§2 Data Structures

KAIST
CS500 M. Ziegler

Array:

Heap:

Stack:

List:

Queue:

Trees:

binary

Red-Black, AVL

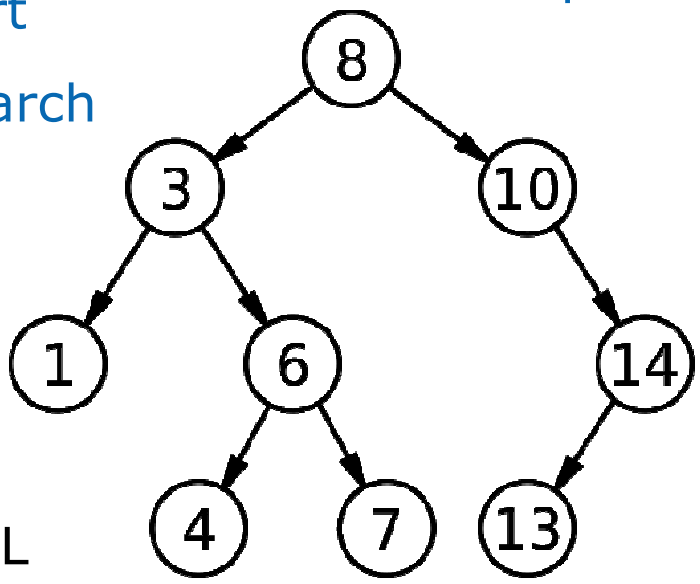
2-3, B

access

sort

search

model of computation?



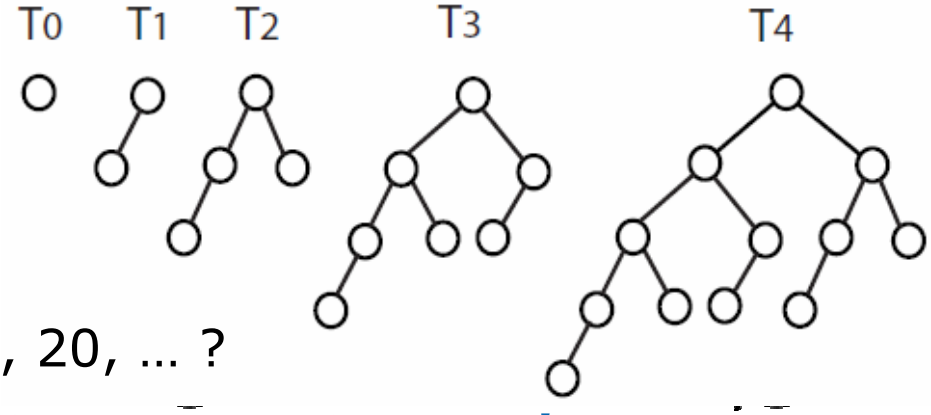
search, insert, delete: $O(\log n)$

§2 Recap: AVL Trees

KAIST
CS500 M. Ziegler

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

Heights of any two sibling subtrees must differ by at most one!



Fibonacci

$F_0=0, F_1=1,$

$F_{h+1} = F_h+F_{h-1}$

1, 2, 4, 7, 12, 20, … ?

Min. #nodes of AVL Tree of height h:

$\#T(0)+1 = F_3$, $\#T(h+1)+1 = \#T(h)+1 + \#T(h-1)+1 = F_{h+4}$

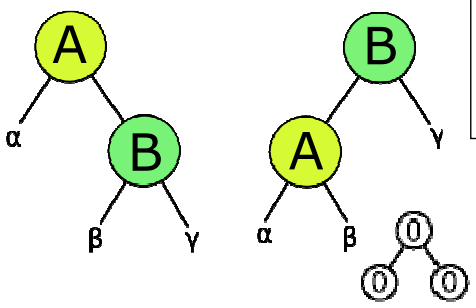
with Fibonacci no.s $F_h = (\varphi^h-(-1/\varphi)^h)/\sqrt{5} \geq \Omega(1.6^h)$

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

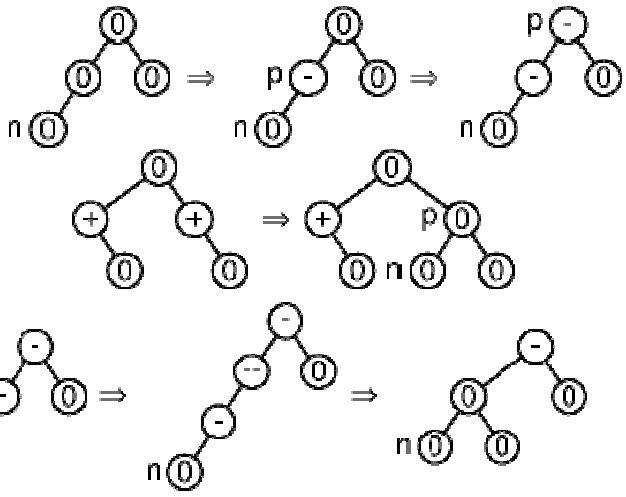
AVL Tree Maintenance

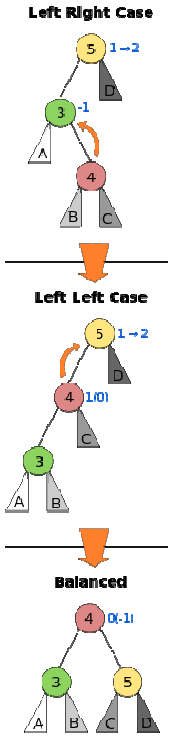
Adelson-Velsky & Landis 1962: $h \leq O(\log n)$

Heights of any two sibling subtrees must differ by at most one!



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





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

merge $O(n)$

Mar. 14: Offline Quiz

KAIST
CS500 M. Ziegler

Recall the *Fibonacci Numbers*

$$F_0=0, \quad F_1=1, \quad F_{h+1} = F_h+F_{h-1}$$

Abbreviate $\varphi := (1+\sqrt{5})/2$ the *Golden Ratio*.

Prove: a) $\varphi^2 = \varphi+1$

b) $\varphi-1 = 1/\varphi$

c) $\varphi^2+1 = \varphi\cdot\sqrt{5}$

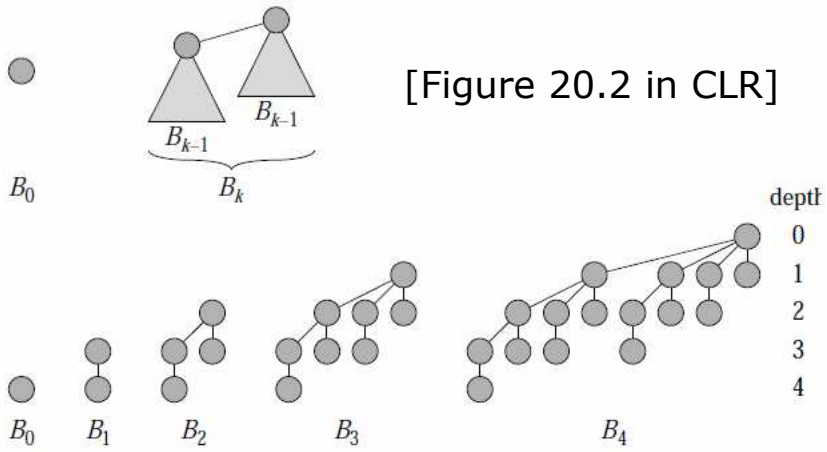
d) $F_h = (\varphi^h-(-1/\varphi)^h) \,/\, \sqrt{5}$

Binomial Trees [CLR,§20]

KAIST
CS500 M. Ziegler

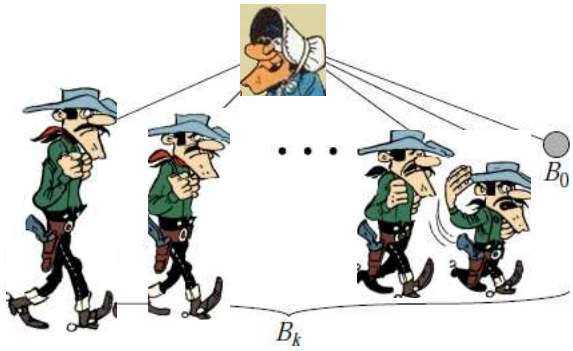
A *binomial tree* is an ordered tree defined recursively:

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

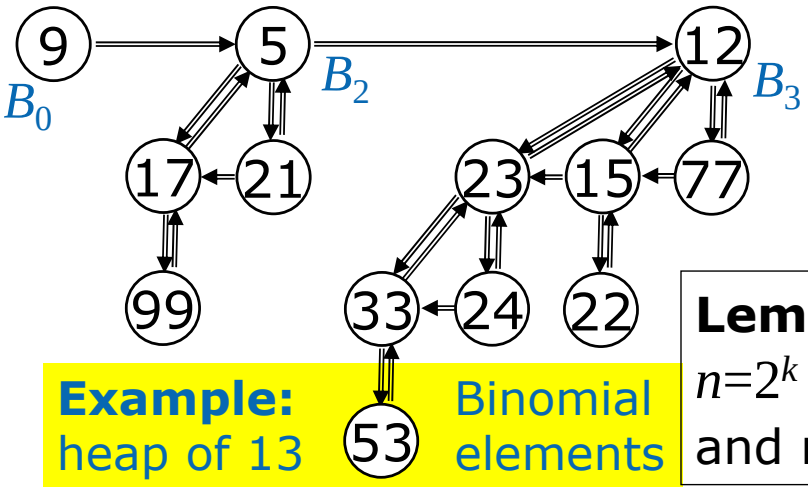
KAIST
CS500 M. Ziegler

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, left sibbling, next binm. tree

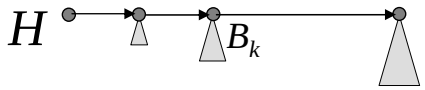
List len.+deg. ≤ $O(\log n)$

Operations on Binomial Heaps

KAIST CS500 M. Ziegler

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.



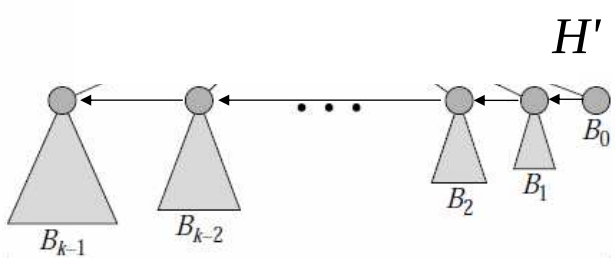
Operations:

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

Merge

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

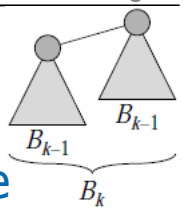


List len.+deg. ≤ $O(\log n)$

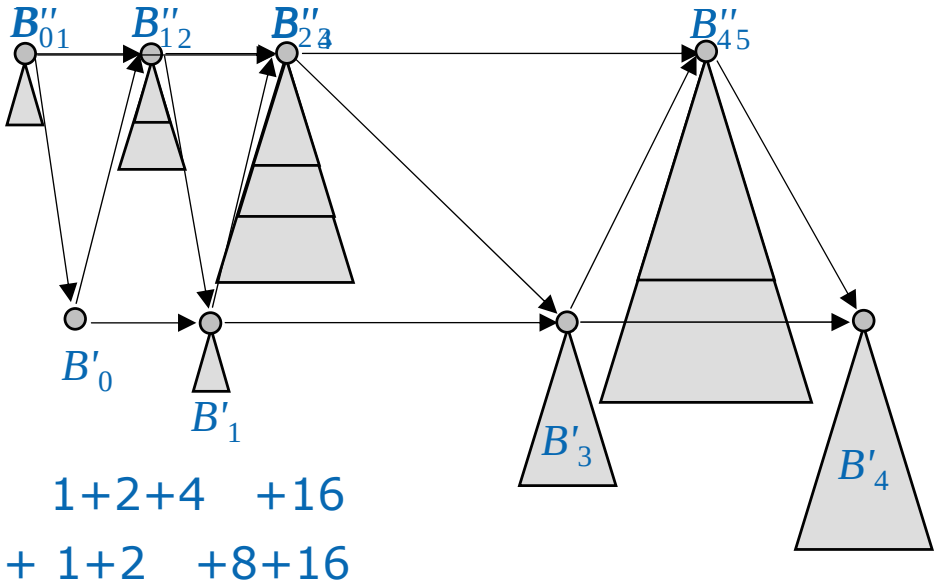
Merging two Binomial Heaps

KAIST CS500 M. Ziegler

- 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.



1+2+4 +16
+ 1+2 +8+16
= 2 +16+32 √

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

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

List len.+deg. ≤ $O(\log n)$