Today: Dynamic graphs I (of 3)
- link-cut trees
- preferred paths (again) [LG]
- heavy-light decomposition

Link-cut trees: [Sleator & Tarjan-1983; Tarjan-1984]
maintain forest of rooted (unordered) trees subject to \(O(lg n)\)-time operations:
- makeTree: return new vertex in new tree
- \(\text{link}(v, w)\): make \(v\) new child of \(w\) \(\Rightarrow\) adding edge \((v, w)\)
- \(\text{cut}(v)\): delete edge \((v, \text{parent}(v))\)
- \(\text{findRoot}(v)\): return root of tree containing \(v\)
- \(\text{pathAggregate}(v)\): compute sum/min/max/etc. of node/edge weights on \(v\)-to-root path

Idea: represent unbalanced trees using balanced trees
Preferred path decomposition: (like Tango trees [L6])
- preferred child of node $v$:
  = \{ \text{none} if last access in $v$'s subtree was $v$
  \{ $w$ if last access was in child $w$'s subtree
- preferred path = chain of preferred edges
\Rightarrow partition represented tree into paths

Auxiliary trees: (also like Tango trees [L6])
represent each preferred path by a Splay tree keyed on depth
- root of aux tree stores path parent: path's top node's parent in represented tree (can't easily store path children ~can be many)
- auxiliary trees + path parent pointers
  = tree of auxiliary trees
  - potentially high degree
  - goal: balanced
access(v): make root-to-v path preferred
& make v the root of its aux. tree
⇒ v is the root of tree of aux. trees

- splay v (within its aux. tree)
- remove v's preferred child:
  - w = v. pathparent
- until v. pathparent = none: (i.e. root aux. tree)
  - w = v. pathparent
  - splay w (within its aux. tree)
  - switch w's preferred child to v:
    - if w. right.
      - w. right. pathparent = v
        - w. right.
          - w. right. parent = none
        - w. right = v
      - v. parent = w
        - v. pathparent = none
    ⇒ splay v = rotate v
⇒ v. pathparent = w. pathparent

⇒ v has no right child
(deepest node on preferred path
because v has no preferred child)
\[ \text{findroot}(v): \]
- \text{access}(v)
- \( v = v.\text{left} \) until \( v.\text{left} = \text{none} \)
- \text{splay} \( v \rightarrow \) so fast next time
- return \( v \)

\[ \text{path-aggregate}(v): \] (for vertex weights)
- \text{access}(v)
- return \( v.\text{subtree sum} \) augmentation within each aux. tree

\[ \text{Cut}(v): \]
- \text{access}(v)
- \( v.\text{left}, \text{parent} = \text{none} \)
- \( v.\text{left} = \text{none} \)
- \( v \rightarrow \text{root} \)

\[ \text{link}(v, w): \] \( \Rightarrow v \) alone in its aux. tree
- \text{access}(v)
- \text{access}(w)
- \( v.\text{left} = w \)
- \( w.\text{parent} = v \)
- \( v \) becomes deepest node in \( w \)’s preferred path

\[ \text{OR} \ w.\text{right} = v \sim \text{similar analysis} \]
$O(\log^2 n)$ amortized bound:
- link & cut & path-aggregate cost $O(1 + \text{access})$
- findroot costs access + find/splay min
- access costs splay · #preferred child changes
- \textbf{lemma}: splay analysis works in this setting (or use balanced BSTs)

$\Rightarrow O(\log n)$ amortized/splay

$\downarrow m$ operations cost
$O(\log n) \cdot (m + \text{total # preferred child changes})$

\textbf{claim:} $O(m \log n)$
- for this, need a tool:

\textbf{Heavy-light decomposition:} (in represented tree)
- $\overline{\text{size}}(v) =$ # nodes in $v$’s subtree
- call edge $(v, \text{parent}(v))$:
  - \textbf{heavy} if $\text{size}(v) > \frac{1}{3} \text{size}(\text{parent}(v))$
  - \textbf{light} otherwise

$\Rightarrow \leq 1$ heavy child of a node
$\Rightarrow$ heavy edges form heavy paths
  which partition the nodes
- $\overline{\text{light depth}}(v) =$ # light edges on root-to-$v$ path
  \leq $\log n$ (size halves each time)

$\Rightarrow$ represented edge can be (preferred) \textbf{not} (heavy) (light)
\(O(m \lg n)\) preferred child changes:

- \(\#\text{changes} \leq \# \text{light preferred edge creations} + \# \text{heavy preferred edge destructions} + n-1\)
- \#edges \sim \text{in case created \& not destroyed or destroyed \& not created}

- access(v):
  - creates preferred edges along root-to-v path
  - \(\leq \lg n\) of them can be light
  - each heavy preferred edge destroyed \(\Rightarrow \text{light preferred edge created}\)
  - except former preferred child of \(v\) \(\Rightarrow \leq \lg n + 1\)
  - \(\Rightarrow O(\lg n)\) total

- link(v, w): "heavens" nodes on root-to-w path
  - some of these edges might become heavy
  - some edges off path might become light
    - \(\Rightarrow\) create light edges \& destroy heavy edges
  - but former preferred \& latter not, by access
  - \(\Rightarrow \emptyset\)

- cut(v): lightens nodes on root-to-v path
  - \(\leq \lg n\) of path edges can be (come) light
  - also destroy edge \((v, \text{parent}(v))\), possibly heavy
  - \(\Rightarrow O(\lg n)\)
$O(lg n)$ amortized bound:
- $W(v) = \#$ nodes in $v$'s subtree in tree of aux. trees
  $= \sum_{w \in \text{aux.} \exists v} (1 + \text{size(aux. trees hanging off } w))$
- potential $\Phi = \sum_{v} lg W(v) \sim \text{splay potential}$
- access lemma: amortized cost of splay($v$)
  $\leq 3(lg W(\text{root of } v\text{'s aux. tree}) - lg W(v)) + 1$
  - splay($v$) affects $W$'s only within $v$'s aux. tree
  $\Rightarrow$ standard splay analysis applies:
  - amortized cost of one splay step
    $\leq 3(lg W_{after}(v) - lg W_{before}(v))$
    (some checking & concavity of $lg$)
  $\Rightarrow$ telescopes, +1 for final rotation
- amortized cost of access($v$)
  $= O(lg n) + O(\# \text{ preferred child changes})$
  $\Rightarrow O(lg n)$ amortized
- changing preferred children doesn't affect $W$
  (tree of aux. trees remains the same)
- $W(v) \leq W(\text{root of } v\text{'s aux. tree}) \leq W(w)$
- splay($v$) costs $\leq 3(lg W(w) - lg W(v)) + 1$
- sum telescopes
  $\Rightarrow \leq 3(lg W(\text{root}) - lg W(v)) + O(\# \text{ preferred child changes})$
- cut($v$) only decreases $W$'s $\Rightarrow \Phi$ only decreases
- link($v, w$) increases only $W(v)$, by $\leq n$
  $\Rightarrow \leq lg n$ increase in $\Phi$
Worst-case $O(\log n)$: [Sleator & Tarjan]
- store heavy paths in aux. trees
- aux. tree = globally biased search tree
  [Bent, Sleator, Tarjan – SICOMP 1985]
- similar to weight-balanced trees in L16 but dynamic with careful split/concat.
6.851 Advanced Data Structures
Spring 2012

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.