**SOS Talk: Dynamic Connectivity on Forests with Link-Cut Trees**

**Daniel Graf**


**Setting:** Object of study: Forest of rooted trees

**Wanted:** Queries:
- Connectivity \((u,v)\)?
- Are \(u\) and \(v\) in the same tree?

**Updates:**
- InsertEdge \((u,v)\) /where \(v\) is a root
- DeleteEdge \((u,v)\) /makes \(v\) the root of its tree

**Goals:** Small (amortized) runtime for all (and more) operations

**Warmups:**
- Incremental: union-find in \(O(\alpha(n))\) [Tarjan '75]
- Decremental: use component labels for \(O(1)\) queries
  - Upon delete, only relabel the smaller remaining tree
  - Find it with parallel BFS \(\rightarrow O(\log n)\) updates
    (with bit tricks in \(O(1)\)) [Evensh-Shiloach '75]

**Link-Cut Trees** (all in \(O(\log n)\) amortized) [Sleator; Tarjan '83]

**Idea:**
- Take an unbalanced tree
- Decompose it into disjoint paths
- Store end path in a balanced tree

**Preferred Path Decomposition**

<table>
<thead>
<tr>
<th>Preferred child of (v)</th>
<th>(stored implicitly)</th>
<th>(stored explicitly)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(w) if last access to (v) in its subtree</td>
<td>(v) if last access to (w) in subtree</td>
<td></td>
</tr>
<tr>
<td>(v)'s preferred root</td>
<td>(v)'s preferred root</td>
<td></td>
</tr>
<tr>
<td>some older path piece too</td>
<td>some older path piece too</td>
<td></td>
</tr>
</tbody>
</table>

**Basic Operation** (building block for all others)

**access\((v)\)** (make root-v-path preferred)

1. **splay \(v\)** (within its aux-tree)
2. Cut the preferred path below \(v\)
   - \(v\)'s right path.parent = \(v\)
   - \(v\)'s right parent = none
   - Go up the tree of aux trees until \(v\)'s path.parent = none
3. Splay \(w\)
4. Switch, \(w\)'s preferred child
5. Splay \(v\) (rotate)

**Auxiliary Tree**
- Represent each preferred path by a splay tree, keyed by depth
- Root of this aux.tree stores a pointer to the path's parent

<table>
<thead>
<tr>
<th><strong>represent tree</strong></th>
<th><strong>tree of aux trees</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>access((v))</strong></td>
<td><strong>splay (v)</strong> (within its aux-tree)</td>
</tr>
<tr>
<td><strong>splay (v)</strong> (within its aux-tree)</td>
<td><strong>cut the preferred path below (v)</strong></td>
</tr>
<tr>
<td><strong>splay (w)</strong></td>
<td><strong>splay (v)</strong> (rotate)</td>
</tr>
</tbody>
</table>
| **switch, \(w\)'s preferred child** | **exit condition:** \(v\) is the root of the tree of aux trees.
"Standard" Operations

1. findroot(v)
   - access(v) // root-v-path preferred
   - walk left to find r // minimum of aux.tree
   - access(r) // so that it is fast next time

2. link(v, w)
   - access(v)
   - access(w)
   - v.left = w
   - w.parent = v

3. cut(v)
   - access(v)
   - v.left.parent = none
   - v.left = none

"Cool" Operations

4. path_aggregate(v)
   - access(v)
   - return v.value
   - subtree min/max sum aggregate within the aux.splay tree
     (modify access, link, cut accordingly)

5. reroot(v) / event(v) // reverse the direction of
   - all the edges on the root-v-path
   - swap (v.left, v.right)
   - flip lazy-reverse bit of v.right

Quick Analysis $O(\log^2 n)$ per operation

- #splays = #preferred child changes + m $\in O(m \log n)$
- #changes $\leq$ #light pref.edges created + #heavy pref.edges deleted + (n-1)
- Heavy-light decomposition (useful tool for unbalanced trees)
  - (v.parent, v) is heavy if $\text{size}(v) > \frac{1}{2} \text{size}(v.parent)$ (w.r.t. represented tree)
  - light-depth(v) $\leq \log_2(n)$
- check the operations
  - access(v): weights unchanged, only preferredness
    - new root-v-pref. path: $\leq \log n$ light edges
    - deleted pref edges: $\leq \log n$ heavy edges
  - link(v, w): weight of root-w path increases
    - new pref heavy & non-pref light $\Rightarrow O$ changes
  - cut(v): root-v path lightens
    - $\leq \log n$ light pref edges on this path created
    - $\leq 1$ heavy pref edge (v.parent, v) deleted

Continuations
- $O(\log n)$ amortization
- $\Omega(\log n)$ worst case
- applications:
  - Dinic in $O(nm \log n)$
  - Dyn. connectivity on general graphs
  - Tango trees