Merging Heaps

How can we make it fast?

- Array-based implementation:
- Pointer-based implementation:

Leftist Heaps

- Idea:

  make it so that all the work you have to do in maintaining a heap is in one small part
- Leftist heap:
  - almost all nodes are on the left
  - all the merging work is on the right

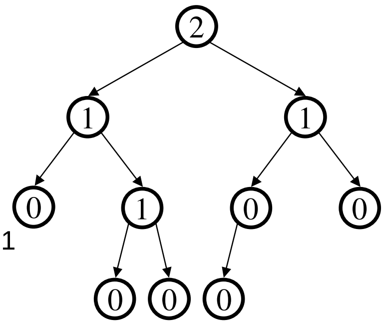
Not-so Random Definition: Null Path Length

the *null path length (npl)* of a node is the number of nodes between it and a null in the tree

- npl(null) = -1
- npl(leaf) = 0
- npl(single-child node) = 0

npl(n) = min(npl(left(n)), npl(right(n)))+ 1

another way of looking at it:
npl is the height of complete subtree rooted at this node



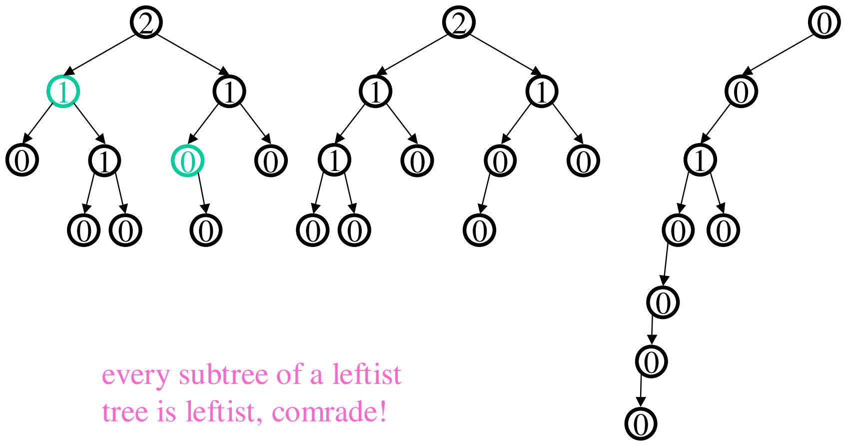
Leftist Heap Properties

- Heap-order property
  - parent's priority value is ≤ to childrens' priority values
  - result: minimum element is at the root
- Leftist property
  - null path length of left subtree is ≥ npl of right subtree
  - result: tree is at least as "heavy" on the left as the right

Are leftist trees complete? Balanced?

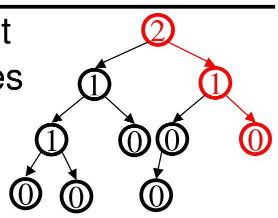
Leftist tree examples

NOT leftist — leftist — leftist



every subtree of a leftist
tree is leftist, comrade!

Right Path in a Leftist Tree is Short



- If the right path has length at least r, the tree has at least $2^r - 1$ nodes
- Proof by induction

  Basis: $r = 1$. Tree has at least one node: $2^1 - 1 = 1$

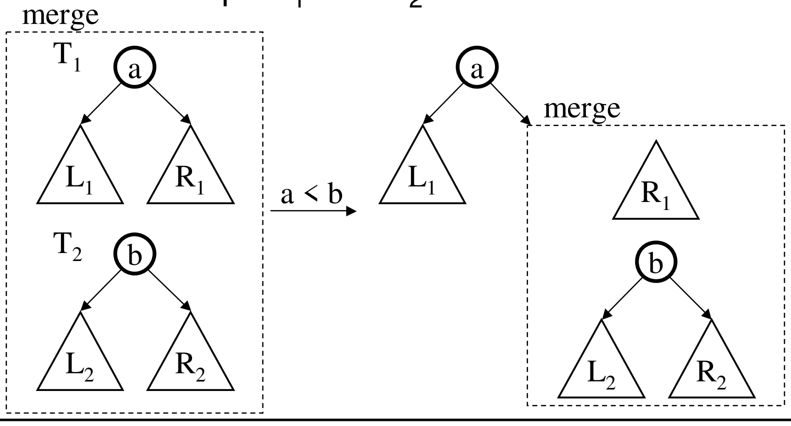
  Inductive step: assume true for $r' < r$. The right subtree has a right path of at least r - 1 nodes, so it has at least $2^{r-1} - 1$ nodes. The left subtree must also have a right path of at least $r - 1$ (otherwise, there is a null path of $r - 3$, less than the right subtree). Again, the left has $2^{r-1} - 1$ nodes. All told then, there are at least:

  $$2^{r-1} - 1 + 2^{r-1} - 1 + 1 = 2^r - 1$$

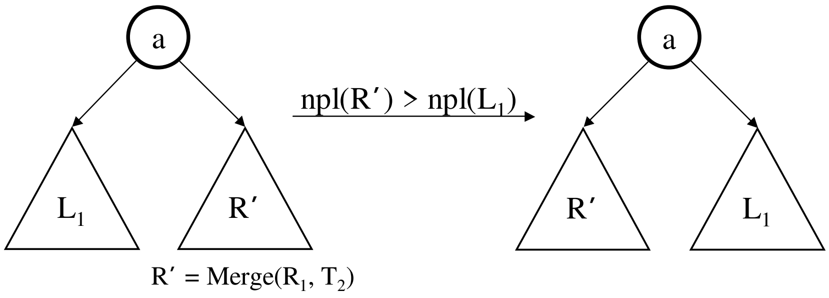
- So, a leftist tree with at least n nodes has a right path of at most $\log n$ nodes

Merging Two Leftist Heaps

- merge(T_1,T_2) returns one leftist heap containing all elements of the two (distinct) leftist heaps T_1 and T_2



Merge Continued

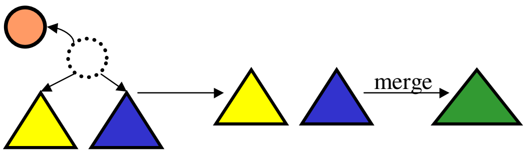


runtime:

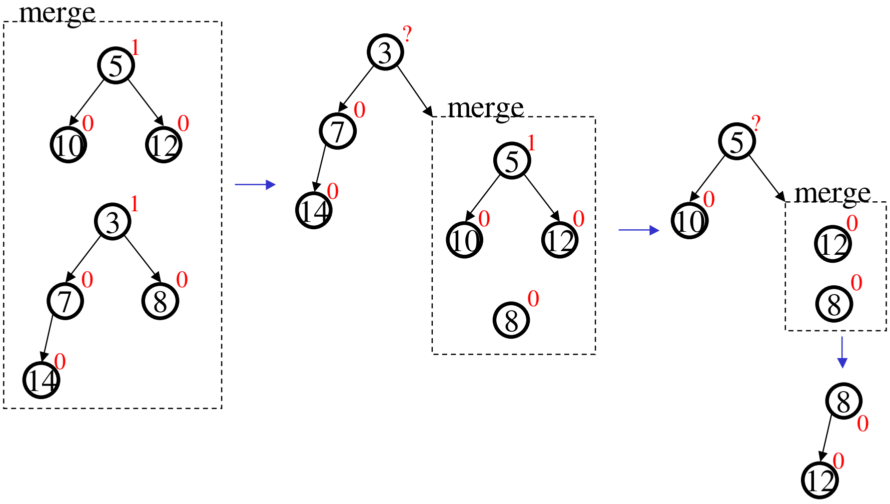
Operations on Leftist Heaps

- merge with two trees of total size n: O(log n)
- insert with heap size n: O(log n)
  - pretend node is a size 1 leftist heap
  - insert by merging original heap with one node heap

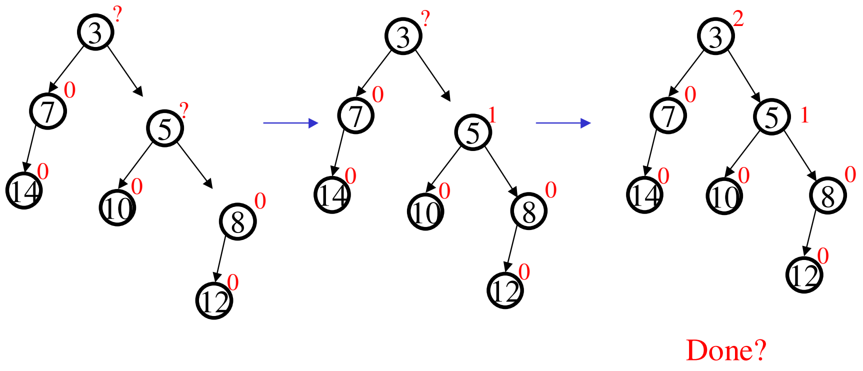
- deleteMin with heap size n: O(log n)
  - remove and return root
  - merge left and right subtrees



Example



Sewing Up the Example



Finally…