⑤ combine jump pointers → expon. decreasing hops
& ladder decomp. → expon. increasing hops

- query: 1 jump pointer ⇒ go up $\lfloor\lfloor \ell \rfloor\rfloor \geq \ell/2$
  ⇒ height(v') $\geq \ell/2$
  1 ladder step ⇒ ladder(v') extends $\geq \ell/2$ above v'
  ⇒ contains ℓ^{th} ancestor of v
  $= (\leq \ell/2)^{th}$ ancestor of v'

⇒ O(1) query
- O(n lg n) space

⑥ jump-pointer tuning:
- each node v stores descendant leaf, leaf(v)
- depth-d ancestor of v = depth-d ancestor of leaf(v)
⇒ can start queries at leaves
⇒ only need jump pointers at leaves
⇒ O(n + L lg n) space where L = # leaves
ladder decomp. // jump pointers

⑦ ART decomposition:
- top tree has O(n/lg n) leaves
⇒ ⑥ uses O(n) space (& achieves O(1) query)
- bottom trees have O(lg n) nodes
$\leq \frac{1}{4} \lg n$ for appropriate constants
- # distinct trees on $\frac{1}{4} \lg n$ nodes
$= \underbrace{C_{\frac{1}{4}\lg n}}_{\text{Catalan \#}} \leq 4^{\frac{1}{4}\lg n} = \underbrace{2^{2(\frac{1}{4}\lg n)}}_{\text{2n-bit encoding: Euler-tour tree; 0 for "up", 1 for "down"}} = 2^{\frac{1}{2}\lg n} = \sqrt{n}$
- mega lookup table: for each possible tree,
① lookup table of size $\frac{1}{4} \lg n$
(query = node # in Euler-tour order + ℓ or d
answer = ditto)
⇒ $O(\sqrt{n} \lg^2 n)$ space, O(1) query
- query: look in bottom tree, else get parent(root), query top
⇒ O(n) space, O(1) query