ART decomposition

- for each maximally high node whose subtree contains $\leq \lg n$ leaves.
  (no ancestor relations between such nodes)
  define bottom tree = this subtree ($\Rightarrow < \lg n$ branching nodes)
- top tree on remaining $O(n/\lg n)$ leaves [& roots of bottom trees]
  $\Rightarrow$ after second application, top tree has $O(n/\lg n)$ nodes
  (or can deal with compression in top instead of bottom)

Decremental connectivity in a tree

- goal: $O(1)$ amortized assuming all $n-1$ edges eventually deleted
- in top tree of $O(n/\lg n)$ nodes, can afford $O(\lg n)$ am. cost
  ($\Rightarrow$ total cost $= O(n)$)
- $O(\lg n)$ amortized solution: [or use link-cut trees] [or use Euler-tour trees]
  - each node stores explicit component id.
  - delete(v, w):
    - run DFS from both v & w in parallel
    - stop when one DFS stops $\Rightarrow$ smaller component
    - update all nodes in smaller component to new id.
  - component containing updated node shrinks by $2\times$
    $\Rightarrow O(\lg n)$ charges to each node.
- in bottom tree, use $O(1)$ solution for each maximal nonbranching path for queries to nodes inside such paths:
  - break path into $O(n/\lg n)$ chunks of $\lg n$ nodes.
  - use $O(\lg n)$ am. solution to find chunk with lowest ancestral deleted edge
  - store each chunk in word of $\lg n$ bits
    $\Rightarrow$ can find predecessor deleted edge in $O(1)$ (shift + LSB)
  - delete affects 1 chunk & structure of chunks
- in bottom tree, use 1 word to represent compressed tree
  ↳ bits specify which paths have deleted edges
  - each node stores bitmask of ancestors in bottom tree
  - bits ordered by depth
    $\Rightarrow$ mask, shift, LSB finds lowest ancestral path with a deleted edge
  - if any, recurse in path
  - if not, recurse in top tree