6.897
L24.2
May 5, 2005

Insert x after y:
- add x after y in y's leaf by re-arranging $O(\lg n)$ elts. in leaf
- if leaf was full (outside threshold)
  - walk up to first ancestor within threshold
  - rebalance all descendant elements, evenly distributing among all descendant slots

Delete x:
- remove x from its leaf & re-arrange the $O(\lg n)$ elts
  ↳ preserve gaps of $O(1)$
- if leaf $< 1/4$ full, bring within threshold as before

Analysis:
- thresholds get looser as you go deeper
- after rebalancing node at its thresholds, children have densities within that threshold
  ⇒ they are far within their thresholds
  ↳ by at least threshold difference, $\sim \frac{1/4}{h} = \Theta\left(\frac{1}{\lg N}\right)$
- before this node is rebalanced again, one of its children must again go outside threshold which requires $\Omega\left(\frac{\text{capacity}}{\lg N}\right)$ updates in the subtree
  (of child or parent $\sim \Theta$ of each other)
  ↳ here need leaves to have $\Omega(\lg n)$ size (not e.g. $O(1)$) so that this ratio is always ≥ 1 (to avoid floor)

⇒ amortized amount of rebalancing caused by inserting/deleting below a node is $\Theta(\lg N)$
- each element is below $h = \Theta(\lg N)$ ancestors

⇒ amortized amount of rebalancing per update $= \Theta(\lg^2 N)$

Worst-case bounds possible [Willard – I&C 1992]
[Bender, Cole, Demaine, Farach-Colton, Zito – ESA 2002]