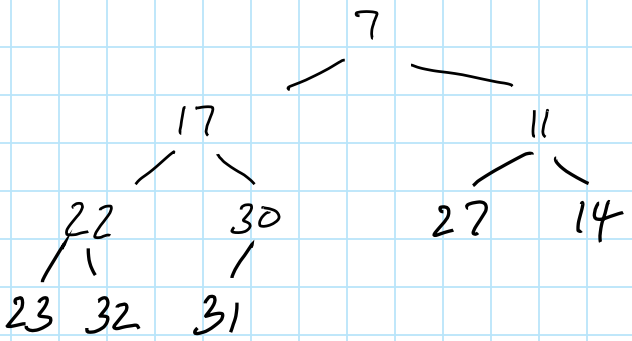
lecture 9 – Sep 23



heap:

- complete binary tree
- heap-order property

(entry at node n, where $n \neq$ root, is at least as large as entry at parent of n)

Insertion into a heap

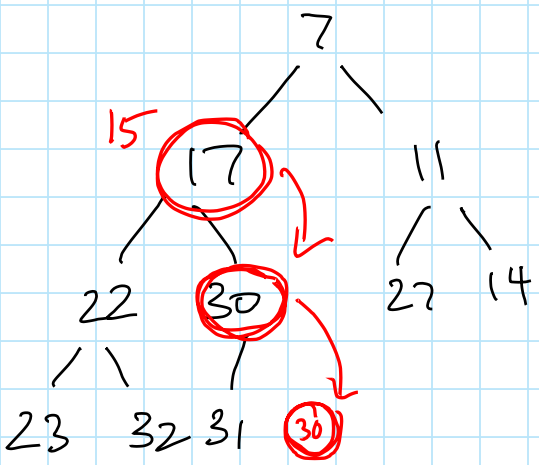
to insert k,

- next position p in the array (= next free leaf position in the tree) must be filled
- as long as $p \neq 1$ and $k <$ array$[\frac{p}{2}]$
  - array$[p]$ = array$[\frac{p}{2}]$
  - $p = p/2$
- array$[p] = k$

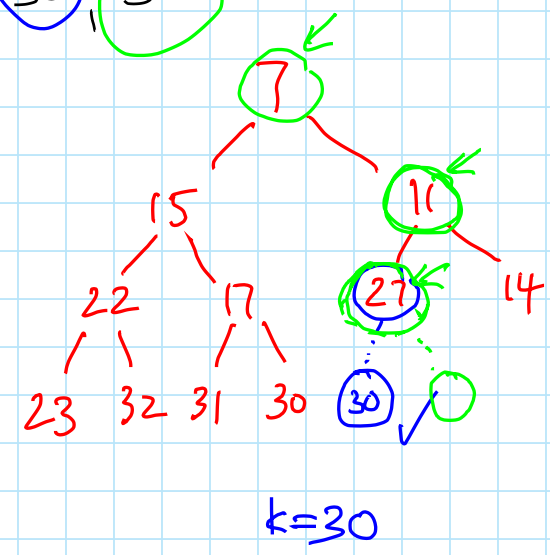
"percolate up" strategy

Example 18.

insert 15, 30, 5



$k = 15$



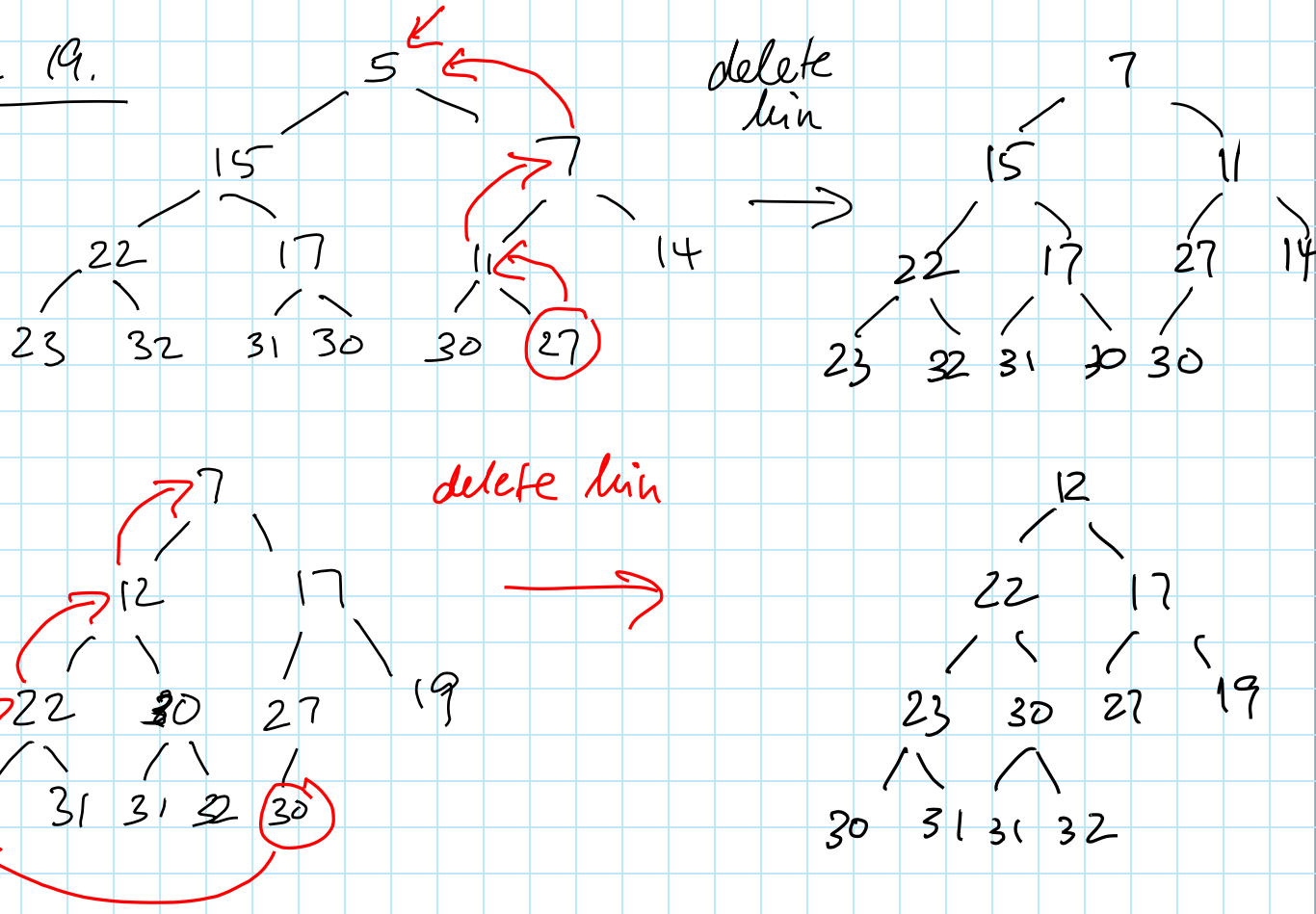
$k = 30$

complexity: $T_{worst}(N) = \Theta(\log(N))$

deletion from a heap to delete the minimum,

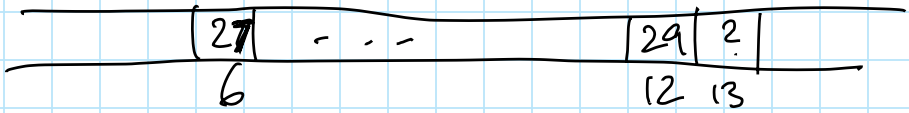
- identify the last element k in the array (the last leaf entry)
- percolate k down from the root, always moving up the smaller of the two children.

Example 19.



Watch out when implementing deletion!

What if right child empty?



2: could be a small integer and moved up when percolating down.

trick: replace "empty" array cells by sentinels (value larger than any heap entry)

Other heap operations

- findMin: easy, $\Theta(1)$
- findMax: need to check all the leaves, i.e., half the nodes! $\frac{N}{2} = \Theta(N)$.