Dynamic Graph Coloring

Luis Barba [1] Jean Cardinal [2] Matias Korman [3]
Stefan Langerman [2] André van Renssen [4,5]
Marcel Roeloffzen [4,5] Sander Verdonschot [6]

[1]ETH Zürich

[2]Université Libre de Bruxelles

[3]Tohoku University

[4]National Institute of Informatics

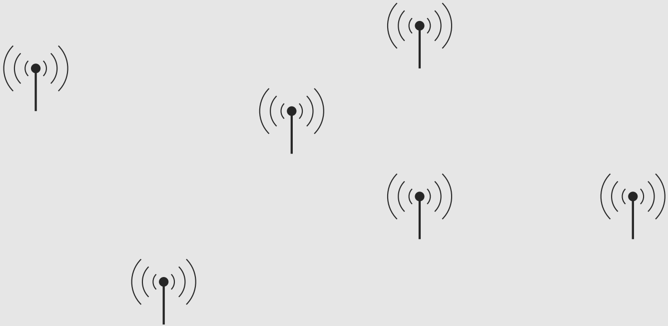
[5]JST, ERATO, Kawarabayashi Large Graph Project

[6]Carleton University

July 31, 2017

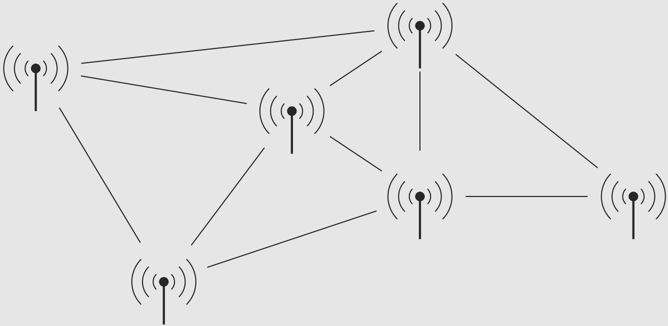
Problem

- Maintain a proper coloring of a changing graph



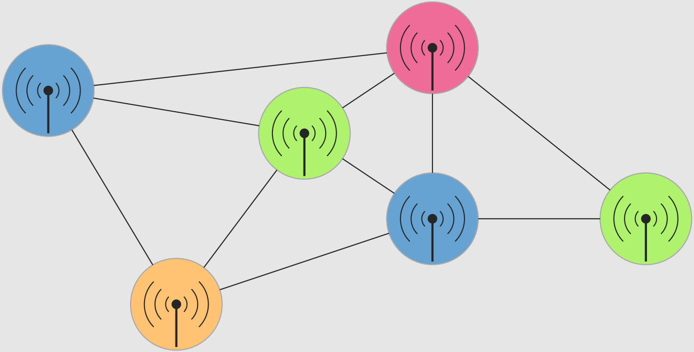
Problem

- Maintain a proper coloring of a changing graph



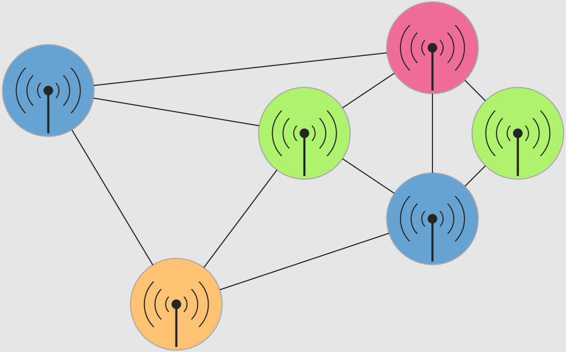
Problem

- Maintain a proper coloring of a changing graph



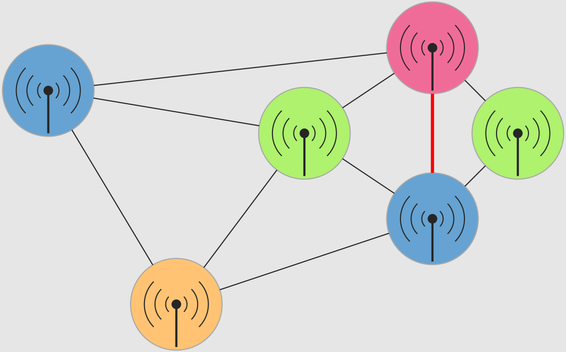
Problem

- Maintain a proper coloring of a changing graph



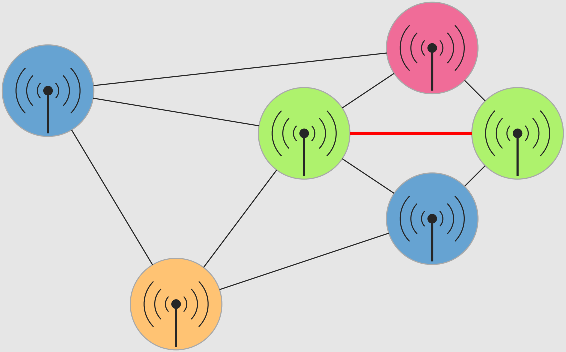
Problem

- Maintain a proper coloring of a changing graph



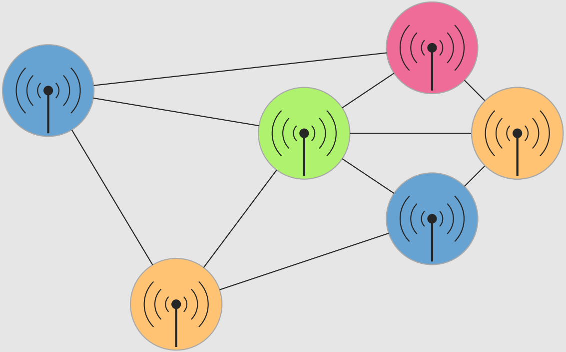
Problem

- Maintain a proper coloring of a changing graph



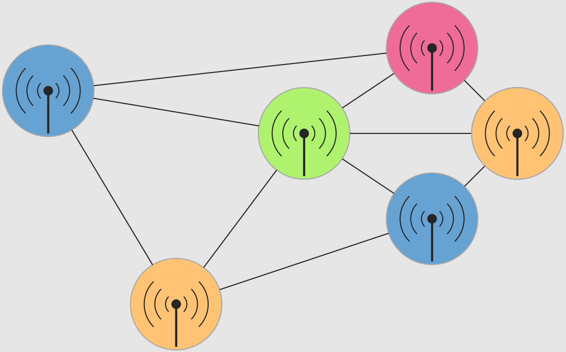
Problem

- Maintain a proper coloring of a changing graph



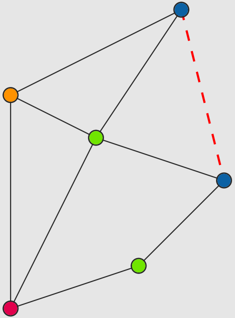
Problem

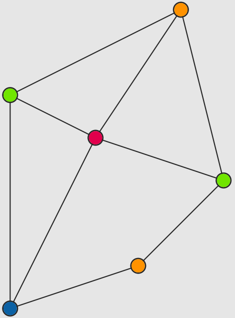
- Maintain a proper coloring of a changing graph:
  - Add & remove edges
  - Add & remove vertices with incident edges



Problem

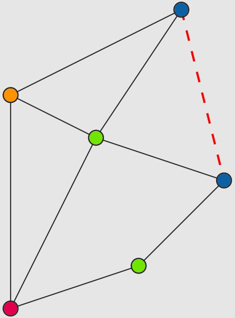
- Easy! Recompute the coloring for every change

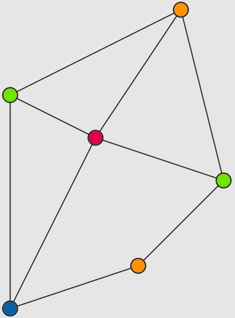




Problem

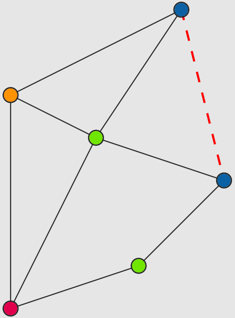
- Easy! Recompute the coloring for every change
  $\Rightarrow$ Limit the number of vertex color changes

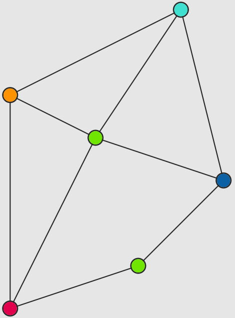




Problem

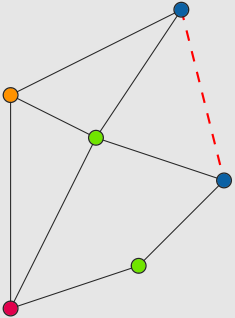
- Easy! Recompute the coloring for every change
  $\Rightarrow$ Limit the number of vertex color changes
- Easy! Use a new color for every change

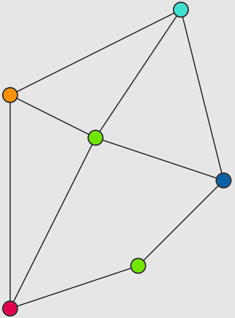




Problem

- Easy! Recompute the coloring for every change
  $\Rightarrow$ Limit the number of vertex color changes
- Easy! Use a new color for every change
  $\Rightarrow$ Limit the number of colors





Problem

- Trade off the number of colors vs vertex color changes
- Optimal coloring $\Rightarrow \Omega(n)$ vertex recolorings per update

Problem

- Trade off the number of colors vs vertex color changes
- Optimal coloring $\Rightarrow \Omega(n)$ vertex recolorings per update
- No vertex recolorings $\Rightarrow \frac{2n}{\log n}$ colors for $\log n$-colorable graph [Halldórsson & Szegedy, 1992]

Problem

- Trade off the number of colors vs vertex color changes
- Optimal coloring $\Rightarrow \Omega(n)$ vertex recolorings per update
- No vertex recolorings $\Rightarrow \frac{2n}{\log n}$ colors for $\log n$-colorable graph [Halldórsson & Szegedy, 1992]

Our results

- $O(d)$-approximate coloring with $O(dn^{(1/d)})$ recolorings
- $O(dn^{(1/d)})$-approximate coloring with $O(d)$ recolorings

Problem

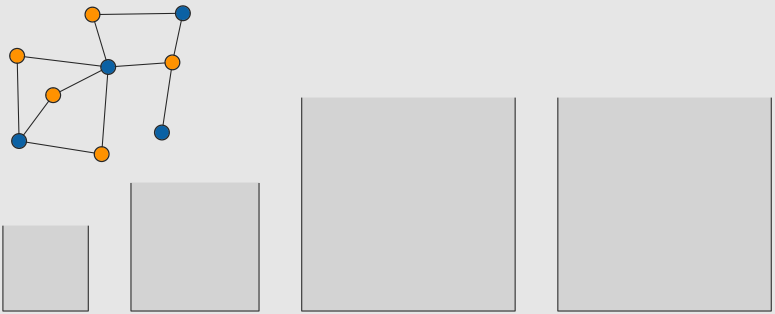
- Trade off the number of colors vs vertex color changes
- Optimal coloring $\Rightarrow \Omega(n)$ vertex recolorings per update
- No vertex recolorings $\Rightarrow \frac{2n}{\log n}$ colors for $\log n$-colorable graph [Halldórsson & Szegedy, 1992]

Our results

- $O(d)$-approximate coloring with $O(dn^{(1/d)})$ recolorings
- $O(dn^{(1/d)})$-approximate coloring with $O(d)$ recolorings
- Maintaining a c-coloring requires $\Omega(n^{\frac{2}{c(c-1)}})$ recolorings

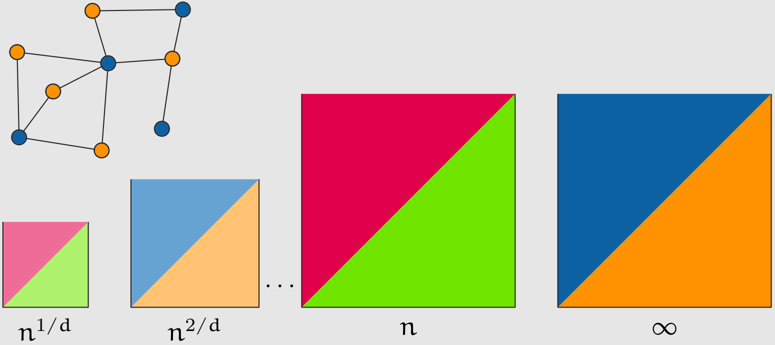
Upper bound: big-buckets

- Vertices are placed in buckets



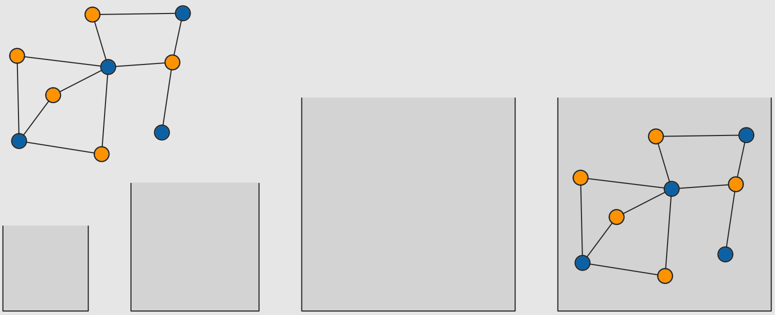
Upper bound: big-buckets

- Each bucket has a fixed size and its own set of colors



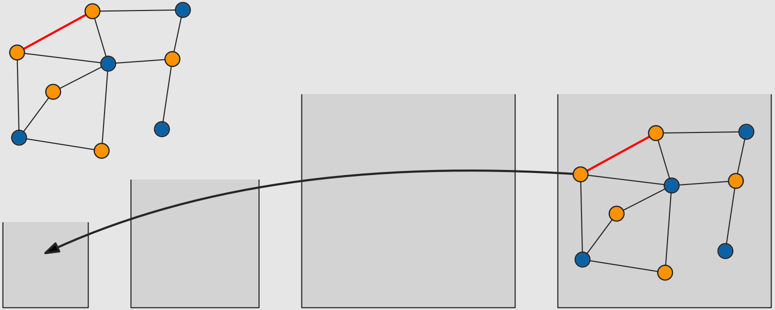
Upper bound: big-buckets

- Initially, all vertices are in the reset bucket



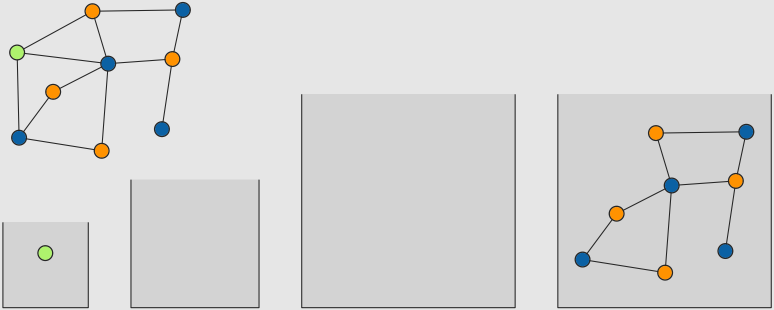
Upper bound: big-buckets

- Changed vertices are placed in the first bucket



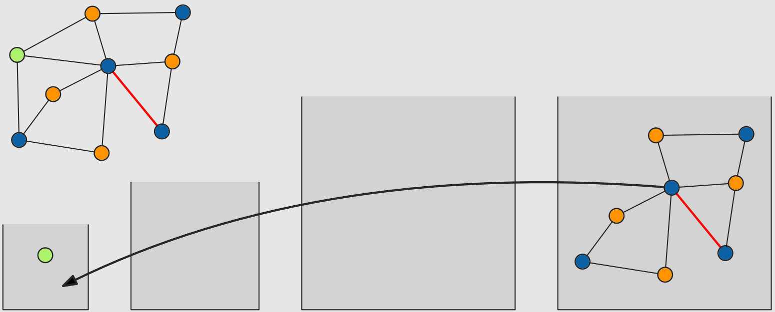
Upper bound: big-buckets

- Changed vertices are placed in the first bucket



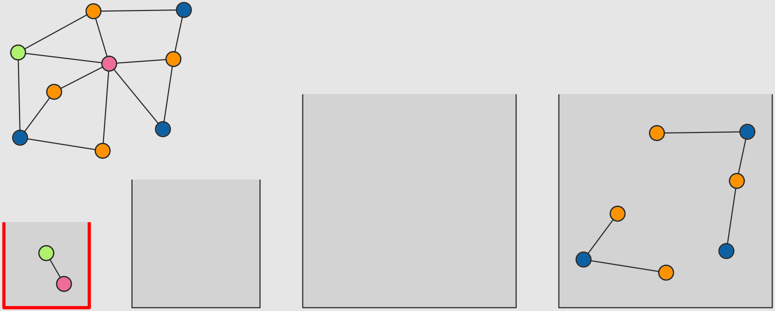
Upper bound: big-buckets

- Changed vertices are placed in the first bucket



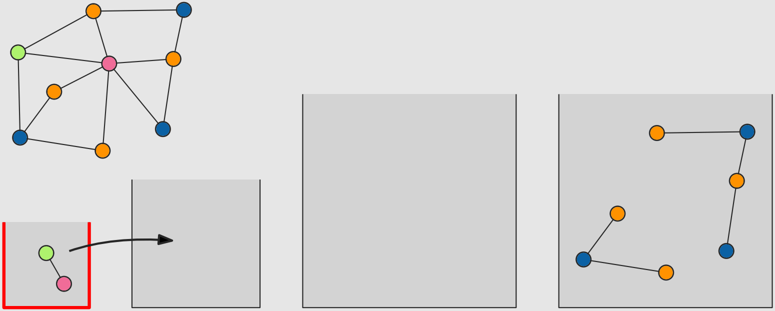
Upper bound: big-buckets

- Changed vertices are placed in the first bucket



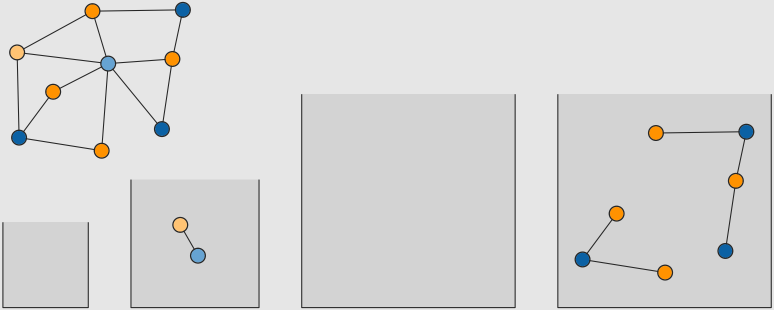
Upper bound: big-buckets

- When a bucket fills up, it is emptied in the next one



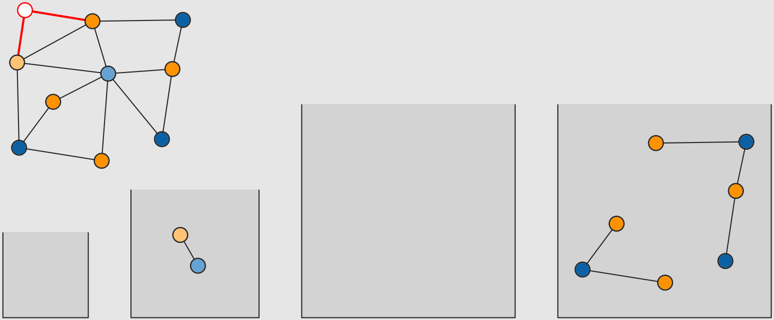
Upper bound: big-buckets

- When a bucket fills up, it is emptied in the next one



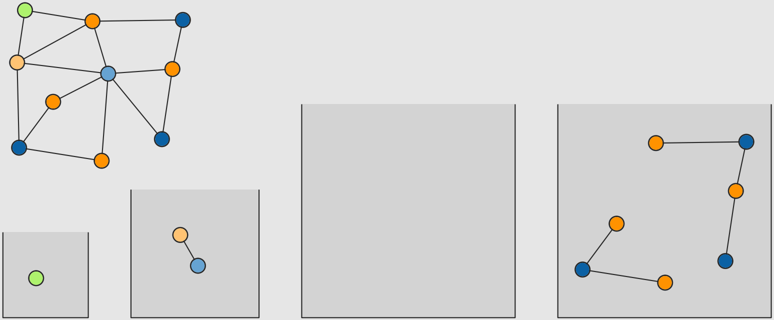
Upper bound: big-buckets

- New vertices also go to the first bucket



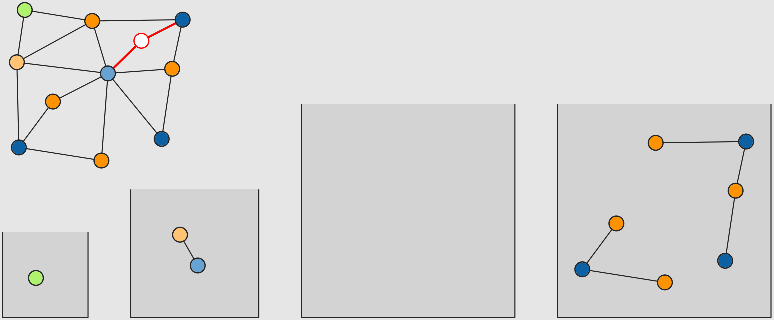
Upper bound: big-buckets

- New vertices also go to the first bucket



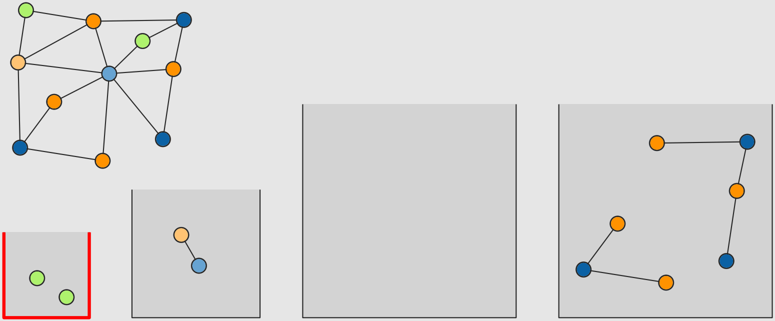
Upper bound: big-buckets

- New vertices also go to the first bucket



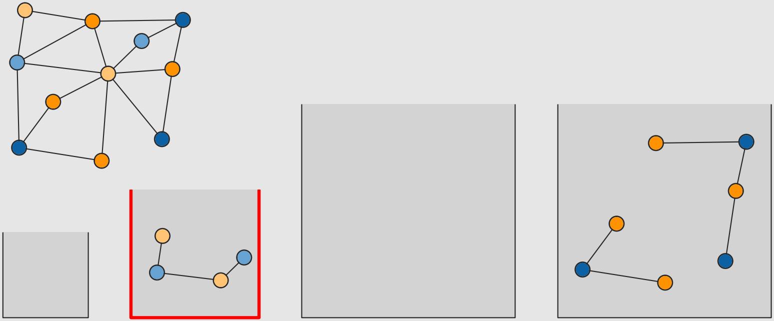
Upper bound: big-buckets

- New vertices also go to the first bucket



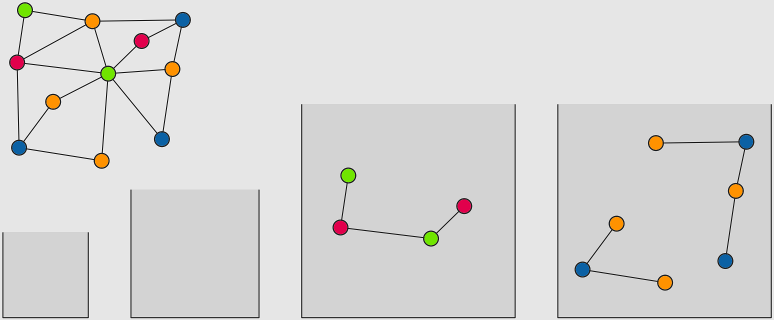
Upper bound: big-buckets

- New vertices also go to the first bucket



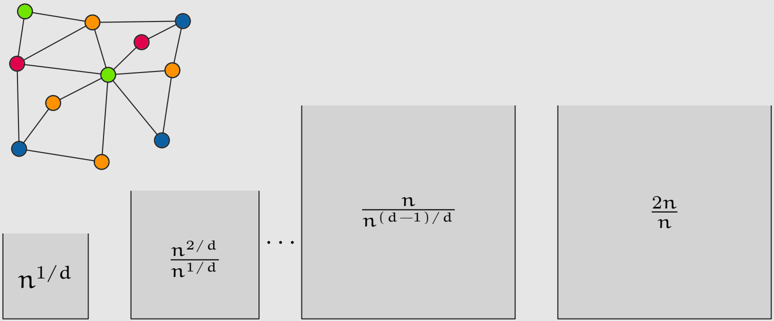
Upper bound: big-buckets

- New vertices also go to the first bucket



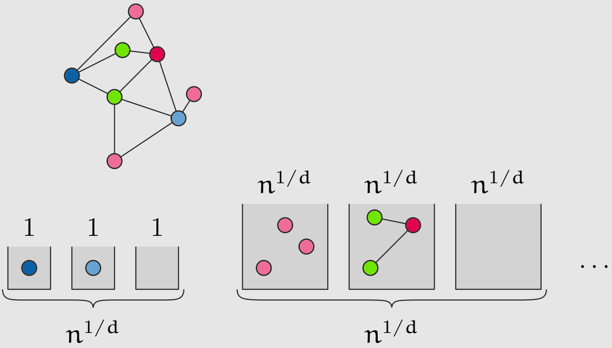
Upper bound: big-buckets

- New vertices also go to the first bucket $(d+1)$-approximate coloring with $O(dn^{1/d})$ recolorings per update



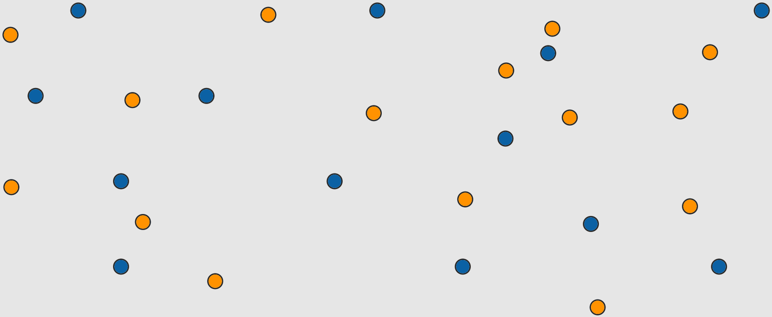
Upper bound: small-buckets

- Split each big bucket into $n^{1/d}$ smaller ones
- $O(dn^{1/d})$-approximate coloring with $d + 2$ recolorings per update



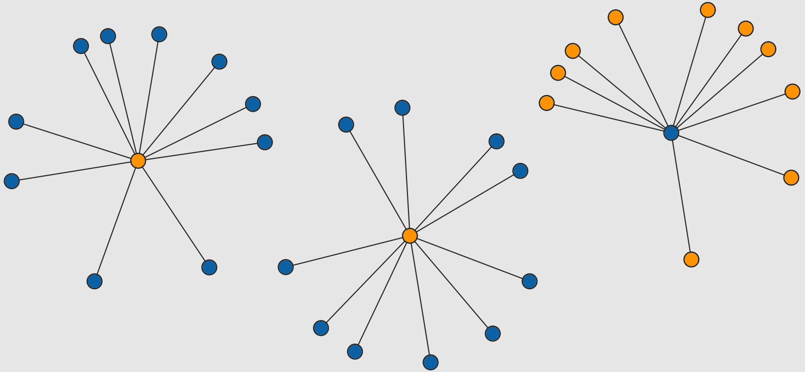
Lower bound

- Warm-up: 2-coloring a forest



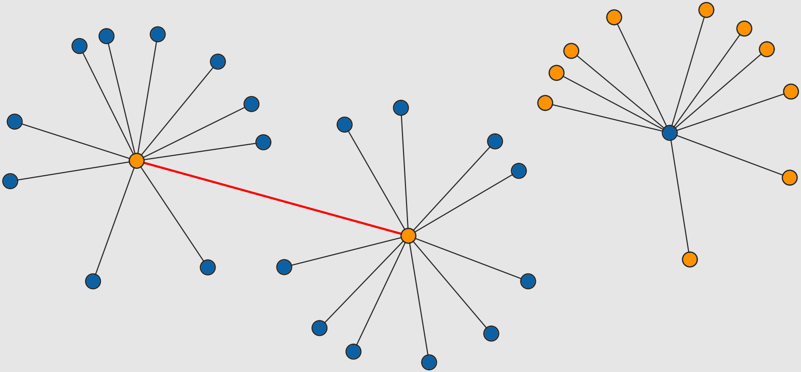
Lower bound

- Build 3 stars of size $n/3$



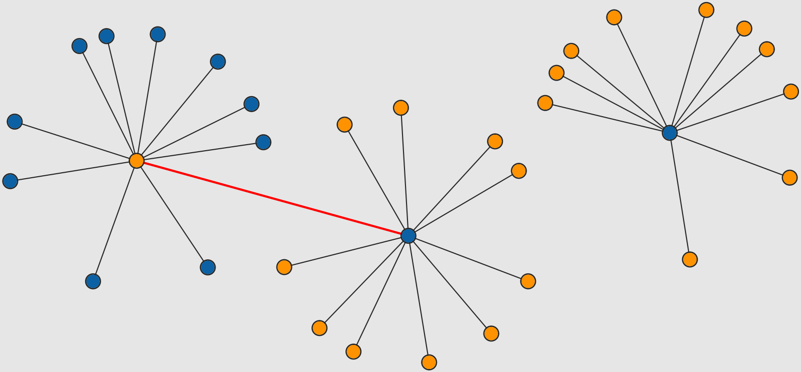
Lower bound

- Connect 2 with the same color root



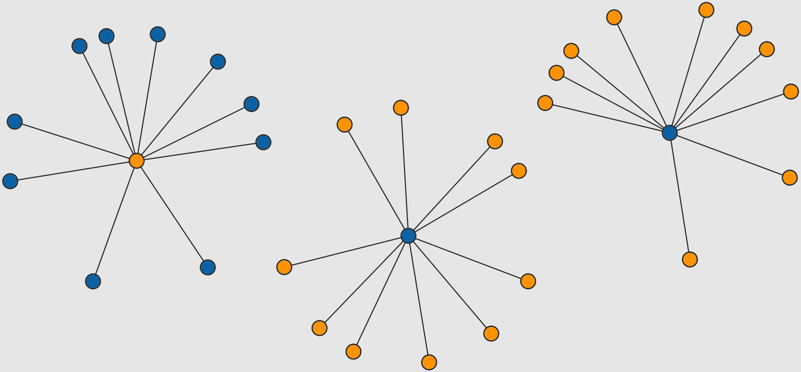
Lower bound

- Connect 2 with the same color root



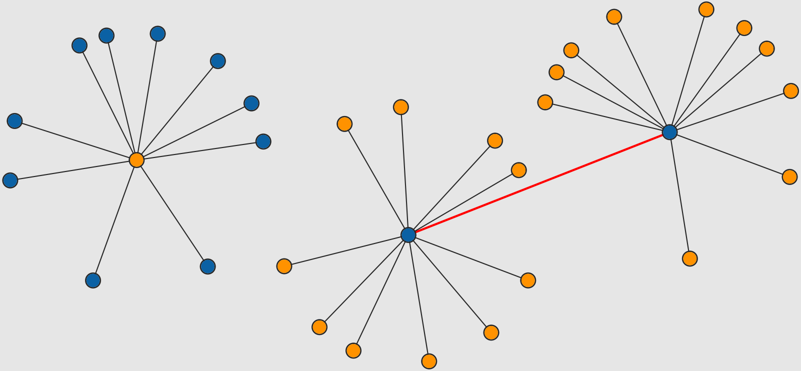
Lower bound

- Connect 2 with the same color root . Repeat



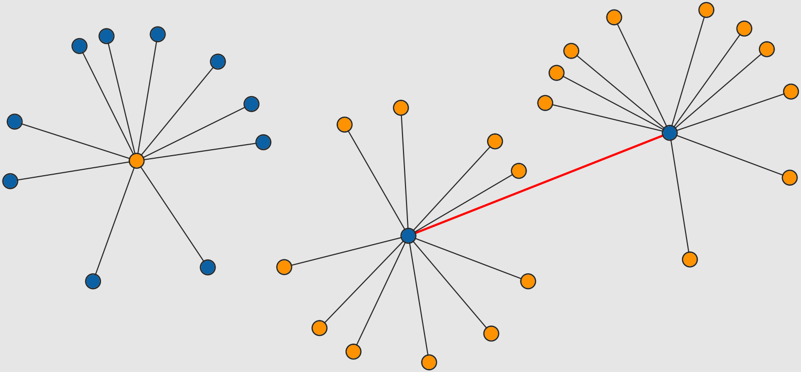
Lower bound

- Connect 2 with the same color root . Repeat



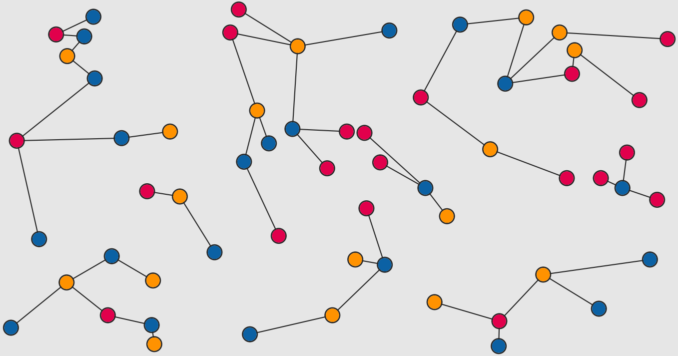
Lower bound

- Connect 2 with the same color root . Repeat
- Maintaining a 2-coloring of a forest requires $\Omega(n)$ recolorings per update



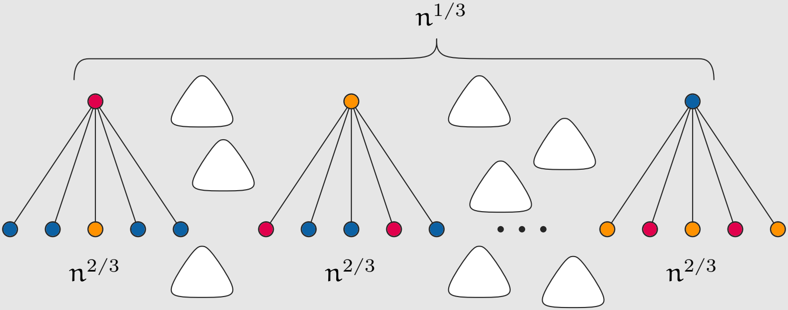
Lower bound

- 3-coloring a forest



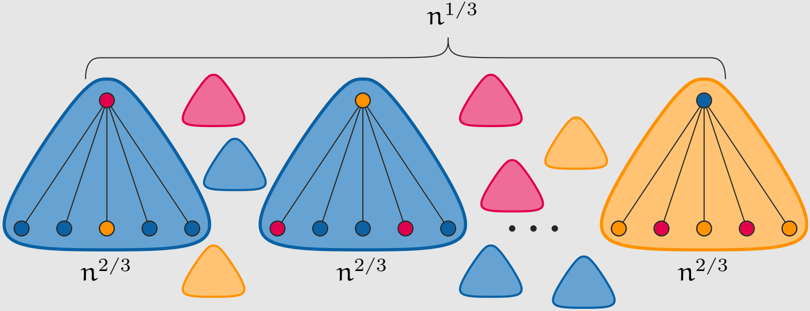
Lower bound

- Build $n^{1/3}$ stars of size $n^{2/3}$



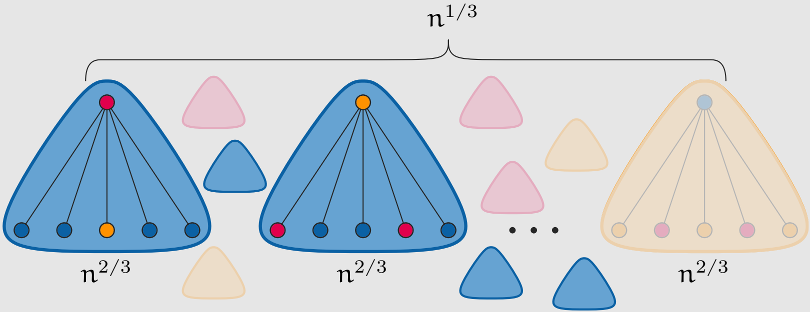
Lower bound

- Assign most common leaf colour to trees



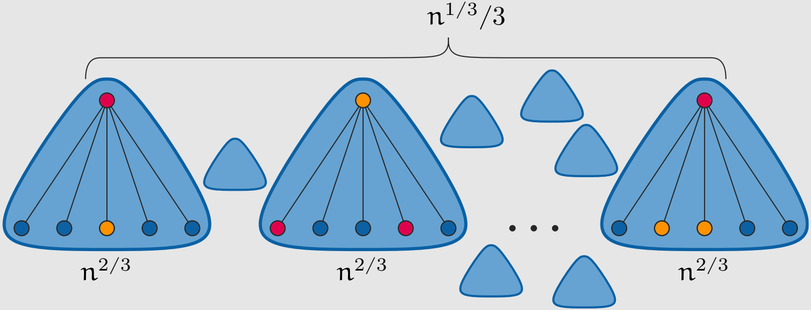
Lower bound

- Keep at least $n^{1/3}/3$ with the same color



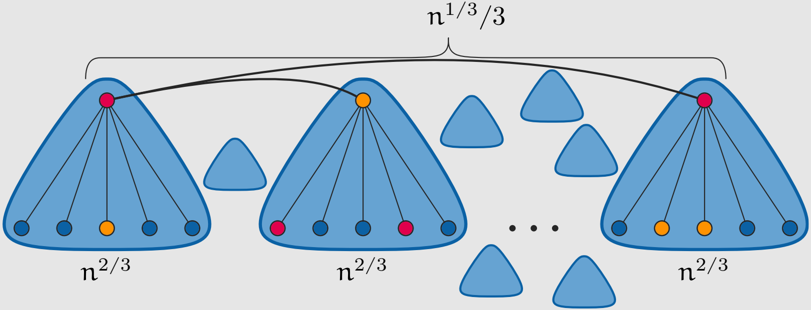
Lower bound

- Keep at least $n^{1/3}/3$ with the same color



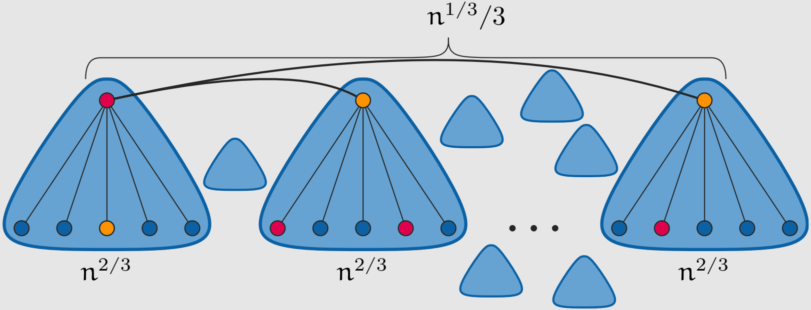
Lower bound

- Group into 3 big trees, each with $n^{1/3}/9$ small trees



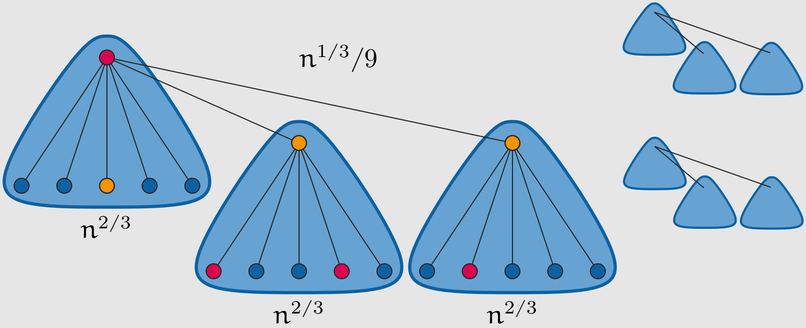
Lower bound

- Group into 3 big trees, each with $n^{1/3}/9$ small trees



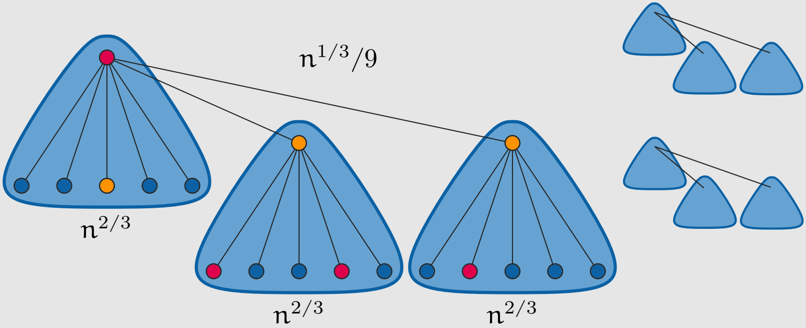
Lower bound

- If at any point, a small tree has no blue children, reset



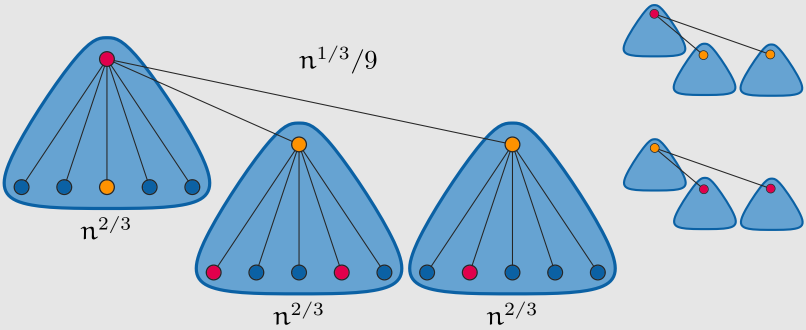
Lower bound

- If at any point, a small tree has no blue children, reset



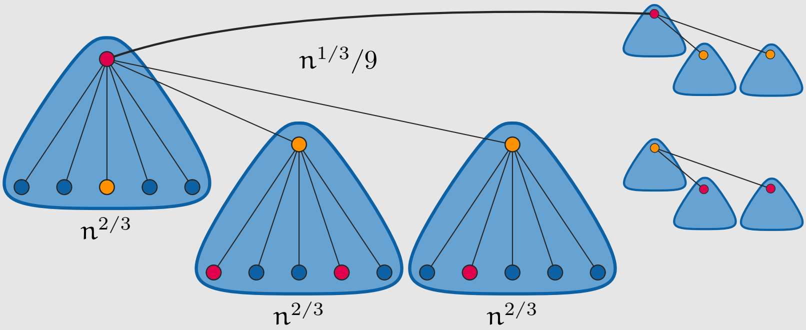
Lower bound

- Roots of small trees are orange or red



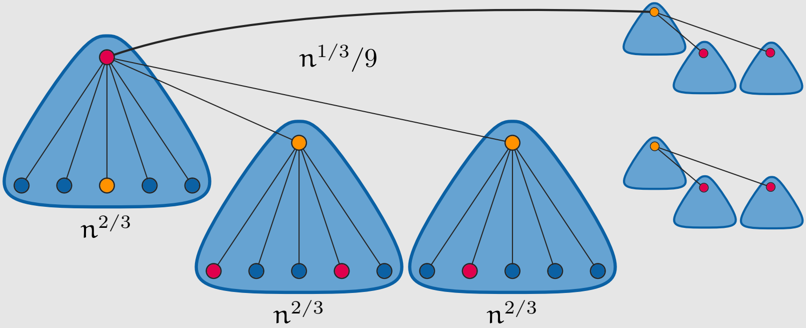
Lower bound

- Connect two big trees with same root color



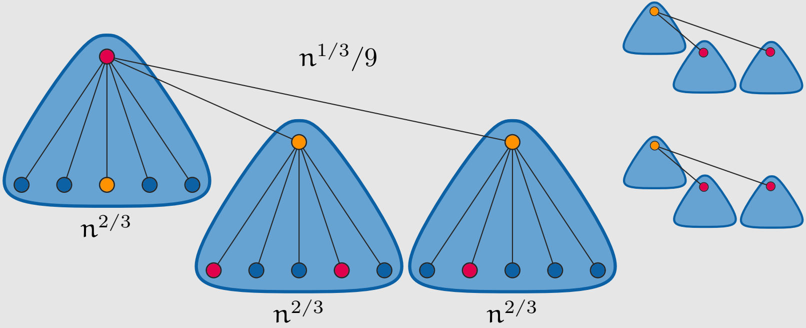
Lower bound

- Forces $\Omega(n^{1/3})$ recolorings



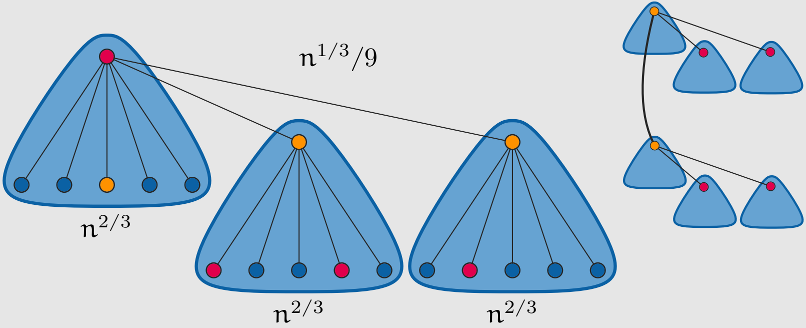
Lower bound

- Repeat $n^{1/3}$ times or until we reset



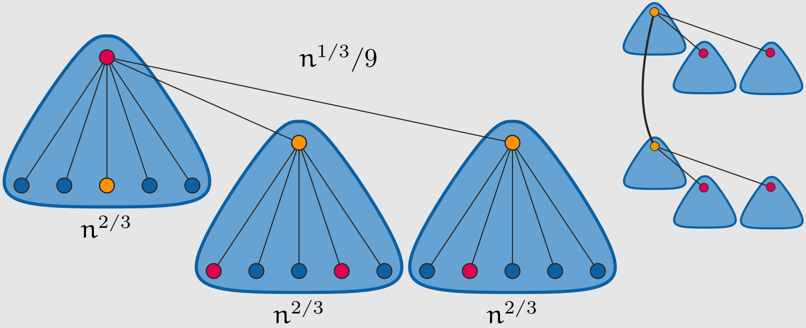
Lower bound

- Repeat $n^{1/3}$ times or until we reset



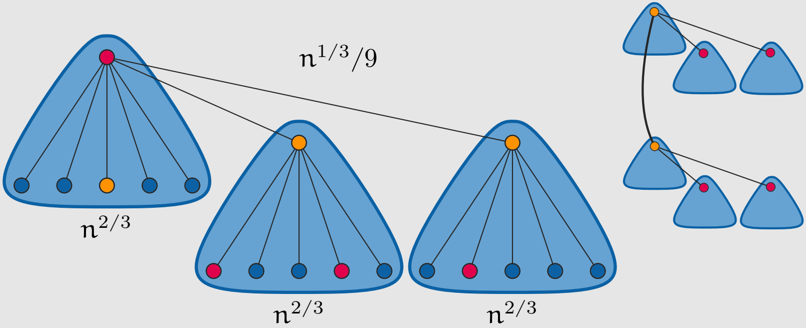
Lower bound

- $\Omega(n^{2/3})$ recolorings either way, for $O(n^{1/3})$ updates



Lower bound

- Maintaining a 3-coloring of a forest requires $\Omega(n^{1/3})$ recolorings per update

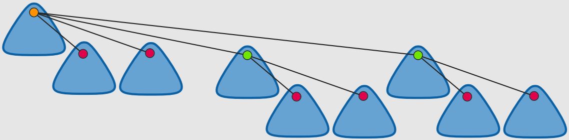


Lower bound

Theorem

For constant c, the number of recolorings per update required to maintain a c-coloring of a forest is

$$\Omega\left(n^{\frac{2}{c(c-1)}}\right).$$



Summary

- Maintain an $O(d)$-approximate coloring with $O(dn^{(1/d)})$ vertex recolorings per update
- Maintain an $O(dn^{(1/d)})$-approximate coloring with $O(d)$ vertex recolorings per update
- Maintaining a c-coloring requires $\Omega(n^{\frac{2}{c(c-1)}})$ recolorings per update

Questions?