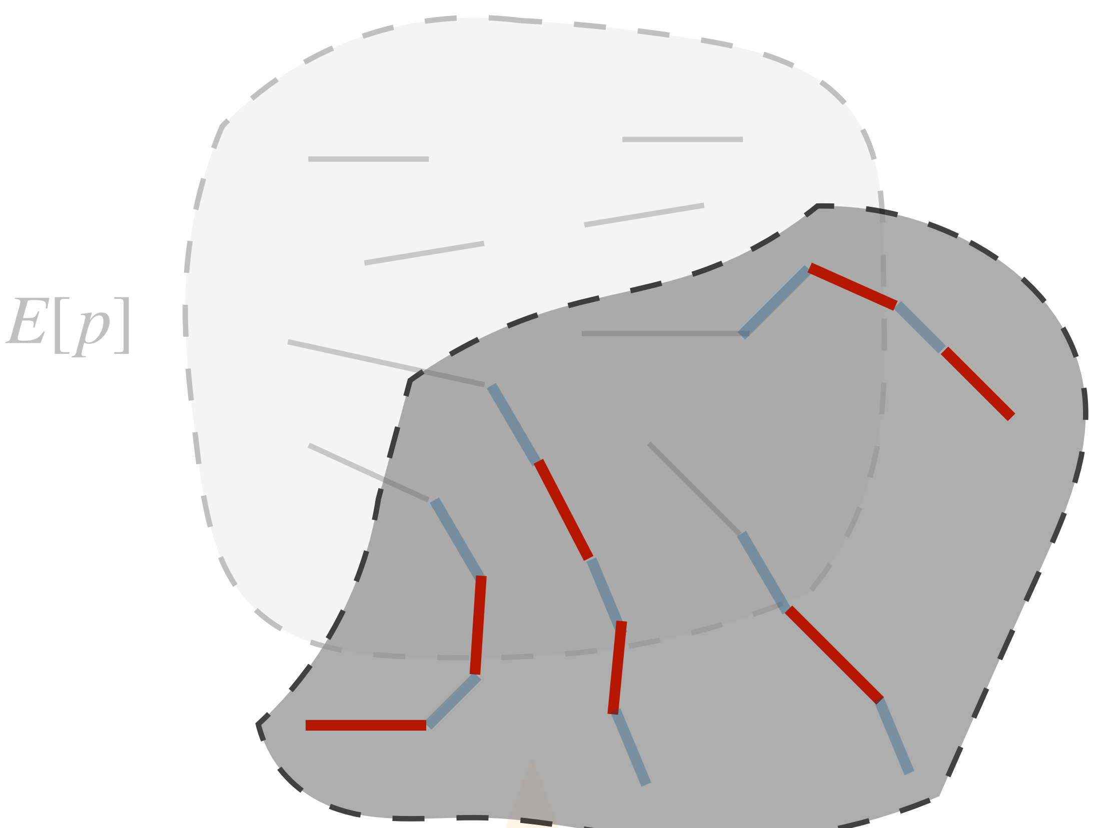
Assume with prob. $\mu[p]$, uncolored edge is **uniformly distributed** among $E[p]$

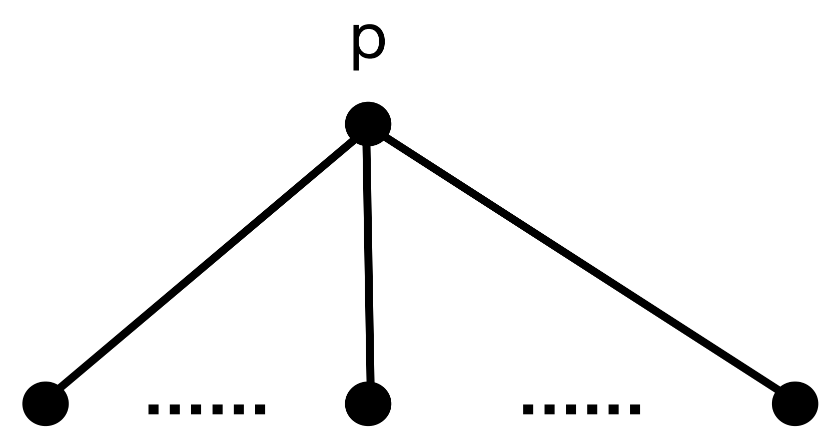


Model the algorithm as a tree $\mathscr{T}$

Each node $p \in \mathscr{T}$ is associated with two fields:

- a probability mass $\mu[p] \in [0,1]$
- a set $E[p]$ of vertex-disjoint edges





Assume with prob. $\mu[p]$ uncolored edge is **uniformly distributed** among $E[p]$

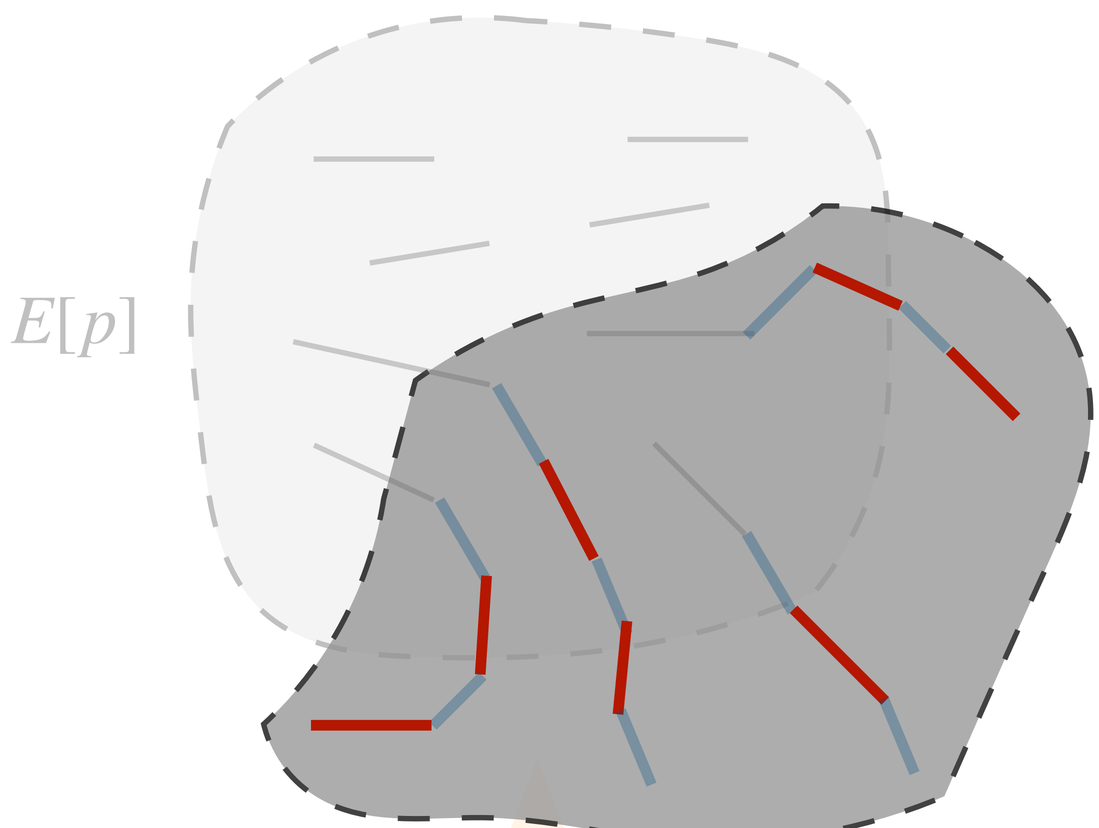
E[child[blue, red]] = **red** edges

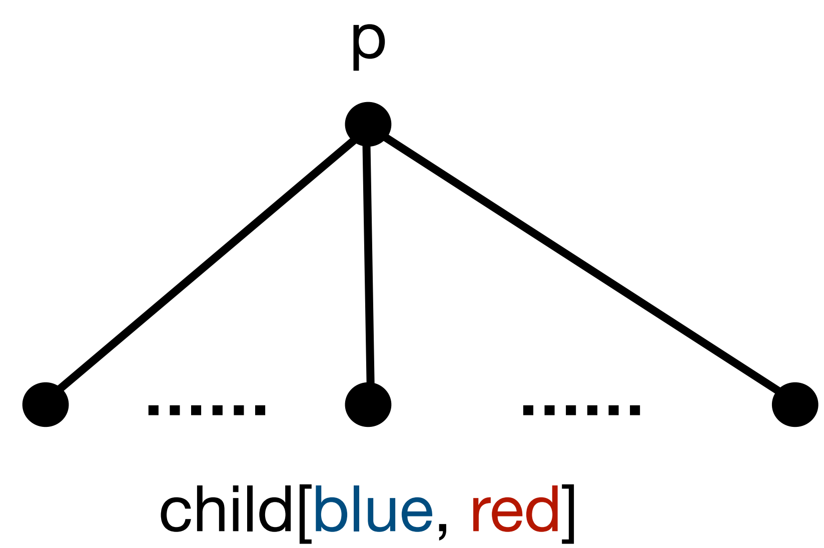
$$\mu[\text{child[blue, red]}] = \mu[p] \cdot \frac{\text{\#alt-blue-red}}{|E[p]|}$$

Model the algorithm as a tree $\mathscr{T}$

Each node $p \in \mathscr{T}$ is associated with two fields:

- a probability mass $\mu[p] \in [0,1]$
- a set $E[p]$ of vertex-disjoint edges





E[child[blue, red]] = **red** edges

$$\mu[\text{child[blue, red]}] = \mu[p] \cdot \frac{\text{\#alt-blue-red}}{|E[p]|}$$

If #blue-red alt-paths is larger than

$$\epsilon^2/\log^3 n \cdot |E[p]|$$

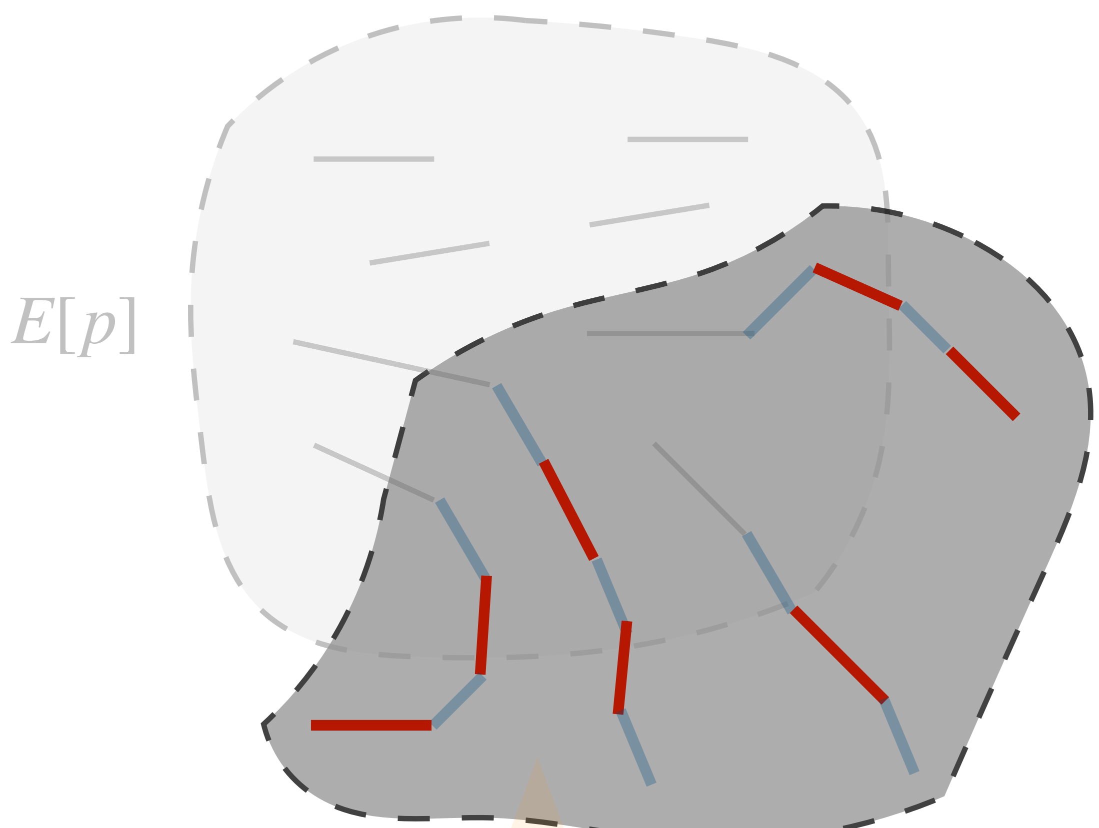
then,

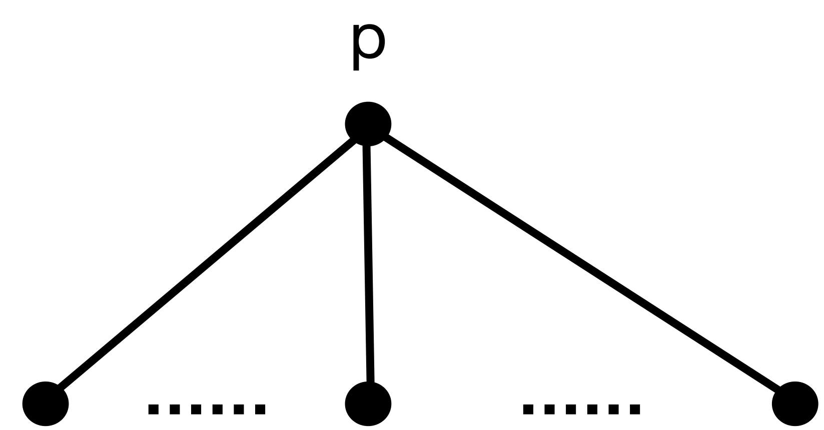
$$|E[\text{child[blue, red]}]| \geq h \cdot \epsilon^2/2\log^3 n \cdot |E[p]|$$

Model the algorithm as a tree $\mathscr{T}$

Each node $p \in \mathscr{T}$ is associated with two fields:

- a probability mass $\mu[p] \in [0,1]$
- a set $E[p]$ of vertex-disjoint edges





E[child[blue, red]] = **red** edges

$\mu[\text{child[blue, red]}] = \mu[p] \cdot \dfrac{\text{\#alt-blue-red}}{|E[p]|}$

If #blue-red alt-paths is smaller than

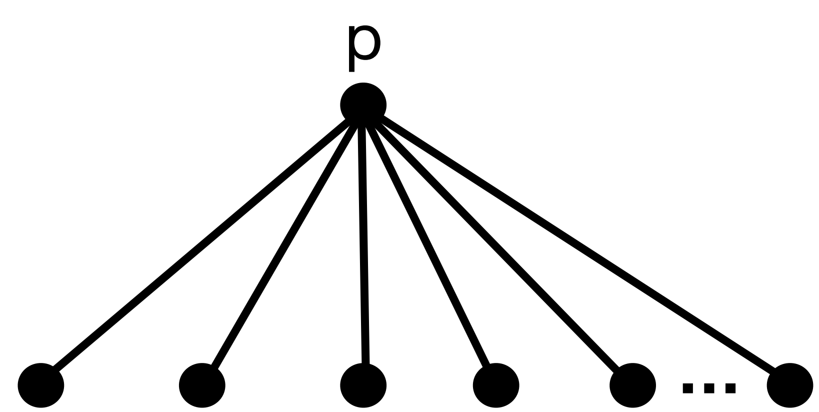
$$\epsilon^2/\log^3 n \cdot |E[p]|$$

then,

$$\sum \mu\,[\text{child[blue, red]}] \le \mu[p]/\log n$$

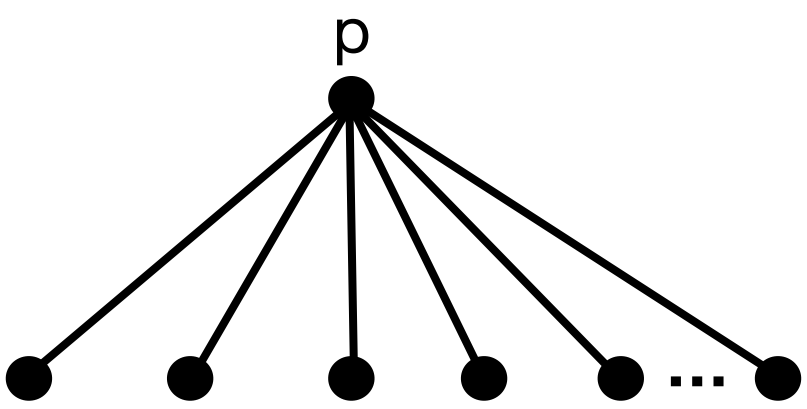
Refining the update time analysis

For simplicity, assume Vizing's algorithm never finds an alternating path shorter than *h* **before the last iteration**



Refining the update time analysis

For simplicity, assume Vizing's algorithm never finds an alternating path shorter than *h* **before the last iteration**



If #alt-paths of a certain type is larger than

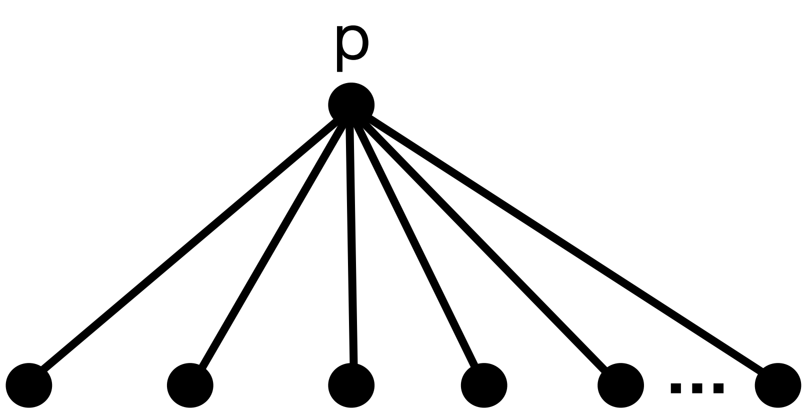
$$\epsilon^2/\log^3 n \cdot |E[p]|$$

then,

$$|E[\text{child[this type]}]|$$
$$\geq h \cdot \epsilon^2/2\log^3 n \cdot |E[p]|$$

Refining the update time analysis

For simplicity, assume Vizing's algorithm never finds an alternating path shorter than h **before the last iteration**



If #alt-paths of a certain type is larger than

$$\epsilon^2/\log^3 n \cdot |E[p]|$$

then,

$$|E[\text{child[this type]}]| \geq h \cdot \epsilon^2/2\log^3 n \cdot |E[p]|$$

If #alt-paths of a certain type is smaller than

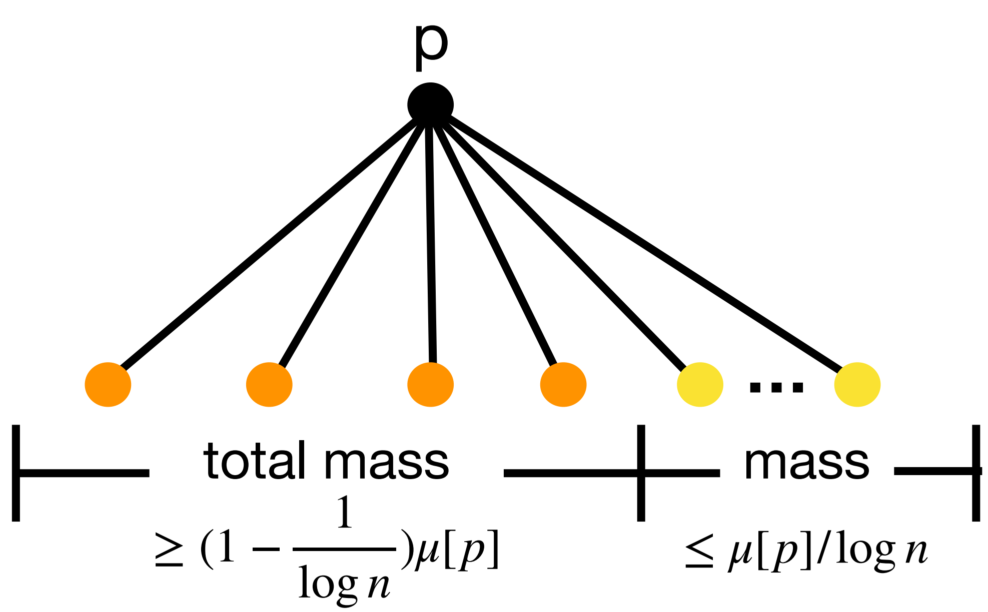
$$\epsilon^2/\log^3 n \cdot |E[p]|$$

then,

$$\sum \mu[\text{child[this type]}] \leq \mu[p]/\log n$$

Refining the update time analysis

For simplicity, assume Vizing's algorithm never finds an alternating path shorter than h **before the last iteration**



If #alt-paths of a certain type is larger than

$$\epsilon^2/\log^3 n \cdot |E[p]|$$

then,

$$|E[\text{child[this type]}]| \geq h \cdot \epsilon^2/2\log^3 n \cdot |E[p]|$$

If #alt-paths of a certain type is smaller than

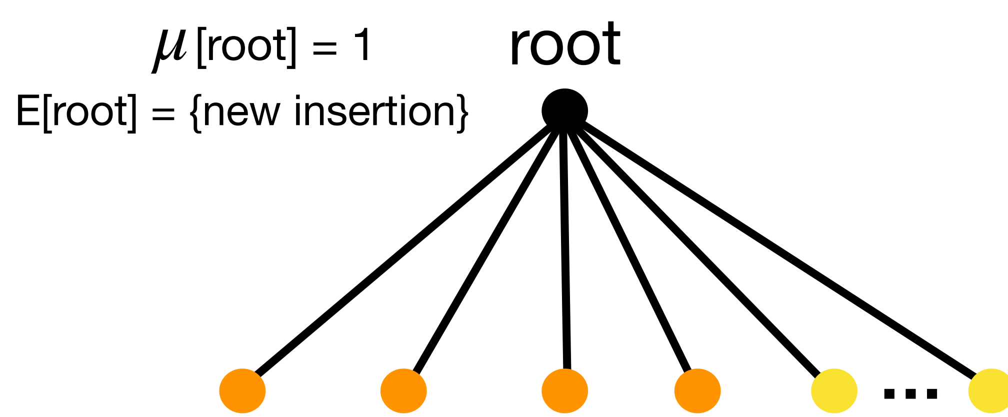
$$\epsilon^2/\log^3 n \cdot |E[p]|$$

then,

$$\sum \mu[\text{child[this type]}] \leq \mu[p]/\log n$$

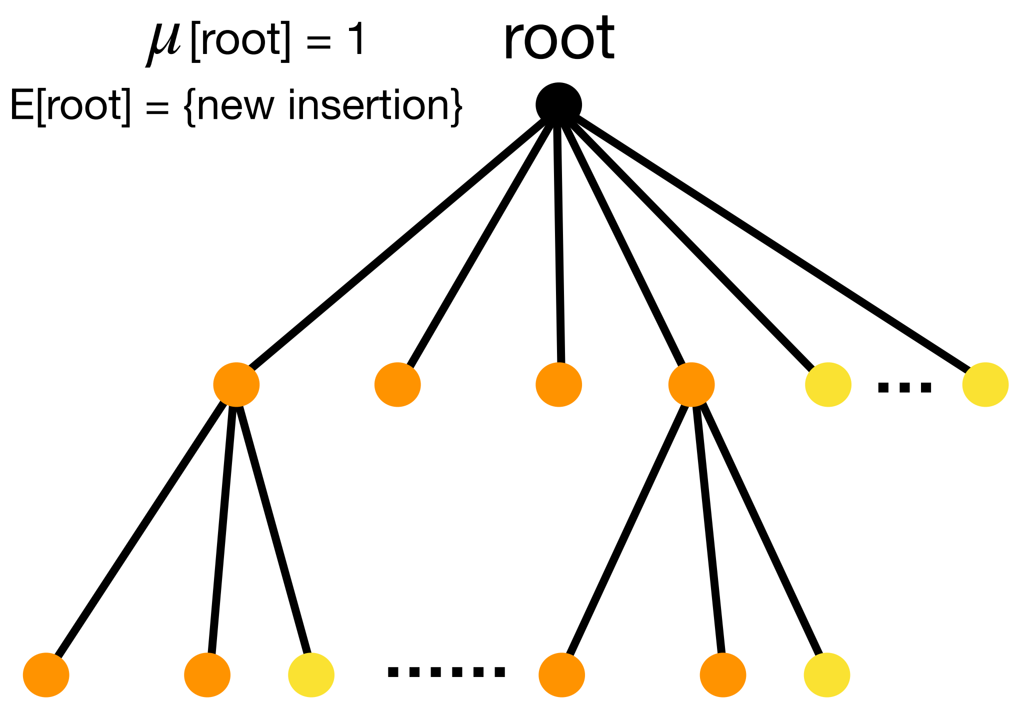
Refining the update time analysis

For simplicity, assume Vizing's algorithm never finds an alternating path shorter than *h* **before the last iteration**



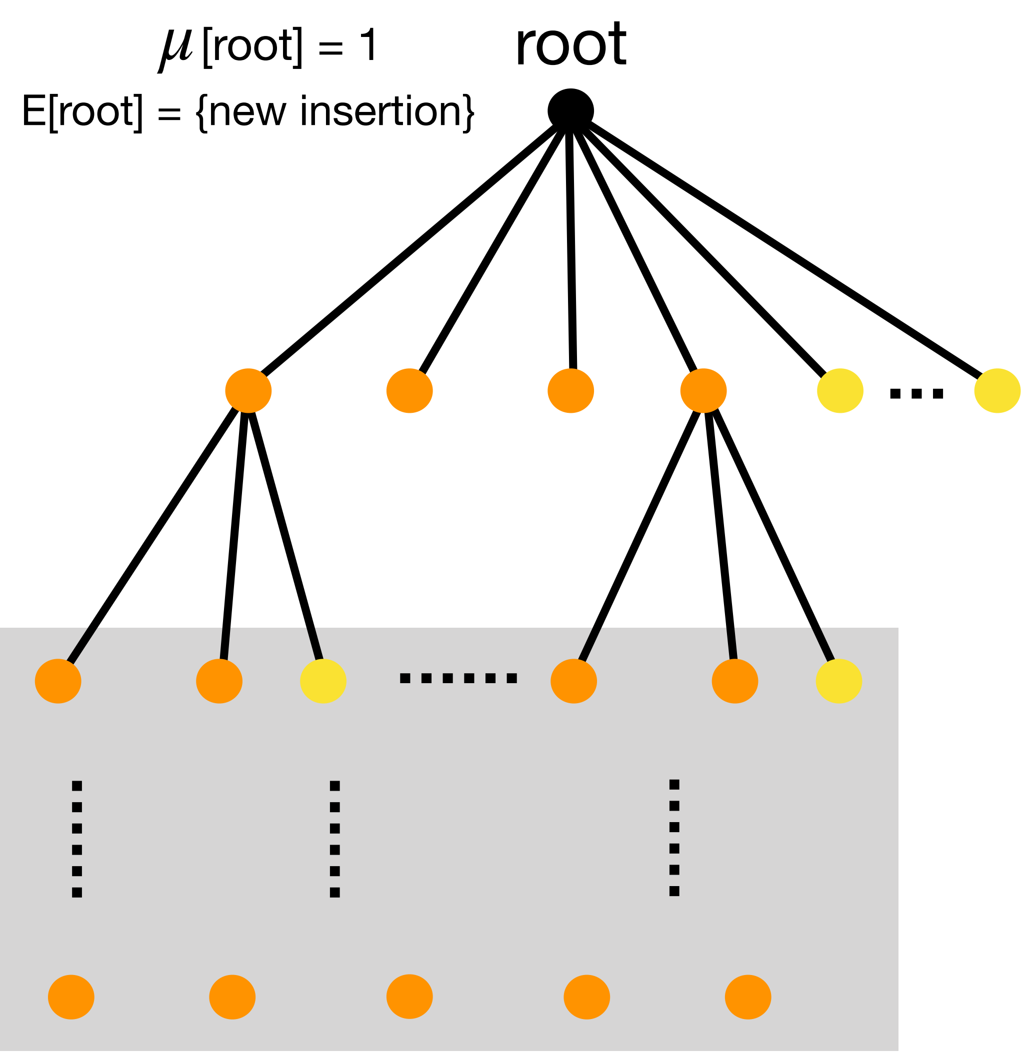
Refining the update time analysis

For simplicity, assume Vizing's algorithm never finds an alternating path shorter than *h* **before the last iteration**



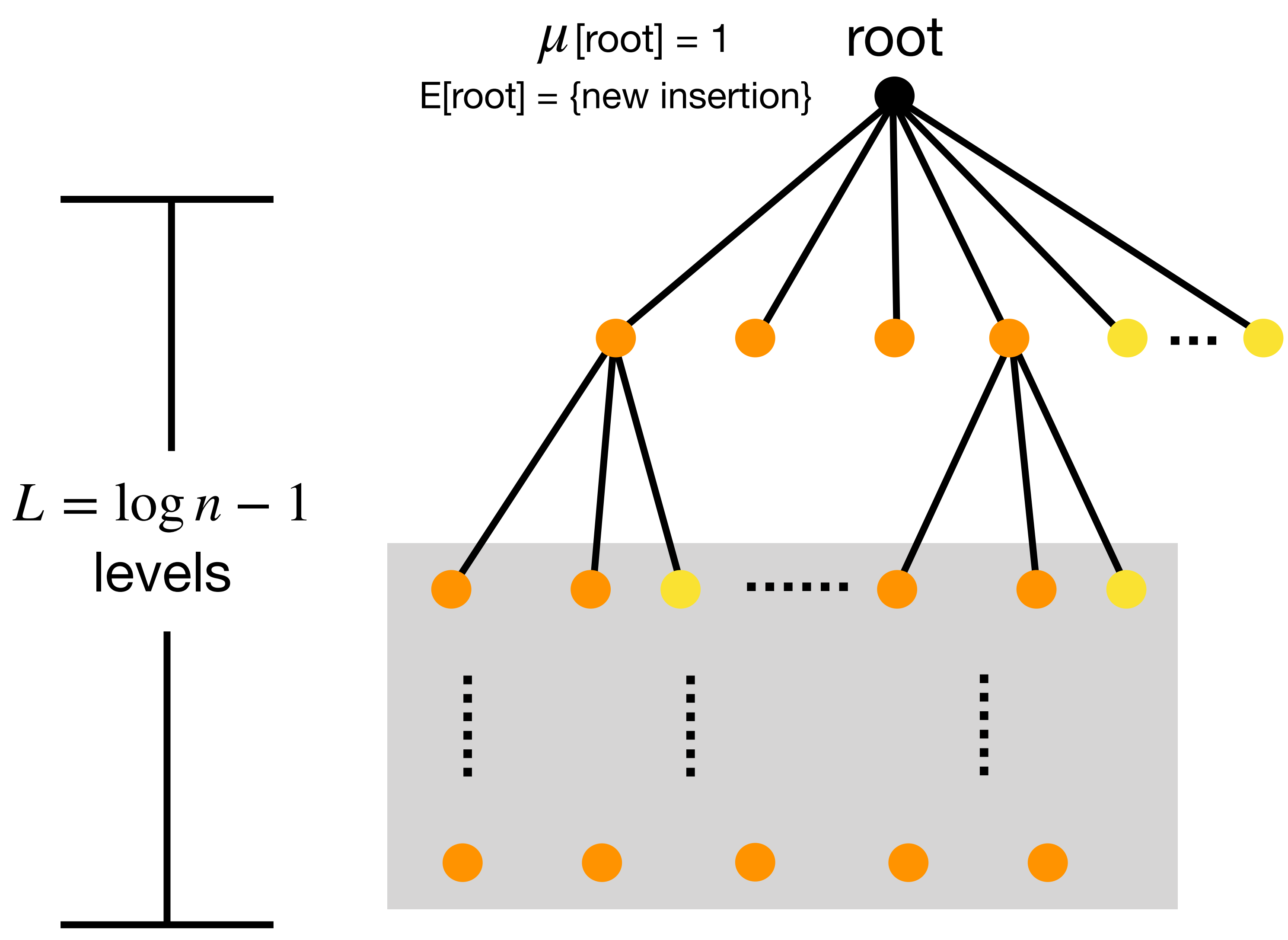
Refining the update time analysis

For simplicity, assume Vizing's algorithm never finds an alternating path shorter than *h* **before the last iteration**



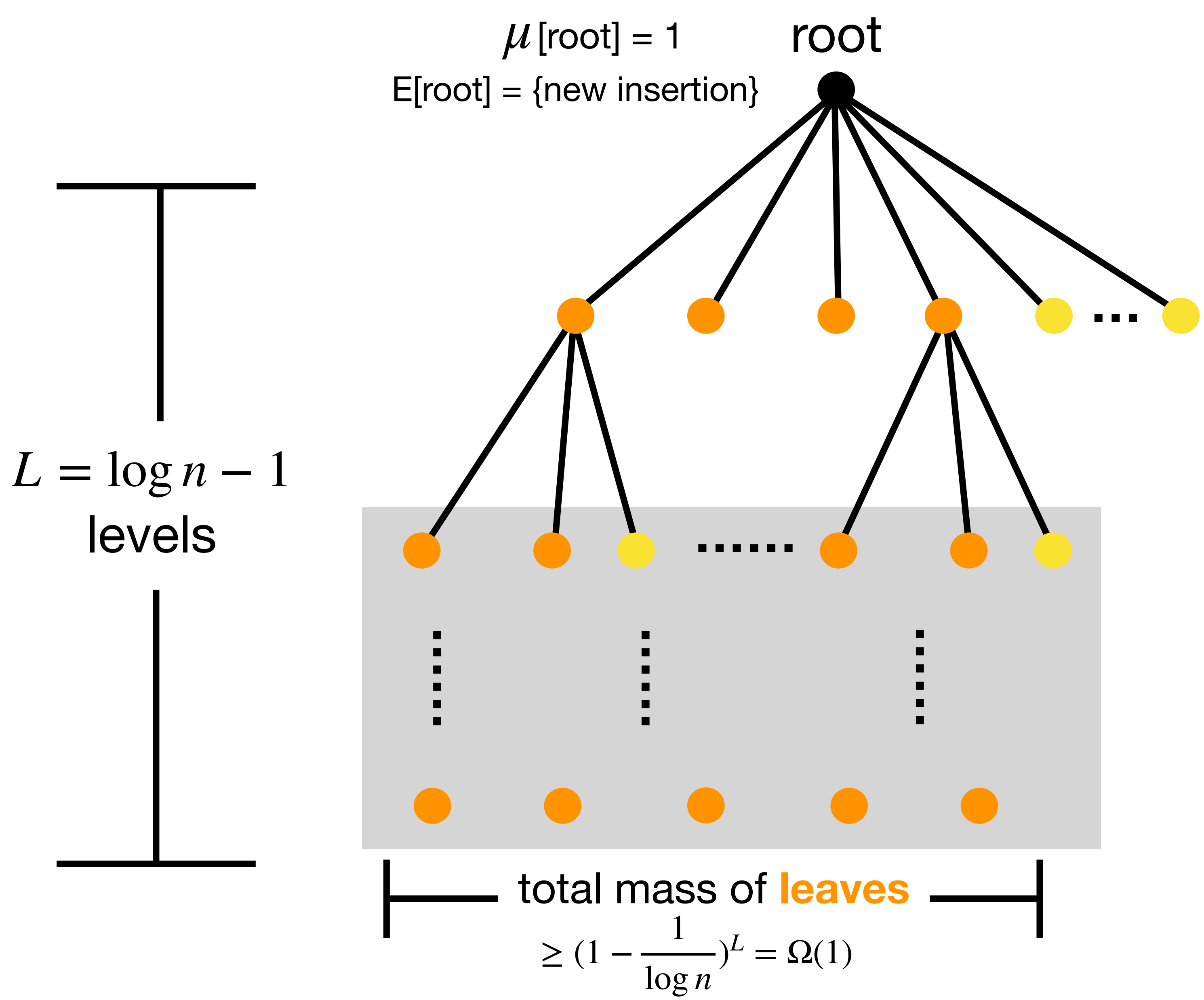
Refining the update time analysis

For simplicity, assume Vizing's algorithm never finds an alternating path shorter than *h* **before the last iteration**



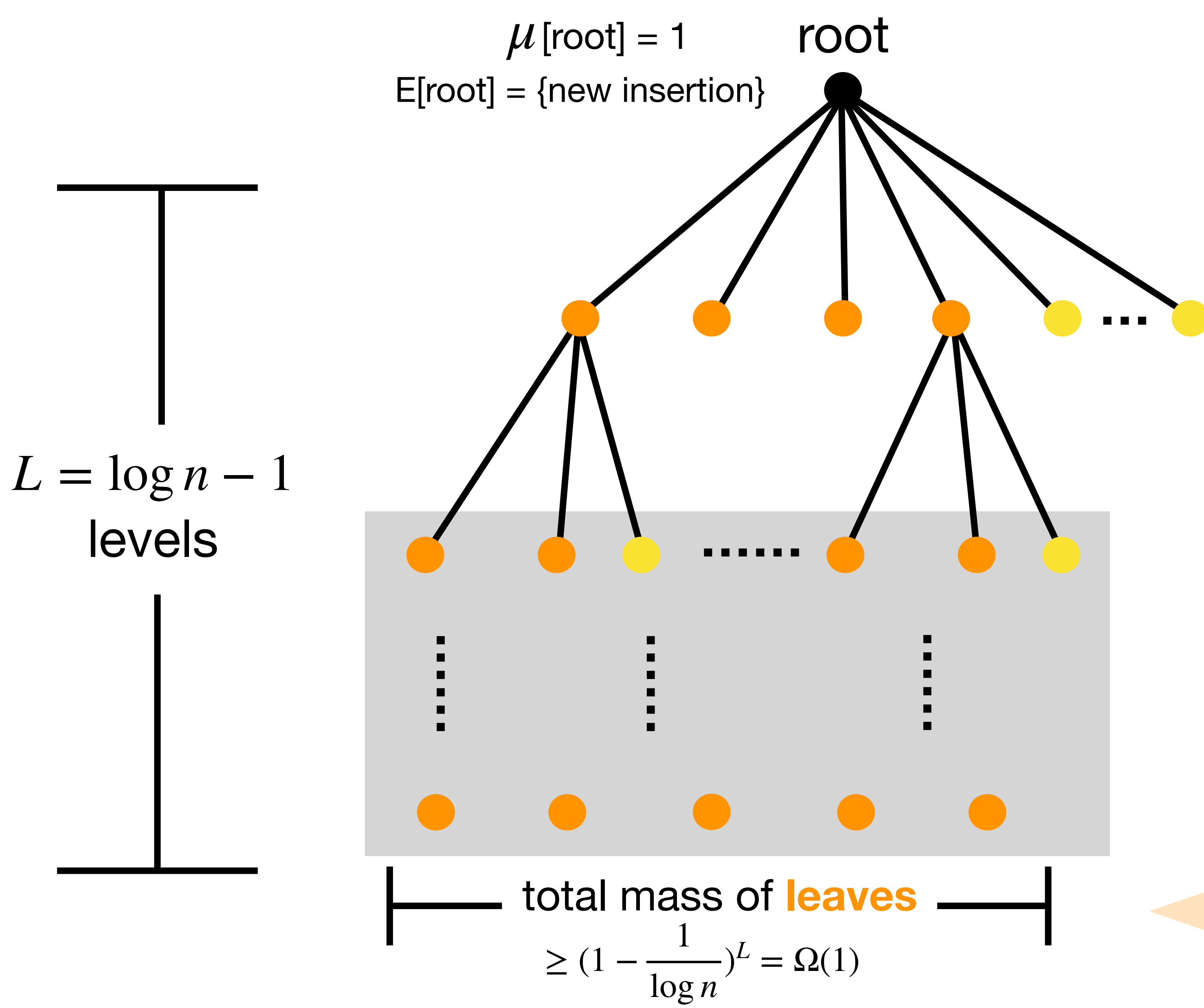
Refining the update time analysis

For simplicity, assume Vizing's algorithm never finds an alternating path shorter than *h* **before the last iteration**



Refining the update time analysis

For simplicity, assume Vizing's algorithm never finds an alternating path shorter than *h* **before the last iteration**



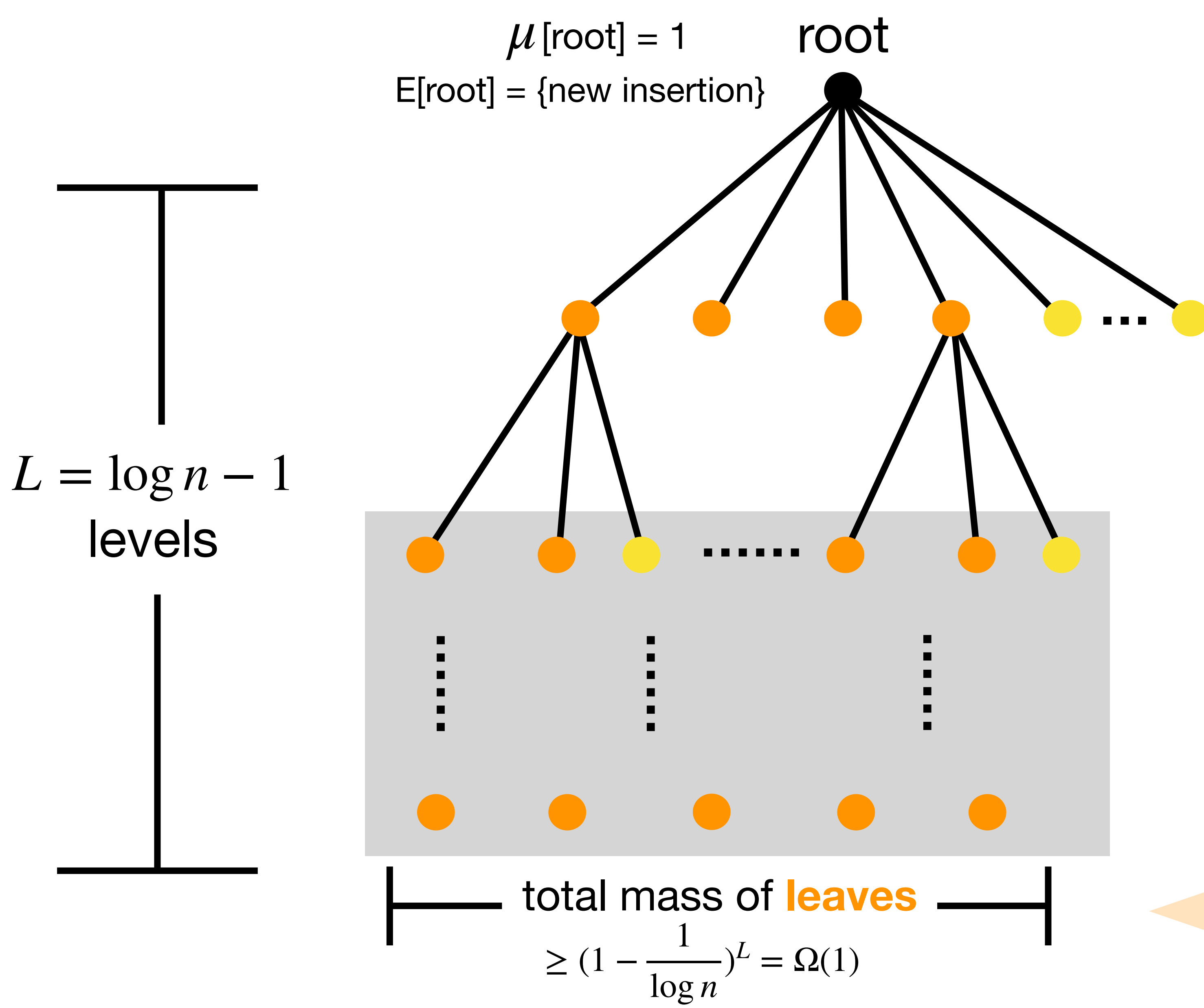
Setting $h \leftarrow 2\log^3 n/\epsilon^2$

Then $|E[\text{any leaf}]|$

$\geq \left(h \cdot \epsilon^2/\log^3 n\right)^L \geq n/2$

Refining the update time analysis

For simplicity, assume Vizing's algorithm never finds an alternating path shorter than h **before the last iteration**



In the end, with **constant** prob., the uncolored edge is **uniformly** distributed among **n/2 vertex-disjoint** edges

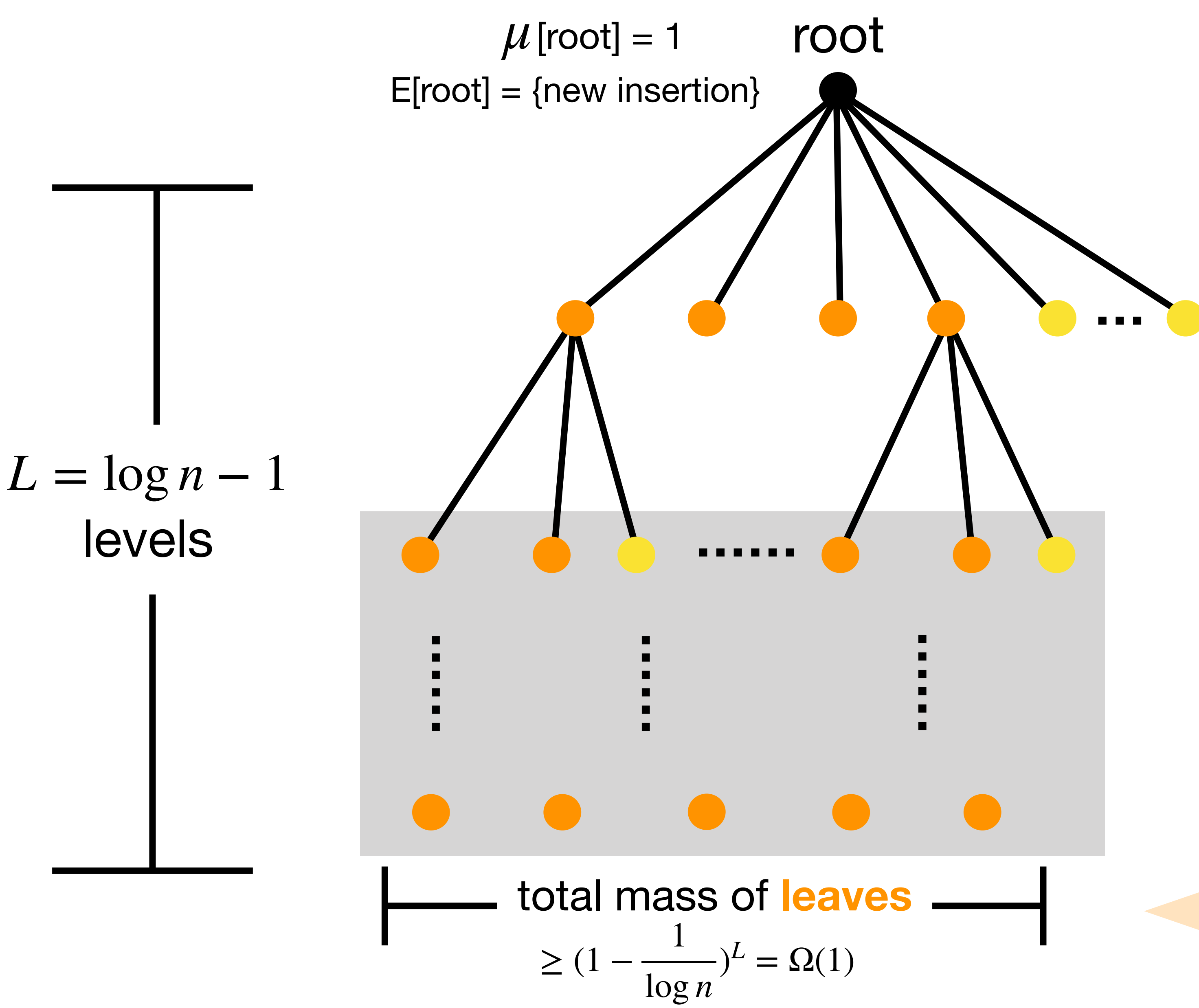
Setting $h \leftarrow 2\log^3 n/\epsilon^2$

Then $|E[\text{any leaf}]|$

$\geq \left(h \cdot \epsilon^2/\log^3 n\right)^L \geq n/2$

Refining the update time analysis

For simplicity, assume Vizing's algorithm never finds an alternating path shorter than *h* **before the last iteration**



Consequently, in the last iteration, most alt-paths have **poly-log** length

In the end, with **constant** prob., the uncolored edge is **uniformly** distributed among **n/2 vertex-disjoint** edges

Setting $h \leftarrow 2\log^3 n/\epsilon^2$

Then $|E[\text{any leaf}]|$

$\geq \left(h \cdot \epsilon^2/\log^3 n\right)^L \geq n/2$

Thank you!