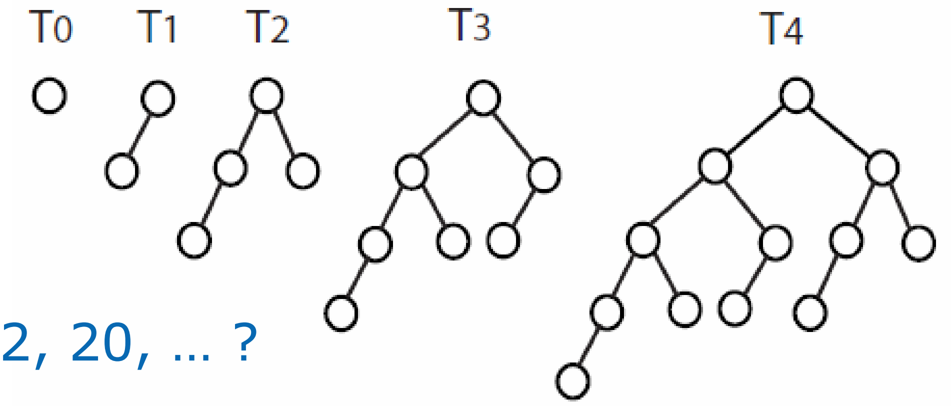
§2 Recap: AVL Trees

Adelson-Velsky & Landis 1962: $h \leq O(\log n)$

Heights of any two sibling subtrees must differ by *__at most one__!*



1, 2, 4, 7, 12, 20, … ?

Min. #nodes of AVL Tree of height h:

$\#T(0)+1 = \boldsymbol{F}_3$, $\#T(h+1)+1 = \#T(h)+1 + \#T(h-1)+1 = \boldsymbol{F}_{h+4}$

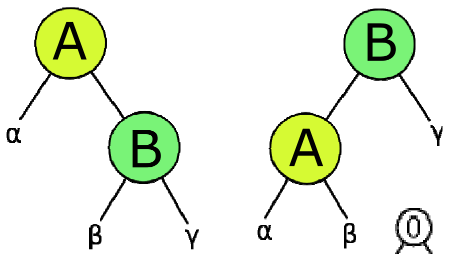
with Fibonacci no.s $F_h = (\Phi^h - (-1/\Phi)^h)/\sqrt{5} \geq \Omega(1.6^h)$

by induction as $\Phi := (1+\sqrt{5})/2 \approx 1.618$ has $\Phi^2 = \Phi + 1$.

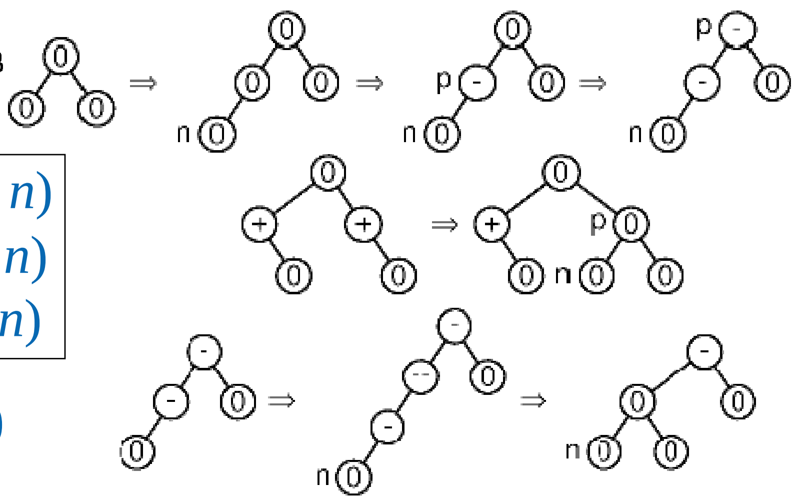
AVL Tree Maintenance

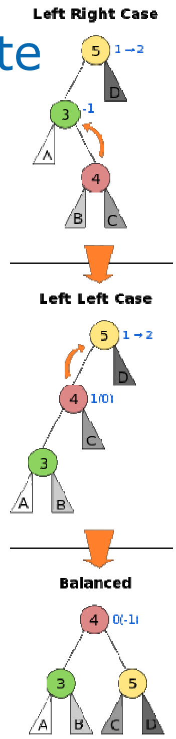
Adelson-Velsky & Landis 1962: $h \leq O(\log n)$

Heights of any two sibling subtrees __must__ differ by *__at most one__!*



Store and recursively update balance indicators **+, 0, -** during insert, three cases:





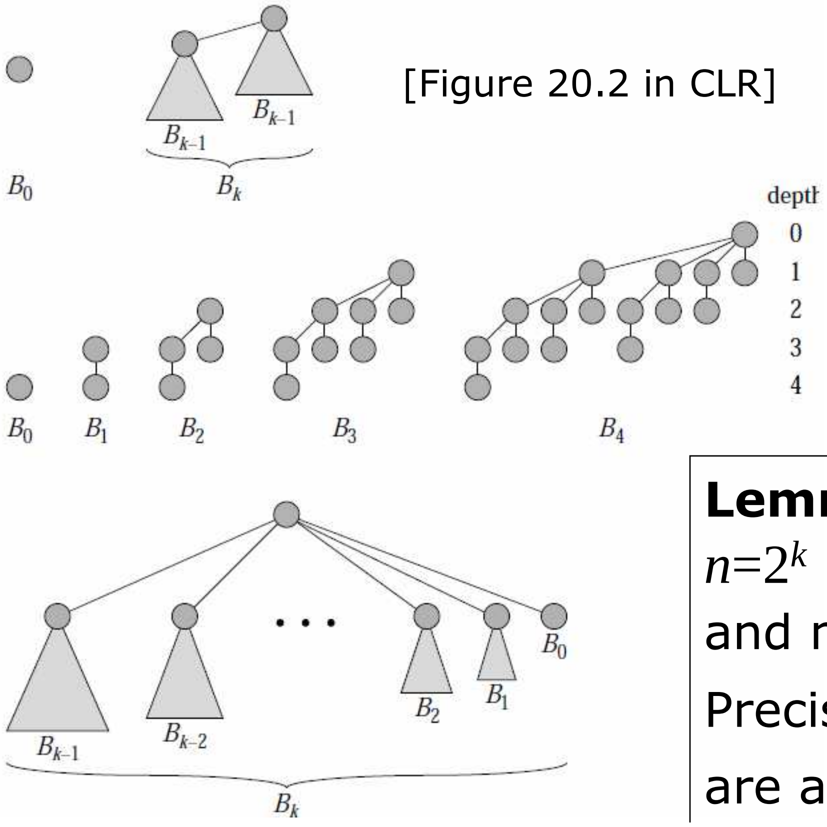
search $O(\log n)$ insert $O(\log n)$ delete $O(\log n)$

merge $O(n)$

Binomial Trees [CLR,§20]

A *binomial tree* is an ordered tree defined recursively:

$B_k + B_k \rightarrow B_{k+1}$ Merge



[Figure 20.2 in CLR]

Require and maintain each B to be *heap-ordered*: key(node) ≤ key(children)

Lemma 20.1: B_k has $n=2^k$ nodes and height k and maximum degree k. Precisely $\binom{k}{d}$ nodes are at depth d.

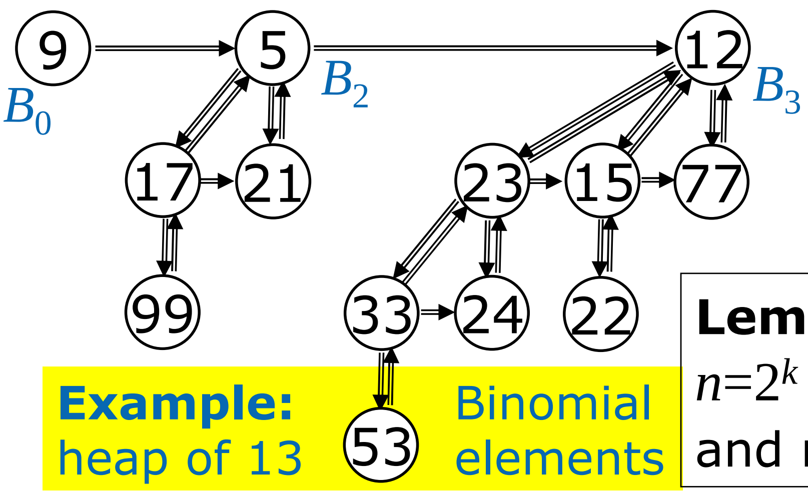
Binomial Heaps [CLR,§20]

A *binomial tree* is an ordered tree defined recursively.

Binomial heap is ascend. list of binomial trees containing, for each k, at most one B_k.

Merge

Require and maintain each B to be *heap-ordered*: key(node) ≤ key(children)



Example: heap of 13 Binomial elements

Lemma 20.1: B_k has $n=2^k$ nodes and height k and maximum degree k.

Pointers to: children, parent, right sibbling, next bin. tree

List len.+deg. ≤ $O(\log n)$

Operations on Binomial Heaps

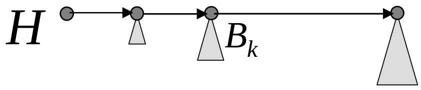
A *binomial tree* is an ordered tree defined recursively.

Binomial heap is ascend. list of binomial trees containing, for each k, at most one B_k.

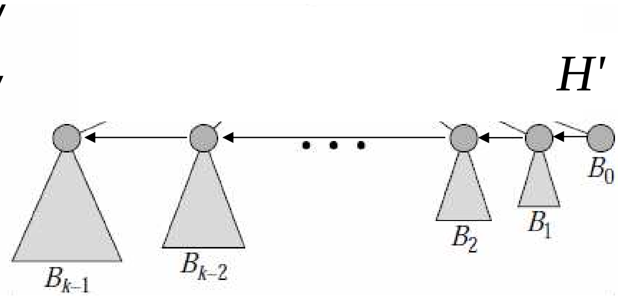
Merge

Require and maintain each B to be *heap-ordered*: key(node) ≤ key(children)

Operations:



1. Create one-elem. bin.heap: $O(1)$ √
2. Extract the minimum key: $O(\log n)$ √
3. Merge two binom. heaps: $O(\log n)$
4. Insert element: $O(\log n)$ √
5. Decrease key: $O(\log n)$ √
6. Delete key: $O(\log n)$ √

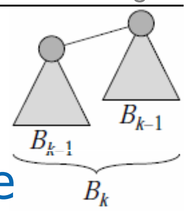


Pointers to: children, parent, right sibbling, next bin. tree

List len.+deg. ≤ $O(\log n)$

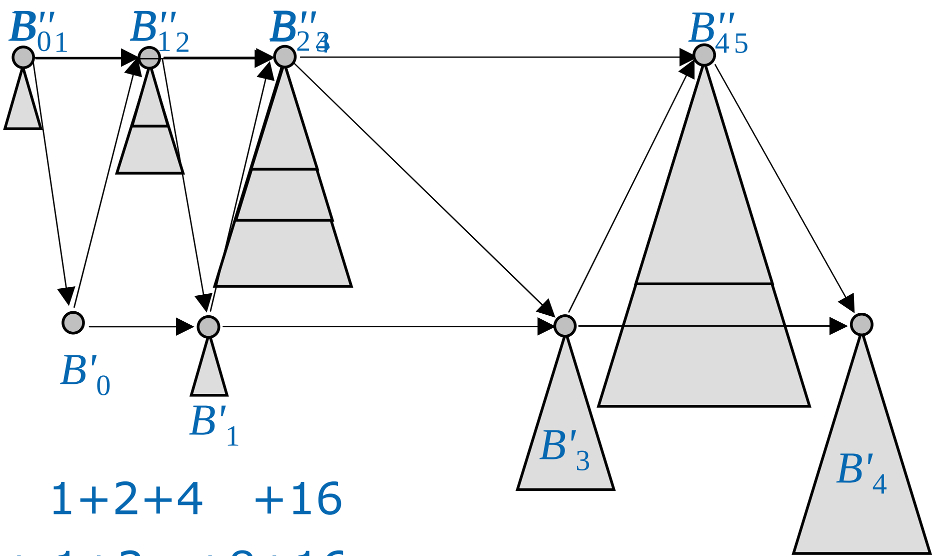
Merging two Binomial Heaps

- Merging two Binomial Trees B_k: $O(1)$



Binomial heap is ascend. list of binomial trees containing, for each k, at most one B_k.

Require and maintain each B to be *heap-ordered*: key(node) ≤ key(children)



Merging two Binomial Heaps in $O(\log n)$ √

1+2+4 +16
+ 1+2 +8+16
= 2 +16+32 √

List len.+deg. ≤ $O(\log n)$