Dynamic graphs & vertex coloring

Daniel Gonçalves, LIRMM, Univ. Montpellier & CNRS

JCRAALMA, 15th January 2024 based on


Introduction

$(\Delta + 1)$-coloring with $O(\log n)$ amortized update time

Warmup
Algorithm with *Hierarchical Partition*

*Coloring with arboricity $\alpha$*

Limits of explicit colorings
Implicit & deterministic $2^{O(\alpha)}$-coloring
Implicit & deterministic $O(\alpha^2)$-coloring
Introduction

$(\Delta + 1)$-coloring with $O(\log n)$ amortized update time

Warmup

Algorithm with *Hierarchical Partition*

Coloring with arboricity $\alpha$

Limits of explicit colorings

Implicit & deterministic $2^{O(\alpha)}$-coloring

Implicit & deterministic $O(\alpha^2)$-coloring
Dynamic graph

**Dynamic graph** $G$:

- A fixed vertex set $V$, with $n = |V|$.
- A sequence of updates (edge additions/deletions): $(\pm e_i)_{1 \leq i \leq t}$

- Initially, the edge set is empty: $E_0 = \emptyset$.
- If the $i^{th}$ update is a $+e_i$, then $E_i = E_{i-1} \cup \{e_i\}$.
- If the $i^{th}$ update is a $-e_i$, then $E_i = E_{i-1} \setminus \{e_i\}$.

\[\implies \text{a sequence of graphs: } ((V, E_i))_{0 \leq i \leq t}\]
Dynamic graph

**Dynamic graph G:**

- A fixed vertex set \( V \), with \( n = |V| \).
- A sequence of updates (edge additions/deletions): \((\pm e_i)_{1 \leq i \leq t}\)

- Initially, the edge set is empty: \( E_0 = \emptyset \).
- If the \( i^{th} \) update is a \(+e_i\), then \( E_i = E_{i-1} \cup \{e_i\} \).
- If the \( i^{th} \) update is a \(-e_i\), then \( E_i = E_{i-1} \setminus \{e_i\} \).

\[ \implies \text{a sequence of graphs: } ((V, E_i))_{0 \leq i \leq t} \]

**Combinatorial Problem \( \Pi \)**

- We want solutions of \( \Pi \) for the graphs \((V, E_i)\).
- Computing a solution for \((V, E_i)\), should be easier given a solution of \((V, E_{i-1})\).
### Updating a solution (for some problem $\Pi$) while $G$ evolves

- Given a preset sequence of updates.
- Start with a solution $Sol_0 \in \Pi((V, E_0))$.
- Goal: correct $Sol_0$ after $i$ updates to obtain $Sol_i$. 
Algorithms for dynamic graphs

Updating a solution (for some problem $\Pi$) while $G$ evolves

- Given a preset sequence of updates (oblivious adversary).
- Start with a solution $Sol_0 \in \Pi((V, E_0))$.
- Goal: correct $Sol_0$ after $i$ updates to obtain $Sol_i$. 
Algorithms for dynamic graphs

<table>
<thead>
<tr>
<th>Updating a solution (for some problem Π) while G evolves</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Given a preset sequence of updates (oblivious adversary).</td>
</tr>
<tr>
<td>▶ Start with a solution $Sol_0 \in \Pi((V, E_0))$.</td>
</tr>
<tr>
<td>▶ Goal: correct $Sol_0$ after $i$ updates to obtain $Sol_i$. Two ways:</td>
</tr>
<tr>
<td>▶ <strong>Explicitly</strong>: a current solution is stored in memory.</td>
</tr>
<tr>
<td>Update the solution ($Sol_{i-1} \rightarrow Sol_i$) so that $Sol_i \in \Pi(V, E_i)$.</td>
</tr>
<tr>
<td>▶ <strong>Implicitly</strong>: a solution can be retrieved from queries.</td>
</tr>
</tbody>
</table>
Algorithms for dynamic graphs

Updating a solution (for some problem \( \Pi \)) while \( G \) evolves

- Given a preset sequence of updates (oblivious adversary).
- Start with a solution \( \text{Sol}_0 \in \Pi((V, E_0)) \).
- Goal: correct \( \text{Sol}_0 \) after \( i \) updates to obtain \( \text{Sol}_i \).
  - Two ways:
    - **Explicitly**: a current solution is stored in memory.
      Update the solution (\( \text{Sol}_{i-1} \rightarrow \text{Sol}_i \)) so that \( \text{Sol}_i \in \Pi(V, E_i) \).
    - **Implicitly**: a solution can be retrieved from queries.
Time complexity measures

Case of explicit algorithms

- Time complexity **per update**.
  It can be
  **Worst-case**: maximum complexity for \( Sol_{i-1} \rightarrow Sol_i \),
  or
  **Amortized**: \( \frac{\text{Complexity}(Sol_0 \rightarrow Sol_t)}{t} \) for a suff. large \( t \).
Time complexity measures

Case of explicit algorithms

- Time complexity *per update*.
  It can be
  - **Worst-case**: maximum complexity for \( Sol_{i-1} \rightarrow Sol_i \),
  or
  - **Amortized**: \( \text{Complexity}(Sol_0 \rightarrow Sol_t) / t \) for a suff. large \( t \).

Case of implicit algorithms

- Twofold complexity: *per update & per query*.
  Also, worst-case/amortized options.
Time complexity measures

Case of explicit algorithms

- Time complexity **per update**.
  
  It can be
  
  **Worst-case**: maximum complexity for \( Sol_{i-1} \rightarrow Sol_i \),
  
  or
  
  **Amortized**: \( \text{Complexity}(Sol_0 \rightarrow Sol_t) / t \) for a suff. large \( t \).

Case of implicit algorithms

- Twofold complexity: **per update & per query**.
  
  Also, worst-case/amortized options.

For **randomized algorithms**, the measures can be weakened: e.g. provided in expectation, or with high probability.
Coloring with respect to $\Delta$ or the arboricity $\alpha$

**Brooks theorem**

$$\chi(G) \leq \Delta + 1$$

**Arboricity $\alpha(G)$**

- $\alpha(G)$: $\min k$ s.t. $G$ decomposes into $k$ forests.
## Coloring with respect to $\Delta$ or the arboricity $\alpha$

### Brooks theorem

$$\chi(G) \leq \Delta + 1$$

### Arboricity $\alpha(G)$

- $\alpha(G) : \min k \text{ s.t. } G \text{ decomposes into } k \text{ forests.}$
- Every graph $G$ has at most $\alpha(G) \times (n - 1)$ edges.
Coloring with respect to $\Delta$ or the arboricity $\alpha$

Brooks theorem

$$\chi(G) \leq \Delta + 1$$

Arboricity $\alpha(G)$

- $\alpha(G) : \min k \text{ s.t. } G \text{ decomposes into } k \text{ forests.}$
- Every graph $G$ has at most $\alpha(G) \times (n - 1)$ edges.
- Every graph $G$ is $(2\alpha(G) - 1)$-degenerate.
**Coloring with respect to $\Delta$ or the arboricity $\alpha$**

**Brooks theorem**

$$\chi(G) \leq \Delta + 1$$

**Arboricity $\alpha(G)$**

- $\alpha(G)$: min $k$ s.t. $G$ decomposes into $k$ forests.
- Every graph $G$ has at most $\alpha(G) \times (n - 1)$ edges.
- Every graph $G$ is $(2\alpha(G) - 1)$-degenerate.

$$\chi(G) \leq 2\alpha(G)$$
# Colorings in dynamic graphs

<table>
<thead>
<tr>
<th>Explicit $(\Delta + 1)$-coloring</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(\log \Delta)$ expected amortized update time</td>
</tr>
</tbody>
</table>
## Colorings in dynamic graphs

### Explicit $(\Delta + 1)$-coloring
- $O(\log \Delta)$ expected amortized update time
  
  [BCHN18]

### $f(\alpha, n)$-colorings
- Explicit $O(\alpha \log n)$-coloring, with $O(\log^2 n)$ expected amortized update time.
  
  [HNW20]
## Colorings in dynamic graphs

### Explicit $(\Delta + 1)$-coloring
- $O(\log \Delta)$ expected amortized update time \[ \text{[BCHN18]} \]

### $f(\alpha, n)$-colorings
- Explicit $O(\alpha \log n)$-coloring, with $O(\log^2 n)$ expected amortized update time. \[ \text{[HNW20]} \]
- For explicit $f(\alpha)$-coloring, the update time is $\Omega(poly(n))$. \[ \text{[BCK+19]} \]
### Colorings in dynamic graphs

#### Explicit $(\Delta + 1)$-coloring

- $O(\log \Delta)$ expected amortized update time

#### $f(\alpha, n)$-colorings

- Explicit $O(\alpha \log n)$-coloring, with $O(\log^2 n)$ expected amortized update time.
- For explicit $f(\alpha)$-coloring, the update time is $\Omega(poly(n))$.
- Implicit & deterministic $2^{O(\alpha)}$-coloring, with $O(\log^3 n)$ amortized update time, and $O(\alpha \log n)$ query time.
### Colorings in dynamic graphs

#### Explicit \((\Delta + 1)\)-coloring

- \(O(\log \Delta)\) expected amortized update time \([\text{BCHN18}]\)

#### \(f(\alpha, n)\)-colorings

- Explicit \(O(\alpha \log n)\)-coloring, with \(O(\log^2 n)\) expected amortized update time. \([\text{HNW20}]\)
- For explicit \(f(\alpha)\)-coloring, the update time is \(\Omega(poly(n))\). \([\text{BCK+19}]\)
- Implicit & deterministic \(2^{O(\alpha)}\)-coloring, with \(O(\log^3 n)\) amortized update time, and \(O(\alpha \log n)\) query time. \([\text{HNW20}]\)
- Implicit & deterministic \(O(\alpha^2)\)-coloring, with \(O(\log \alpha \log^3 n)\) worst-case update time, and \(O(\alpha^5 \log n)\) query time. \([\text{CNR23}]\)
Introduction

\((\Delta + 1)\)-coloring with \(O(\log n)\) amortized update time

Warmup

Algorithm with *Hierarchical Partition*

Coloring with arboricity \(\alpha\)

Limits of explicit colorings

Implicit & deterministic \(2^{O(\alpha)}\)-coloring

Implicit & deterministic \(O(\alpha^2)\)-coloring
Introduction

$(\Delta + 1)$-coloring with $O(\log n)$ amortized update time

Warmup

Algorithm with *Hierarchical Partition*

Coloring with arboricity $\alpha$

Limits of explicit colorings

Implicit & deterministic $2^{O(\alpha)}$-coloring

Implicit & deterministic $O(\alpha^2)$-coloring
2Δ-coloring with $O(1)$ expected amortized update time

Current coloring stored in a table $c[·]$.

**Update algorithm after deleting $(u, v)$**

- Do nothing

**Update algorithm after adding $(u, v)$**

- If $c[u] \neq c[v]$: do nothing.
- If $c[u] = c[v]$:
  - $c(v) \leftarrow$ pick a color absent from $N(v)$ u.a.r..
### 2Δ-coloring with $O(1)$ expected amortized update time

Current coloring stored in a table $c[\cdot]$.

#### Update algorithm after deleting $(u, v)$

- Do nothing
  - Time $O(1)$

#### Update algorithm after adding $(u, v)$

- If $c[u] \neq c[v]$: do nothing.
  - Time $O(1)$
- If $c[u] = c[v]$: This case happen with proba. $\leq 1/\Delta$.
  - $c(v) \leftarrow$ pick a color absent from $N(v)$ u.a.r.
  - Time $O(\Delta)$
2Δ-coloring with $O(1)$ expected amortized update time

Neighborhoods stored with two tables $N[\cdot]$ and $E[\cdot, \cdot]$.

![Graph and table illustration]

Update algorithm after deleting $(u, v)$

- Remove $u$ from $N(v)$ & $v$ from $N(u)$  
  Time $O(1)$

Update algorithm after adding $(u, v)$

- Add $u$ in $N(v)$ & $v$ in $N(u)$  
  Time $O(1)$
- If $c[u] \neq c[v]$: do nothing.  
  Time $O(1)$
- If $c[u] = c[v]$:  
  This case happen with proba. $\leq 1/\Delta$.  
  $c(v) \leftarrow$ pick a color absent from $N(v)$ u.a.r.  
  Time $O(\Delta)$
(1 + \(\varepsilon\))\(\Delta\)-coloring with \(O(1/\varepsilon)\) exp. amortized update time

Current coloring stored in a table \(c[\cdot]\).

**Update algorithm after deleting \((u, v)\)**

- Do nothing  \hspace{1cm} \text{Time } O(1)

**Update algorithm after adding \((u, v)\)**

- If \(c[u] \neq c[v]\): do nothing.  \hspace{1cm} \text{Time } O(1)
- If \(c[u] = c[v]\): This case happen with proba. \(\leq 1/\varepsilon\Delta\).
  \(c(v) \leftarrow\) pick a color absent from \(N(v)\) u.a.r..  \hspace{1cm} \text{Time } O(\Delta)

When recoloring, the algorithm picks among at least \((1 + \varepsilon)\Delta - |N(v)| \geq \varepsilon\Delta\) colors.
(1 + \(\varepsilon\))\(\Delta\)-coloring with \(O(1/\varepsilon)\) exp. amortized update time

Current coloring stored in a table \(c[\cdot]\).

<table>
<thead>
<tr>
<th>Update algorithm after deleting ((u, v))</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Do nothing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Update algorithm after adding ((u, v))</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ If (c[u] \neq c[v]): do nothing.</td>
</tr>
<tr>
<td>▶ If (c[u] = c[v]): This case happen with proba. (\leq 1/\varepsilon\Delta).</td>
</tr>
</tbody>
</table>

When recoloring, the algorithm picks among at least \((1 + \varepsilon)\Delta - |N(v)| \geq \varepsilon\Delta\) colors.

Setting \(\varepsilon = 1/\Delta\), we have a \((\Delta + 1)\)-coloring with \(O(\Delta)\) expected amortized update time.
Introduction

$(\Delta + 1)$-coloring with $O(\log n)$ amortized update time

Warmup

Algorithm with *Hierarchical Partition*

Coloring with arboricity $\alpha$

- Limits of explicit colorings
- Implicit & deterministic $2^{O(\alpha)}$-coloring
- Implicit & deterministic $O(\alpha^2)$-coloring
Idea for \((\Delta + 1)\)-coloring

When recoloring a vertex \(v\): pick among \(O(\Delta)\) colors.

- Maybe few vertices in \(\{1, \ldots, \Delta + 1\} \setminus c(N(v))\)
- But in that case, many colors of \(c(N(v))\) are used only once.
Idea for $(\Delta + 1)$-coloring

When recoloring a vertex $v$: pick among $O(\Delta)$ colors.

- Maybe few vertices in $\{1, \ldots, \Delta + 1\} \setminus c(N(v))$
- But in that case, many colors of $c(N(v))$ are used only once.

- Pick among the colors used at most once in $N(v)$, there are $\Delta/2$.
- May create a "path of recolorings".
- How to bound the length of this path?
## Tool: Hierarchical Partition

### Partition of the vertices into levels.

- **L levels**: $V_1, \ldots, V_L$, with $L = \log_\beta \Delta$ for some $\beta > 20$.
- **Level of v**: $\ell(v)$

\[
\begin{align*}
N^< (v) &= \{ u \in N(v) \mid \ell(u) < \ell(v) \} \\
N^\leq (v) &= \{ u \in N(v) \mid \ell(u) \leq \ell(v) \}
\end{align*}
\]

- **[Large $N^< (v)$]**: $|N^< (v)| \geq \beta^{\ell(v)}$ \quad $\forall v \in V$ unless $\ell(v) = 0$.
- **[Small $N^\leq (v)$]**: $C \beta^{\ell(v)} \geq |N^\leq (v)|$
$(\Delta + 1)$-coloring algorithm based on hierarchical partition

$(\Delta + 1)$-coloring : After an update $\pm(u, v)$

1) Insert($u,v$) or Delete($u,v$)
2) Maintain_HP() assume $\ell(v) \leq \ell(u)$ w.l.o.g.
3) If necessary (i.e. $+(u, v) \& c(u) = c(v)$) : Recolor($v$)
$(\Delta + 1)$-coloring algorithm based on hierarchical partition

$(\Delta + 1)$-coloring: After an update $\pm(u,v)$

1) Insert($u,v$) or Delete($u,v$)
2) Maintain$\_HP()$  
   assume $\ell(v) \leq \ell(u)$ w.l.o.g.
3) If necessary (i.e. $+(u,v)$ & $c(u) = c(v)$) :  
   \textbf{Recolor($v$)}

\textbf{Recolor($v$)}

$c(v) \leftarrow$ Pick u.a.r. among colors used at most once in $N(v)$  
& not used in $N(v) \setminus N^{<}(v)$
If $\exists w \in N^{<}(v)$ s.t. $c(w) = c(v)$ :  
   \textbf{Recolor($w$)}
(Δ + 1)-coloring algorithm based on hierarchical partition

(Δ + 1)-coloring: After an update ±(u, v)

1) Insert(u, v) or Delete(u, v)
2) Maintain_HP() assume ℓ(v) ≤ ℓ(u) w.l.o.g.
3) If necessary (i.e. +(u, v) & c(u) = c(v)) : Recolor(v)

Recolor(v)

\[c(v) \leftarrow \text{Pick u.a.r. among colors used at most once in } N(v) \text{ and not used in } N(v) \setminus N^<(v)\]

If \(\exists w \in N^<(v)\) s.t. \(c(w) = c(v)\) : Recolor(w)

- Among consecutive recolorings, the level decreases.
Maintaining the Hierarchical Partition (1)

Data structure

- Matrix $E$
- Doubly chained lists $L_i$ \( \forall i \)

For every $v \in V$:

- $\ell(v)$
- $N^<(v) = \{ u \in N(v) \mid \ell(u) < \ell(v) \}$ \& $d^<(v) = |N^<(v)|$
- $N_i = N(v) \cap L_i$ and $d_i(v) = |N_i(v)|$ \( \forall i \) s.t. $\ell(v) \leq i \leq L$.

For properties [Large $N^<(v)$] \& [Small $N^<(v)$]

- $Q_L$ \& $Q_S$: Queues with vertices violating these properties
Insert\((u,v)\)

- \(\text{neighb}_u \leftarrow \text{New\_Neighbor}(u)\)
- \(\text{neighb}_v \leftarrow \text{New\_Neighbor}(v)\)
- \(E[u, v] \leftarrow (\text{TRUE}, \text{neighb}_u, \text{neighb}_v)\)

- If \(\ell(u) \geq \ell(v)\):
  - then: \(N_{\ell(u)}(v).\text{add(neighb}_u), d_{\ell(u)}(v) ++\)
  - else: \(N^{<}(v).\text{add(neighbor}_u), d^{<}(v) ++\)
- If \(d^{<}(v) + d^{+}_{\ell(v)}(v) > C\beta^{\ell(v)}\) then \(Q_S.\text{push}(v)\)

- If \(\ell(v) \geq \ell(u)\):
  - then: \(N_{\ell(v)}(u).\text{add(neighbor}_v), d_{\ell(v)}(u) ++\)
  - else: \(N^{<}(u).\text{add(neighbor}_v), d^{<}(u) ++\)
- If \(d^{<}(u) + d^{+}_{\ell(u)}(u) > C\beta^{\ell(u)}\) then \(Q_S.\text{push}(u)\)
Maintaining the Hierarchical Partition (3)

Maintain\_HP()

If $Q_S$ is not empty, then

- $v \leftarrow Q_S.pop()$
- $k \leftarrow \min. \ level > \ell(v)$ s.t. $\sum_{i=1}^{k} d_i(v) \leq C \beta^k$
- Move $v$ up to level $k$, and update data structure

Elseif $Q_L$ is not empty, then

- $v \leftarrow Q_L.pop()$
- $k \leftarrow \max. \ level < \ell(v)$ s.t. $C \beta^{k-1} \leq \sum_{i=1}^{k-1} d_i(v)$ or level 1
- Move $v$ down to level $k$, and update data structure

Else return

Maintain\_HP()
Maintaining the Hierarchical Partition (3)

**Maintain\_HP()**

If $Q_S$ is not empty, then

- $v \leftarrow Q_S.\text{pop}()$
- $k \leftarrow \text{min. level } > \ell(v) \text{ s.t. } \sum_{i=1}^{k} d_i(v) \leq C\beta^k$
- Move $v$ up to level $k$, and update data structure

Elseif $Q_L$ is not empty, then

- $v \leftarrow Q_L.\text{pop}()$
- $k \leftarrow \text{max. level } < \ell(v) \text{ s.t. } C\beta^{k-1} \leq \sum_{i=1}^{k-1} d_i(v) \text{ or level 1}$
- Move $v$ down to level $k$, and update data structure

Else return

**Maintain\_HP()**

**Property**

In both cases we have $\beta^k < C\beta^{k-1} \leq d^<(v) \leq d^<=(v) \leq C\beta^k$
Maintaining the Hierarchical Partition (4)

GOAL: $O(\log \Delta) = O(L)$ amortized update time

**Budget function**

- $Budg(uv) = L - \max(\ell(u), \ell(v))$
- $Budg(v) = \frac{1}{2\beta} \max(0, C\beta^{\ell(v)-1} - d^<(v))$
Maintaining the Hierarchical Partition (4)

**GOAL:** \( O(\log \Delta) = O(L) \) amortized update time

**Budget function**

\[
\begin{aligned}
\text{Budg}(uv) &= L - \max(\ell(u), \ell(v)) \\
\text{Budg}(v) &= \frac{1}{2\beta} \max(0, C\beta^{\ell(v)-1} - d^<(v))
\end{aligned}
\]

When adding an edge \( uv \) the budget increase is:
\[
\Delta \text{Budg}(uv) + \Delta \text{Budg}(u) + \Delta \text{Budg}(v) \leq +L - \frac{1}{2\beta} - \frac{1}{2\beta} < L
\]

When deleting an edge \( uv \) the budget increase is:
\[
\Delta \text{Budg}(uv) + \Delta \text{Budg}(u) + \Delta \text{Budg}(v) \leq 0 + \frac{1}{2\beta} + \frac{1}{2\beta} < L
\]
GOAL: $O(\log \Delta) = O(L)$ amortized update time

Budget function

- $Budg(uv) = L - \max(\ell(u), \ell(v))$
- $Budg(v) = \frac{1}{2\beta} \max(0, C\beta^{\ell(v)-1} - d^<(v))$

When adding an edge $uv$ the budget increase is:
$$\Delta Budg(uv) + \Delta Budg(u) + \Delta Budg(v) \leq +L - \frac{1}{2\beta} - \frac{1}{2\beta} < L$$

When deleting an edge $uv$ the budget increase is:
$$\Delta Budg(uv) + \Delta Budg(u) + \Delta Budg(v) \leq 0 + \frac{1}{2\beta} + \frac{1}{2\beta} < L$$

Remaining to prove

One call to $\text{Maintain}_\text{HP}()$ is done in time $O(\beta^{\max(\ell(v), k)})$. Show that this is $O(-\Delta Budg)$. 
Maintaining the Hierarchical Partition (5)

Budget function

- ▶ \( Budg(uv) = L - \max(\ell(u), \ell(v)) \)
- ▶ \( Budg(v) = \frac{1}{2\beta} \max(0, C\beta^{\ell(v)-1} - d^<(v)) \)
Maintaining the Hierarchical Partition (5)

**Budget function**

- \( Budg(uv) = L - \max(\ell(u), \ell(v)) \)
- \( Budg(v) = \frac{1}{2\beta} \max(0, C\beta^{\ell(v)-1} - d^<(v)) \)

When **moving up** \( v \) from level \( \ell(v) \) to \( k \), the budget decrease is:

\[
\Delta Budg(v) + \sum_{u \in N(v)} \Delta Budg(uv) \geq 0 + \sum_{u \in N(v) \text{ with } \ell(u) < k} \frac{1}{2\beta} \\
\geq C\beta^{k-1} \frac{1}{2\beta} \\
\geq O(\beta^{\max(\ell(v), k)})
\]
Maintaining the Hierarchical Partition (5)

**Budget function**

- \( Budg(uv) = L - \max(\ell(u), \ell(v)) \)
- \( Budg(v) = \frac{1}{2\beta} \max(0, C\beta^{\ell(v) - 1} - d^<(v)) \)

When **moving up** \( v \) from level \( \ell(v) \) to \( k \), the budget decrease is:

\[
\Delta Budg(v) + \sum_{u \in N(v)} \Delta Budg(uv) \geq 0 + \sum_{u \in N(v) \text{ with } \ell(u) < k} \frac{1}{2\beta} \geq C\beta^{k-1} \frac{1}{2\beta} \geq O(\beta^{\max(\ell(v), k)})
\]

When **moving down** \( v \) from level \( \ell(v) \) to \( k \): similar ;-)

---

When moving up \( v \) from level \( \ell(v) \) to \( k \), the budget decrease is:

\[
\Delta Budg(v) + \sum_{u \in N(v)} \Delta Budg(uv) \geq 0 + \sum_{u \in N(v) \text{ with } \ell(u) < k} \frac{1}{2\beta} \geq C\beta^{k-1} \frac{1}{2\beta} \geq O(\beta^{\max(\ell(v), k)})
\]

When moving down \( v \) from level \( \ell(v) \) to \( k \): similar ;-)
Introduction

$(\Delta + 1)$-coloring with $O(\log n)$ amortized update time

Warmup

Algorithm with *Hierarchical Partition*

**Coloring with arboricity $\alpha$**
- Limits of explicit colorings
- Implicit & deterministic $2^{O(\alpha)}$-coloring
- Implicit & deterministic $O(\alpha^2)$-coloring
Introduction

($\Delta + 1$)-coloring with $O(\log n)$ amortized update time

Warmup
Algorithm with *Hierarchical Partition*

Coloring with arboricity $\alpha$

Limits of explicit colorings
Implicit & deterministic $2^{O(\alpha)}$-coloring
Implicit & deterministic $O(\alpha^2)$-coloring
2-colorings of dynamic forests

\[ \frac{n}{3} \text{ vertices} \]
$n^{\frac{1}{3}}$ stars $S_{n^{\frac{1}{3}}}$
3-colorings of dynamic forests

$n^{1/3}$ stars $S_{n^{1/3}}$

majority of blue leaves
3-colorings of dynamic forests

$\frac{n}{3}$ stars $S_{\frac{n}{3}}$

majority of blue leaves
3-colorings of dynamic forests

\( n^{1/3} \) stars \( S_{n^{1/3}} \)

majority of blue leaves
3-colorings of dynamic forests

After $O(n^{1/3})$ updates, either

- the thick stars remain 2-colored:
  $O(n^{1/3}) \times O(n^{1/3})$ color changes.

- some thin star has gets a blue root:
  $O(n^{2/3})$ color changes at leaves.
Introduction

$(\Delta + 1)$-coloring with $O(\log n)$ amortized update time

Warmup
Algorithm with *Hierarchical Partition*

*Coloring with arboricity $\alpha$*

Limits of explicit colorings
Implicit & deterministic $2^{O(\alpha)}$-coloring
Implicit & deterministic $O(\alpha^2)$-coloring
**O(α)-forest decomposition**

**Data structure**

- Forests $F_1, \ldots, F_n$
- $\alpha^* = O(\alpha)$ s.t. $F_i = \emptyset \; \forall i > \alpha^*$
- $\forall F_i \; \forall T \in F_i$ is a rooted.
- Query: $\text{DIST}_\text{TO}_\text{ROOT}(v, i)$ $O(\log n)$ time
## $O(\alpha)$-forest decomposition

### Data structure

- Forests $F_1, \ldots, F_n$
- $\alpha^* = O(\alpha)$ s.t. $F_i = \emptyset \ \forall i > \alpha^*$
- $\forall F_i \ \forall T \in F_i$ is a rooted.
- Query: $\text{DIST\_TO\_ROOT}(v, i)$

Query: $\text{COLOR}(v)$

return $(\ldots, \text{DIST\_TO\_ROOT}(v, i) \mod 2, \ldots)$
Introduction

$(\Delta + 1)$-coloring with $O(\log n)$ amortized update time

Warmup
Algorithm with *Hierarchical Partition*

Coloring with arboricity $\alpha$

Limits of explicit colorings
Implicit & deterministic $2^{O(\alpha)}$-coloring
Implicit & deterministic $O(\alpha^2)$-coloring
Reducing to $O(\alpha^4)$ colors

\begin{itemize}
  \item $\alpha^*$-out orientations
  \item $\forall v \quad d^+(v) \leq \alpha^*$
\end{itemize}
Reducing to \(O(\alpha^4)\) colors

\(\alpha^*-\)out orientations

\[\forall v \quad d^+(v) \leq \alpha^*\]

\(r\)-cover free family

There exists a family \(S\) of \(2^{\alpha^*}\) subsets of \(\{1, \ldots, \alpha^*4\}\) such that:

- \(\forall S \in S\)
- \(\forall S_1, \ldots, S_{\alpha^*} \in S\)
- There is a color \(c \in S \setminus (\cup S_i)\).
Reducing to $O(\alpha^4)$ colors

$\alpha^*$-out orientations

$\forall v \quad d^+(v) \leq \alpha^*$

$r$-cover free family

There exists a family $S$ of $2\alpha^*$ subsets of $\{1, \ldots, \alpha^*4\}$ such that:

- $\forall S \in S$,
- $\forall S_1, \ldots, S_{\alpha^*} \in S$,
- There is a color $c \in S \setminus (\bigcup S_i)$.

$2^{2^\alpha^*}$-coloring $c_1$

$\implies \alpha^{*4}$-coloring $c_2$
Reducing to $O(\alpha^4)$ colors

$\alpha^*$-out orientations

\[ \forall v \quad d^+(v) \leq \alpha^* \]

$r$-cover free family

There exists a family $S$ of $2^{\alpha^*}$ subsets of $\{1, \ldots, \alpha^*4\}$ such that:

1. $\forall S \in S$
2. $\forall S_1, \ldots, S_{\alpha^*} \in S$
3. There is a color $c \in S \setminus (\cup S_i)$.

$2^{\alpha^*}$-coloring $c_1$ 
$\implies \alpha^*4$-coloring $c_2$ 
$\implies \alpha^*2$-coloring $c_3$
Thank you!