Paper chase due in 2 weeks

I am traveling next week — no lectures / OH

Dynamic Forests:

unrooted, undirected

Maintain a collection of vertex-disjoint trees

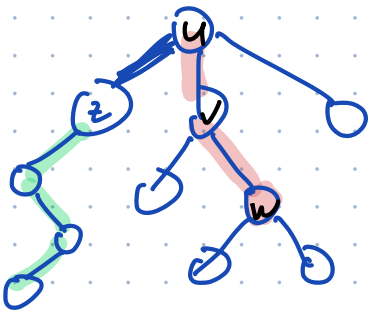
Structural: Cut, link, find root
Subtree: Sum, Min, Add, Mul, Set, Negate
Path: " " " " " "

Last time: Euler tour tree — subtree + struct
$O(\log n)$

ST-trees: path + struct $O(\log n)$

Root trees

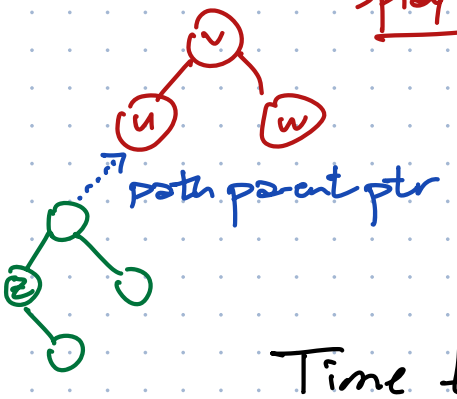
Every node v has a preferred child ptr:
points toward last node accessed in subtree @ v.



After access(v)
all edges on path from root to v are preferred
but no edges touching path

Each pref. path is stored in a path tree

BST of path nodes in depth order
splay trees



Make an edge in T not preferred:
split path tree
Make edge preferred
concat 2 path trees.
— $O(\log n)$ amortized time

Time for access(v) =

$O($# non-pref edges on path to $v) \cdot O(\log n)$

(1) make this +

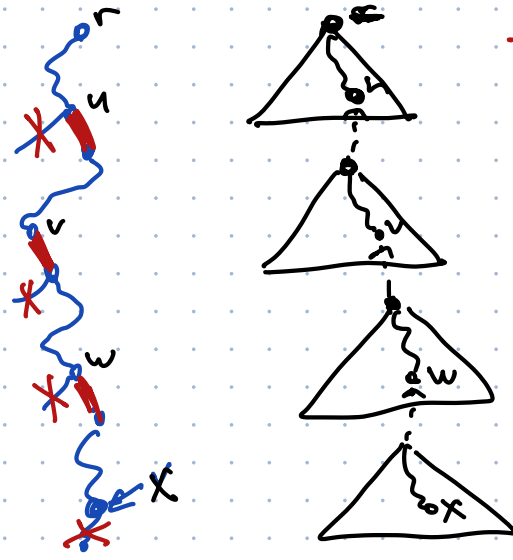
Access Lemma:

Am. cost for splay(v) is $\leq 1 + 3\,\text{newrank}(v) - 3\,\text{oldrank}(v)$

$\text{rank}(v) = \lg \text{size}(v) = \lg \sum_{w \text{ in } v} \text{wt}(w)$ ← purely for analysis

Define wt(w) =

nodes in all path trees connected to w by path-parent ptr



$1 + 3\text{rank}_4(x) - 3\text{rank}_3(x)$

$1 + 3\text{rank}_3(x) - 3\text{rank}_2(x)$

$1 + 3\text{rank}_2(x) - 3\text{rank}_1(x)$

$1 + 3\text{rank}_1(x) - 3\text{rank}_0(x)$

$\leq$ (# path trees) $+ O(\log n)$

(2) amortized $O(\log n)$

Claim: amortize

preferred ptr changes = $O(\log n)$

Heavy-light decomposition

Edge from parent u to child v in T is heavy

if $\text{size}_T(v) > \frac{1}{2}\text{size}_T(u)$

Every node has ≤ 1 heavy child

Claim: Any root-to-leaf path in T has $\leq \lg n$ light edges.

Proof: Defs of light and lg n. □

Edges of T can be either heavy or light

and either preferred or not

$\#\text{edges} \to \text{pref} = \underbrace{\#\text{light} \to \text{pref}}_{\leq \lg n} + \#\text{heavy} \to \text{pref}$

$\sum_{\text{ops}} \#\text{heavy} \to \text{pref} \leq \sum_{\text{ops}} \#\text{heavy} \to \text{not pref} + n - 1$

(#edges stop being both pref + heavy)

Access: $\#\text{heavy} \to \text{not pref} \leq 1 + \#\text{light} \to \text{pref} \leq 1 + \lg n$

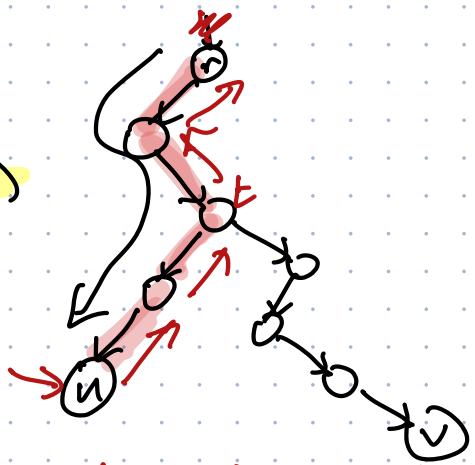
Cut: Some edges on access path heavy → light $\leq \lg n$
off access path light → heavy $\leq \emptyset$

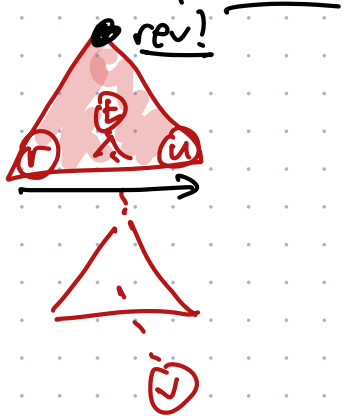
Link: some edges on path light → heavy
off path heavy → light $\Rightarrow \emptyset$

Dealing with roots

PathQuery(u,v):
→ access(u)
$O(\log n)$ amtime → makeroot(u)
→ access(v)

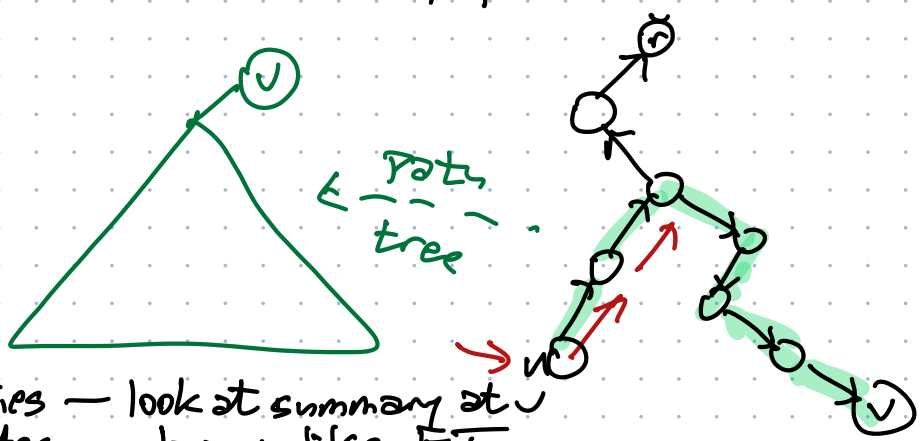
Abstract tree T is unrooted





To make u the root
reverse path from root to u

Set reverse bit in top path tree
Lazily update

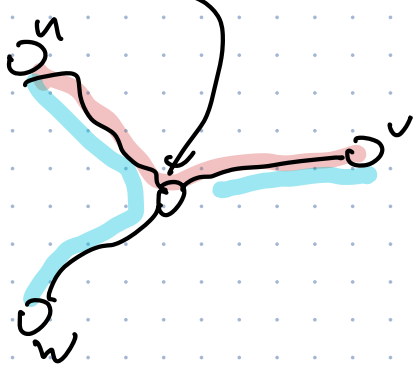


Queries — look at summary at v
Updates — lazy like ET

LCA(u,v)

O(log n) am

join(u,v,w)



access(u)
makeroot(w)
access(v)
access(w)
access(v) ⟶ only one edge changes to pref