Roadmap.

§1. Goal [Op.]
§2. Link-cut Tree Structure
§3. Implementations
§4. Amortized Analysis
§5. Applications on Blocking-flows

Goal.

- Represent a forest of rooted trees
- Each node has an arbitrary # of unordered children
- \texttt{FindRoot}(v): return the root of the tree that contains $v$
- \texttt{Cut}(v): delete the edge between $v$ and par($v$).
- \texttt{Link}(v,w): make $v$ a new child of $w$
  \begin{itemize}
  \item \texttt{User guarantee: $v$ is the root of some tree}
  \end{itemize}
- Will consider Update/FindMin to deal with cap./vol. later.

Warm-up.

- Maintain the tree $\Rightarrow O(1)$ cut/link but $O(n)$ FindRoot.
- No Cut op. $\Rightarrow O(m \log n)$ \texttt{Link/FindRoot} using disjoint-union
- [Sleator Tarjan '82] "A data structure for dyn. trees"
  \begin{itemize}
  \item $O(\log n)$ amortized, using Link-cut Trees
  \end{itemize}

Link-Cut Trees.

- Intuition. If there is a single chain - use a splay tree, ordered by depth
  \begin{itemize}
  \item otherwise, do path decomposition, each path by a splay tree.
  \end{itemize}
- A node has been accessed if was passed to some op. above.
- For a node \( v \), its preferred child is the one that contains the last accessed node in \( T(v) \).
- A preferred edge is \((v, w)\) where \( w \) is \( v \)'s preferred child.
- A preferred path is a path of preferred edges. [Draw Original Tree]

![Original Tree](image)

![Link-cut Tree](image)

- Represent (each of) the original tree as a tree of auxiliary trees, one for each preferred path.
- Each auxiliary tree is a (binary) splay tree keyed by depth. [Left = closer to the root]
  [Draw Link-cut Tree]
- Each root of the auxiliary tree has a path-parent pointer. [except for the root]
  [can't store parent-to-child pointers]
Operations/Implementations

- **Access(v):** no actual change to the original tree
  but change the preferred-path decomposition

  [Go to the example, Access(N)]

  - **Spray(v):** bring v to the top of its auxiliary tree, right = descendents on original tree
  - path-parent(right(v)) ← v, right(v) ← null [setting par(right(v)) ← null]
  - while v ≠ root of the link-cut tree do (X)
    - w ← path-parent(v)
    - Spray(w)
    - path-parent(right(w)) ← w, right(w) ← v
    - path-parent(v) ← null
    - v ← w.
    - Spray(v)

- **Find-Root(v):** Access(v)
  - w ← smallest elem. in the auxiliary tree of v.
  - Spray(w), return w.

- **Cut(v):** Access(v)
  - left(v) ← null [no need to set up the path-parent]

- **Link(v,w):** Access(v), Access(w)
  - left(v) ← w.

Amortized Analysis - O(log² n)

- Suffices to show Access(v) is in O(log² n) amortized.

  Each **Spray** O(log n), suffices to show Loop (X) has O(log n) iter. (amortized)

<table>
<thead>
<tr>
<th>Am I cheating?</th>
<th>n A ops.</th>
<th>nB ops.</th>
<th>n² log n/ n = n log² n ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSS...SSS</td>
<td>SSSS...SSS</td>
<td>SSSS...SSS</td>
<td>t Access</td>
</tr>
</tbody>
</table>

⇒ t log n Spray
⇒ t log² n time
- Heavy-Light Decomposition
  - An edge \((v, \text{par}(v))\) is heavy if \(\text{size}(v) > \frac{1}{2} \text{size} (\text{par}(v))\).
  - Light-depth \((v)\): \# of light edges \(\text{root} \leftrightarrow v\). \(O(\log n)\)
  - [An edge in the original graph can be light/heavy, preferred/unpreferred.]

- \# of thst. in \(n\) = \(O(\#\text{ edge that become preferred})\)
- At most \(O(\log n)\) of light edges will become preferred for each Access.
  \(\#(\text{heavy edges become preferred}) \leq \#(\text{heavy edges become unpreferred}) + (n-1)\)
  \[\leq \#(\text{light edges become preferred}) + (n-1) (\pm O(\epsilon))\]

\[\Rightarrow O(\log n)\text{ thst. amortized.} \leq O(t \cdot \log n)
\text{ for } t \text{ Access}\]

- Amortized Analysis \(- O(\log n)\)
  \[T(v) = \log s(v)\]
  \[s(v): \#\text{nodes in the auxiliary tree}\]
  \[\text{link-cut tree} \quad \text{[tree of aux. trees]}\]

- Does not affect ops. in splay trees [imagine that we have weights now]
  \[\text{cost}(\text{splay}(v)) \leq 3(\log s(u) - \log s(v))\]
  \[\text{where } u \text{ is the root of the auxiliary tree that contains } v.\]

- Access \((v)\) takes \(O(\log n)\) amortized, due to telescoping
- \(\text{Cut}(v): \text{potential} \downarrow\)
  \[\text{Link}(v, w): \text{potential} \uparrow\]
  \[\# \text{ of } \leq O(\log n).\]

- Applications
  - Let each edge be associated with a capacity
    store it on the node, \(c(v) = c(v, w)\) where \(w\) is \(v\)'s parent in the original tree.
    \(c(\text{root}) = \infty\)
  - Update \((v, x)\): add \(x\) to all edges on \(\text{root}(v) \to v\).
  - \(\text{Find-Min}\) \((v)\): return min cap. node \(w\) on \(v\)'s path to \(\text{root}(v)\)
• Maintain the following extra fields
  \[ \Delta c(v) = c(v) - c(w) \text{ if } w = \text{par}(v) \text{ on the aux. (splay) tree.} \]
  \[ = c(v) \text{ if } v = \text{root} \]
  \[ \Delta_{\min}(v) = \text{value of the min cap. node in subtree } (v). \]
  \[ \Delta_{\min}(v) = \min(v) - c(v) \leq 0 \]
  \[ \text{argmin}(v) \]

• All in \(O(1)\) to maintain: Splay:

\[ \Rightarrow \]

re-compute

\[ \text{LINK} : \]

• Update \((v, x)\): Access \((u, v)\), \(\Delta(v) \leftarrow \Delta(v) + x\)

Find-\text{min}(v, x): Access \((v)\), return \text{argmin}(v).

\(\text{Cap}(v)\): Access \((v)\), return \(\Delta c(v)\).

□ Blocking-Flow

\[ S \rightarrow O \rightarrow t \]

Step 0: each vertex forms an ind. tree
Step 1: \(v \leftarrow \text{Find-Root} (s)\)
Step 2: If \(v \neq t\) \& (Advance) select an edge \((v, w)\), \(\text{Link} (v, w, \text{cap})\)
  \[ \text{Refine} \]
  \[ \text{if no such edge, for each } (w, v), \text{if in, } \text{Cut} (w). \]

Goto 1.

Step 3: If \(v = t\) (Augment), \(v \leftarrow \text{Find-\text{min}} (s)\)
  \[ \text{Update} (s, -\text{cap}(v)) \]
  \[ \text{Cut}(v) \]

\[ \text{O}(m \log n) \text{ alg. for blocking-flow, } \Rightarrow \text{O}(mn \log n) \text{ for max-flow.} \]
Fixing a bug on page 4.

- I claimed earlier that \#(heavy edges become preferred) ≤ \#(heavy edges become unpreferred) + (n-1), since "a heavy edge must become unpreferred before it can be preferred again."

- The above claim is INCORRECT! The same mistake has also appeared in the lecture notes of 6.854 in 2007, and 6.851 in 2007.

- The correction is as follows:

\[
\#(\text{heavy edges become preferred}) \leq \#(\text{heavy-preferred edges get destroyed}) + (n-1) \\
= \#(\text{heavy-preferred \Rightarrow unpreferred}) + \#(\text{heavy-preferred \Rightarrow light}) + (n-1) \\
= o + \beta + (n-1).
\]

- \(o\) can only happen in Access but, if a heavy edge gets unpreferred, there must be a light one that gets preferred (except one of them for each Access). But, \#(light edges get preferred) = O(b \log n) for each Access. In sum, \(o = O(t \log n) + t = o(t \log n)\).

- \(\beta\) can only happen during Link/Cut.

  - It won't happen in Link(v,w) because all edges on the way \(v \rightarrow \text{root}\) are preferred after Access(v), and they will only get heavier.

  - It happens in Cut(v). All edges on \(v \rightarrow \text{root}\) will be preferred after Access(v), but at most \(O(b \log n)\) of them will become light. \(\Rightarrow \beta = O(t \log n)\)

  \(\Rightarrow \#(\text{heavy edges become preferred}) = O(t \log n)\).