Dynamic trees (beginning of dynamic graphs)

Problem: maintain a forest of rooted trees topology any # children, unordered

Operations:
- makeTree: return new vertex in new singleton tree
- link(v, w): make v new child of w — add edge (v, w)
  root of tree not containing w
- cut(v): delete edge (v, parent(v)) (v not root)
- findRoot(v): return root of tree containing v

Link-cut trees [Sleator & Tarjan - 1983; Tarjan - 1984]
- even though represented trees in forest may be unbalanced, achieve O(log n) time/op. (amortized)
Link-cut trees are like Tango trees:
- represented tree split into paths:
  - preferred child of node \( v = w \) if last access to \( v \)'s subtree was in \( w \)'s subtree, or none if \( v \) or no last access to \( v \)'s subtree
  - preferred path = chain of preferred edges
  - preferred paths partition nodes of represented tree
- preferred path represented by auxiliary tree
- splay tree keyed on depth
- at root node, store path parent:
  - path's top node's parent in represented tree
  - can't store path children (may be many)
- auxiliary trees + path parent pointers
  - tree of auxiliary trees
  - (potentially high degree)
**access(v):** make preferred path from root to v
& make v the root of auxiliary tree
⇒ v is the root of tree of aux. trees
- splay v (within its aux. tree) ⇒ at root
- remove v's preferred child:
  - path parent(right(v)) ← v
  - right(v) ← none (+sym.)
- until v is in root's preferred path
  - w ← path parent(v) (none ⇒ finished)
  - splay w (within its aux. tree)
  - switch w's preferred child to v:
    - path parent(right(w)) ← w
    - right(w) ← v (+sym.)
    - path parent(v) ← none
  - v ← w

+sym.: + symmetric
setting of parent ptr.
\textbf{findroot}(v):
- \text{access}(v) \Rightarrow \text{root of aux. tree containing root}
- \text{find min. (depth) node in aux. tree:}
  - v \leftarrow \text{left}(v) \text{ until } \text{left}(v) = \text{none}
- \text{splay } v \text{ - to make faster for next time}
- \text{return } v

\textbf{cut}(v):
- \text{access}(v)
- \text{declare } \text{left}(v) \text{ to root new tree of aux. trees:}
  - \text{left}(v) \leftarrow \text{none} \quad (+\text{sym.})
  \text{note: } \text{right}(v) = \text{none too: no preferred child}

\textbf{link}(v, w):
- \text{access}(v) \Rightarrow v \text{ alone in pref. path/aux. tree}
- \text{access}(w) \Rightarrow w \text{ at root}
- \text{left}(v) \leftarrow w \Rightarrow v \text{ becomes deepest node}
  \quad (+\text{sym.}) \quad \text{in } w\text{'s preferred path}
\(O(lg^2 n)\) bound: (amortized)
- link & cut cost \(O(1)\) beyond access
- findroot costs access + time to find & splay min.
- access costs time to splay \(\cdot \) # preferred child changes
- splay analysis works in this setting (splits/concats)
\(\Rightarrow O(lg n) \cdot (m + \text{total } \# \text{ preferred child changes})\)
\text{claim: } O(m \ lg n)

**Heavy-light decomposition:** in represented tree
- \(\text{size}(v) = \# \text{ nodes in } v\text{'s subtree}\)
- edge \((v, \text{parent}(v))\) is \text{heavy} if \(\text{size}(v) > \frac{1}{2} \text{size(parent}(v))\)
  & \text{light} otherwise
\(\Rightarrow \text{at most 1 heavy child of each node}\)
- \text{heavy paths} partitions nodes of tree
- \text{light depth} \((v) = \# \text{ light edges on root-to-} v \text{ path} \leq lg n\)

\(\Rightarrow \text{represented edge can be (preferred) & (heavy)}\)
\(\text{not (light)}\)
$O(m \lg n)$ preferred child changes
- count # light preferred edge creations
- access(v) creates preferred edges along root-v path
  $\Rightarrow \leq \lg n$ of them are light - so all about heavy
- count # heavy preferred edge destructions
  & add $\leq n+1$ for end configuration
- destruction of heavy preferred edge
  $\Rightarrow$ creation of light preferred edge except possibly preferred child of v
  $\Rightarrow \leq \lg n + 1$
  $\Rightarrow$ access(v) creates $O(\lg n)$ preferred edges

- link $(v, w)$ “heavens” nodes on root-to-w path
  $\Rightarrow$ edges hanging off this path may become light
  (worry: create light pref./destroy heavy pref.)
- but access(w) made these edges unpreferred

- cut(v) lightens ancestors of v
- but $\leq \lg n$ of them can become light
- also possibly destroy heavy pref. edge $(v, \text{parent}(v))$
$O(\lg n)$ bound: (amortized)
- $W(v) = \# \text{ nodes in } v\text{'s subtree in tree of aux. trees}
  = \# \text{ nodes in } v\text{'s subtree within } v\text{'s aux. tree}
  + \sum \# \text{ nodes in each "descendent" aux. tree of these nodes}
- potential $\Phi = \sum_v \lg W(v)$ \hspace{1cm} \text{(splay potential)}
- claim: amortized cost of access
  $= O(\lg n) + O(1) \cdot \# \text{ preferred child changes}
  \hspace{1cm} (= O(\lg n) \text{ amortized})$

Proof:
- splay($v$) costs $\leq 3(\lg W(u) - \lg W(v)) + 1$
  where $u$ = root of $v$'s auxiliary tree
- splay($v$) affects $W$'s only within $v$'s aux. tree
  $\Rightarrow$ standard splay analysis holds
- changing $v$'s preferred child doesn't change $W$'s: tree of aux. trees stays same
- $W(v) \leq W(w)$ - $w$ next element to splay
  $\Rightarrow$ telescopes
  $\Rightarrow$ access cost $\leq 3(\lg W(r) - \lg W(v)) + O\left(\# \text{ pref. changes}\right)$
  $= O(\lg n)$
- cut($v$) only decreases $W$'s $\Rightarrow \Phi$ only decreases
- join($v, w$) just increases $W(w)$ by $\leq n$
  $\Rightarrow O(\lg n)$ increase in $\Phi$\hspace{1cm} $\blacksquare$