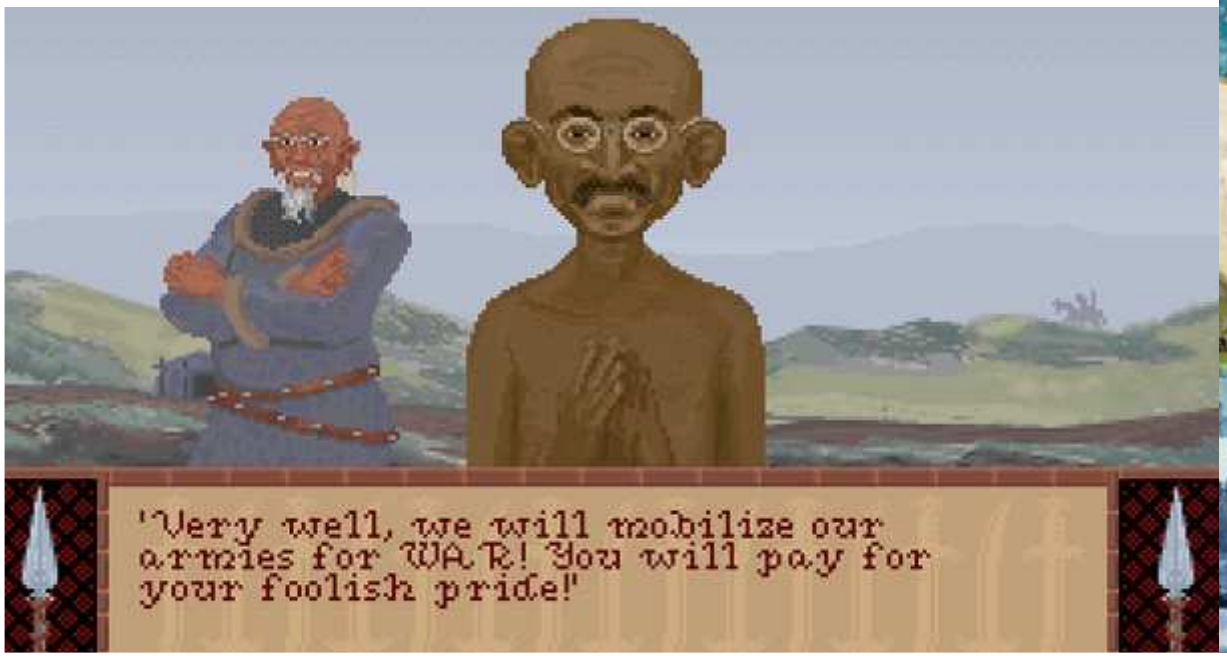
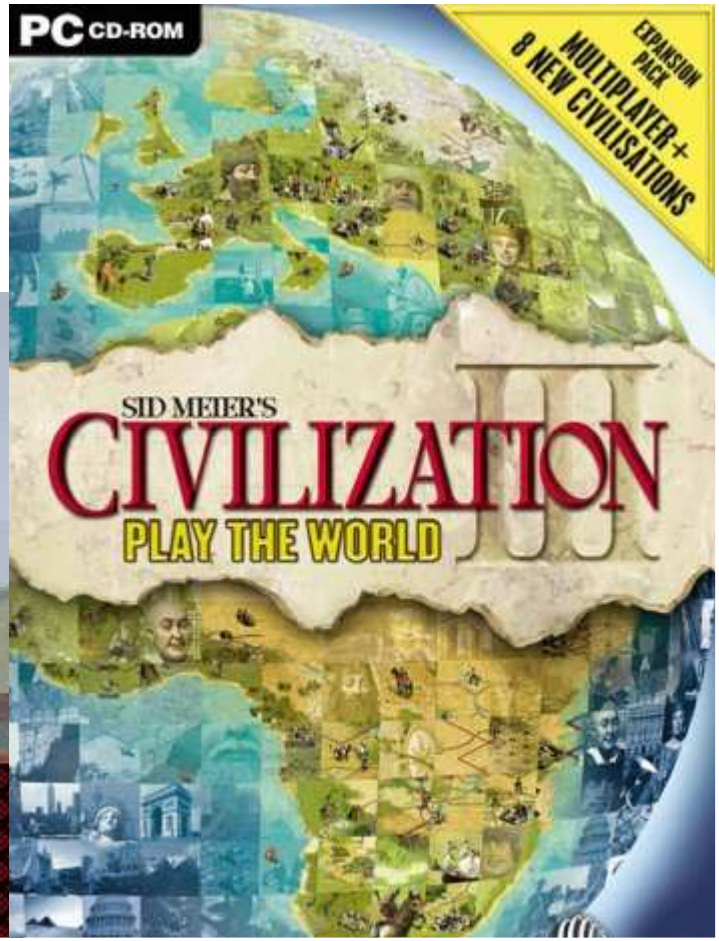
§3 Tree Data Structures

- Abstract Data Types
  - Hide hardware/implementation/data structure
  - Recap: basic / derived/ linked data structures
- AVL Trees:
  - definition, properties,
  - operations/maintenance, deficiency
- Binomial Trees, Binomial Heaps:
  - definition, operations, analysis
  - ExtractMin, DecreaseKey, Merge

Hardware vs. Math. Data Types

- Each country/leader described by "A.I." character parameters.
- Initially *Gandhi*.**aggression** = 1
- When country adopts democracy, **`aggression -= 2`**.





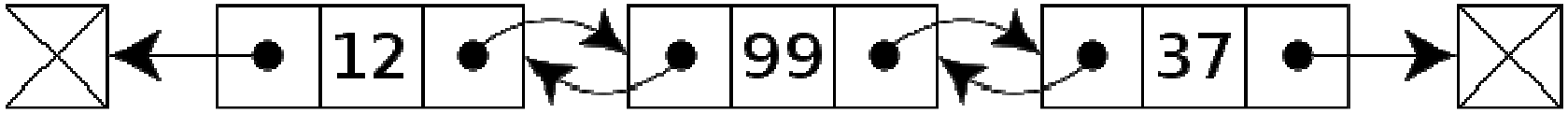
Abstract Data Types

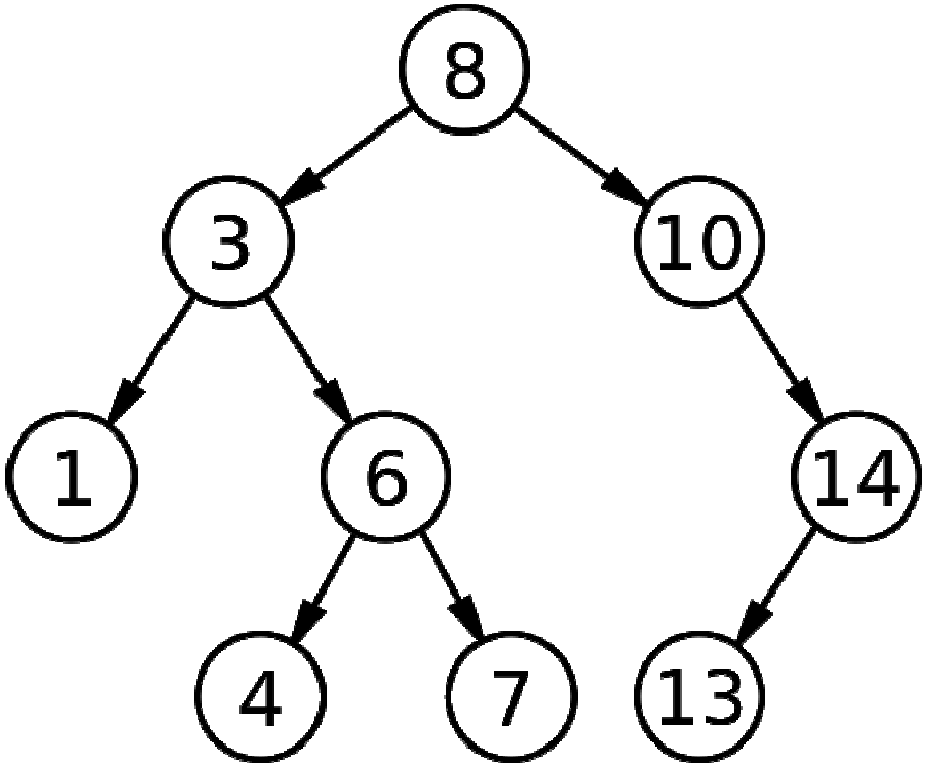
Integer $\mathbb{Z}$ (vs. byte/word etc.)	$0, 1, +, -, \times,$ div $, >$
Real $\mathbb{R}$ (vs. float/double etc.)	$0, 1, +, -, \times, \div, >$
Stack of X	push x, pop, isEmpty
Queue of X	enque x, deque, isEmpty
(Dynamic) array of X	[], size, (re-size), search
Sorted array of Y	search y , insert y , delete y
Priority queue of Y	findmin, insert y

where Y is totally ordered

$O(1)$ $O(n)$ $O(\log n)$ $O(n)$ $O(\log n)$

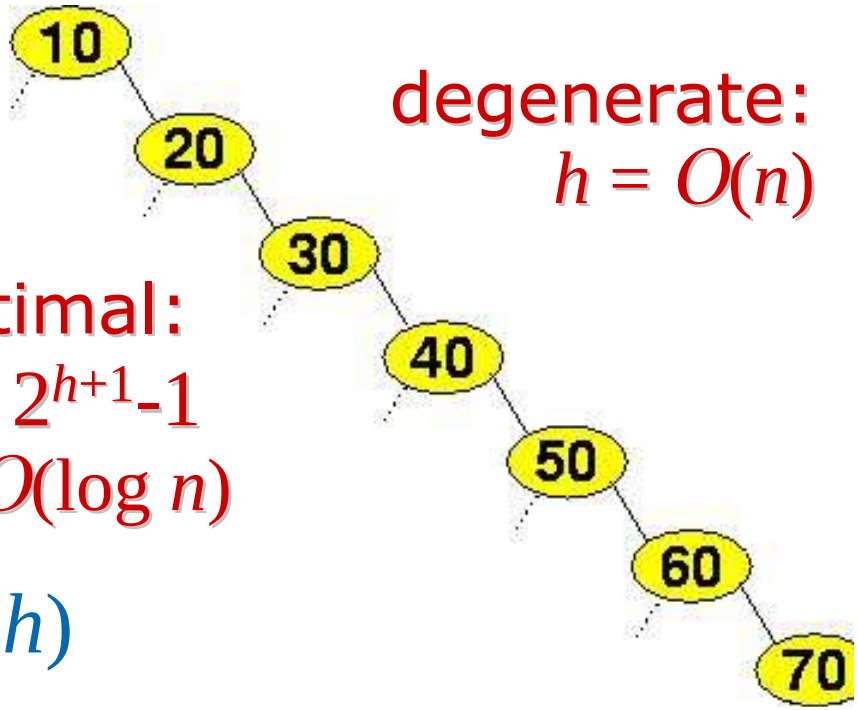
Linked Data Structures





(Doubly) linked list:

- insert, delete in $O(1)$
- search in $O(n)$



degenerate:
$h = O(n)$

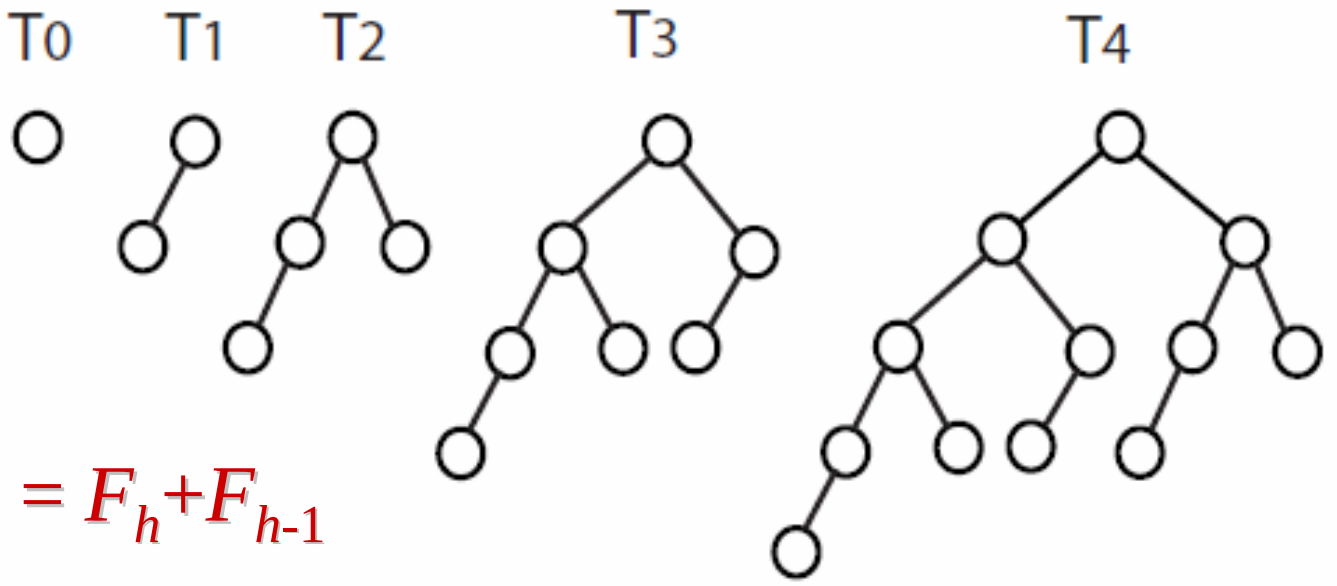
optimal:
$n = 2^{h+1}-1$
$h = O(\log n)$

Binary search tree:
search, insert, delete in $O(h)$

Adelson-Velsky-Landis'62

Binary tree s.t. any two sibling subtrees have height difference at most 1!

minimal AVLTrees:



Fibonacci

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

$n(h)$:=min #nodes of AVLTree of height $h \le \mathrm{O}(\log n)$

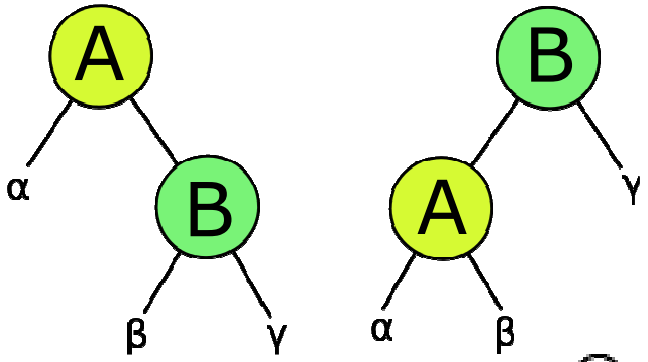
$\#n(0)+1=F_3,\quad \#n(h+1)+1 = \#n(h)+1 + \#n(h-1)+1 = F_{h+4}$

Recall $F_h = (\varphi^h-(-1/\varphi)^h)/\sqrt{5} \ge \Omega(1.6^h) \Rightarrow h=\mathrm{O}(\log F_h)$

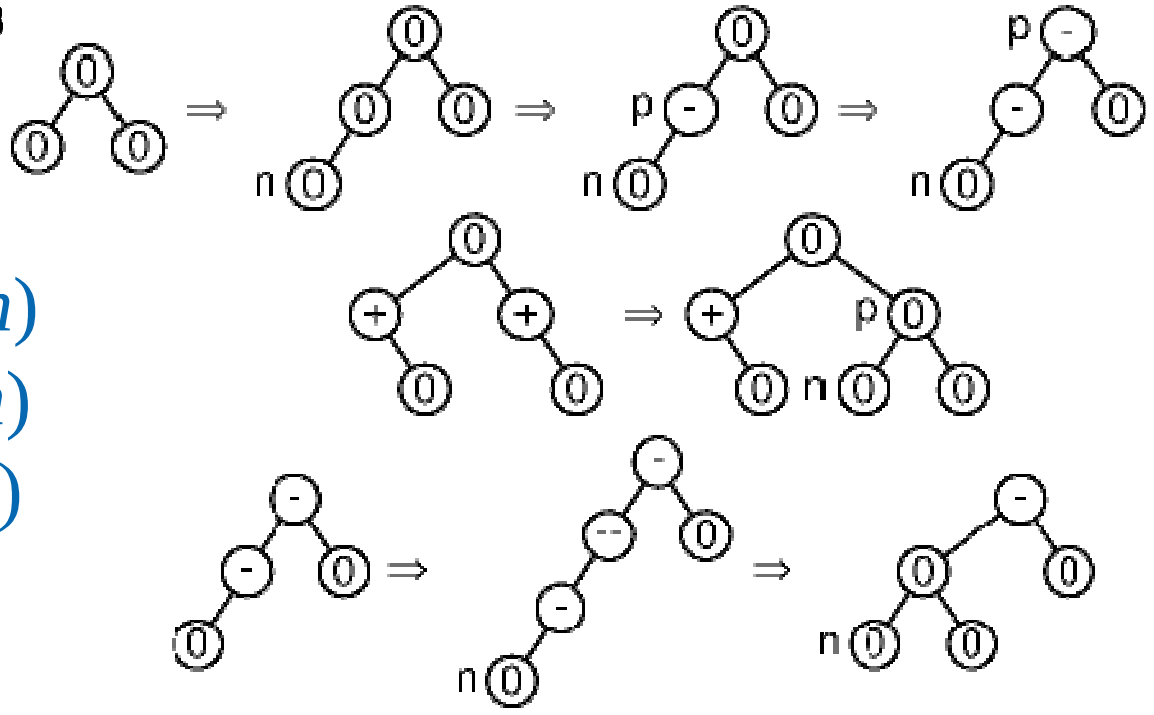
AVL Tree Maintenance

Binary tree s.t. any two sibling subtrees have height difference at most 1!

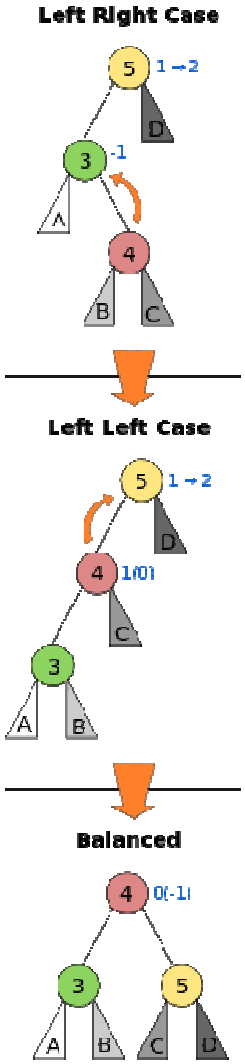
$$h \le \mathrm{O}(\log n)$$



Store & recursively update balance indicators **+, 0, -.**
After **insert** at a left leaf, propagate up: three cases

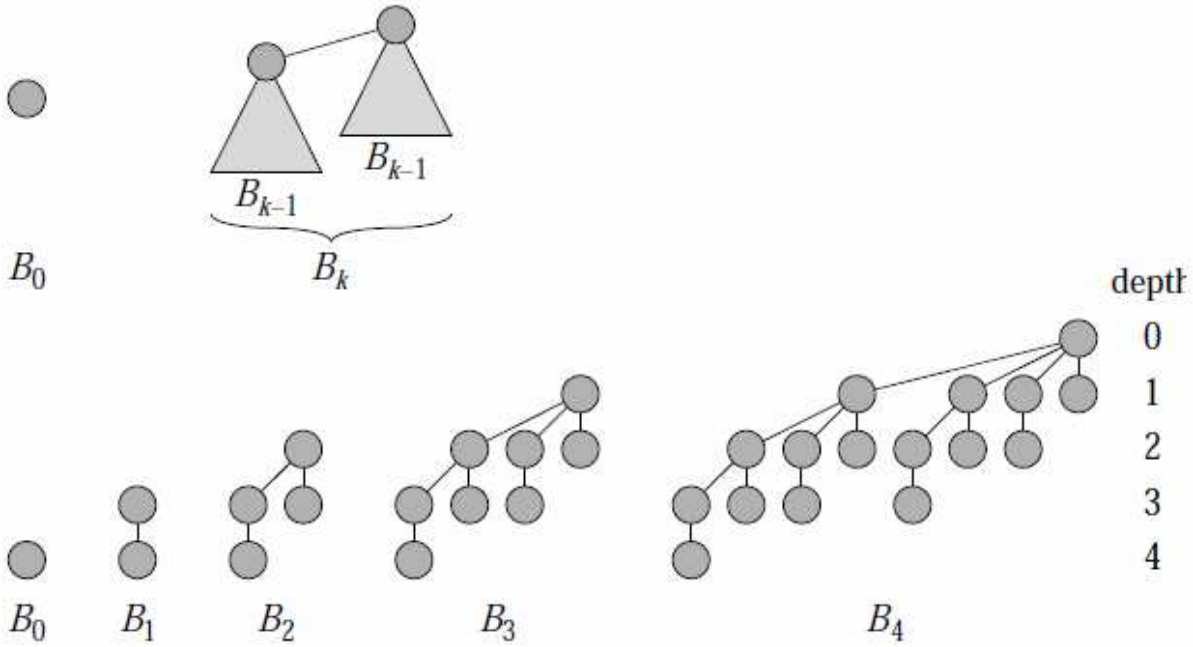


search $O(\log n)$
insert $O(\log n)$
delete $O(\log n)$
merge $O(n)$



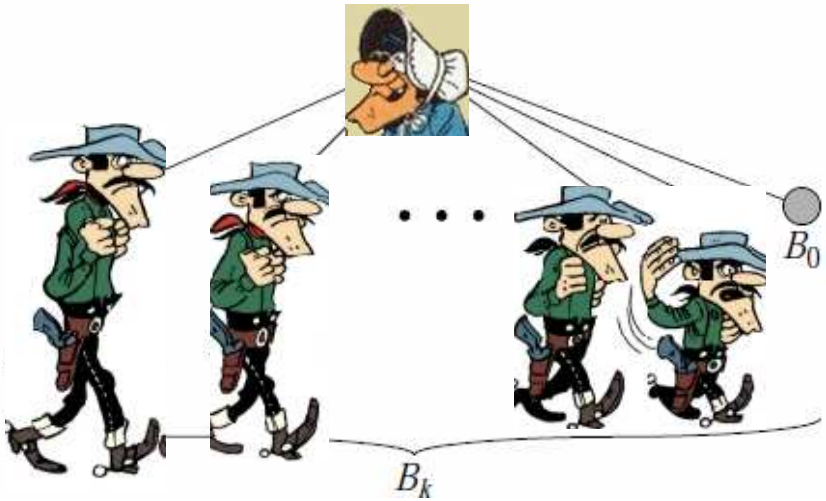
Binomial Trees

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



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

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

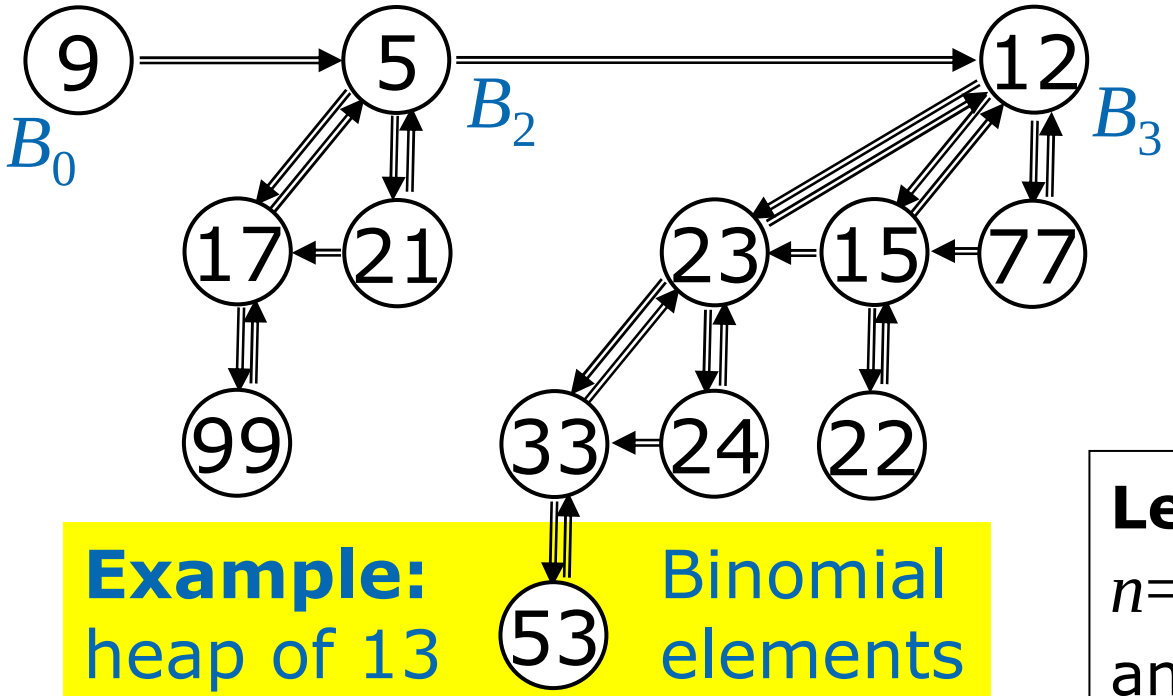


Lemma: B_k has $n=2^k$ nodes and height k and maximum degree k.
Precisely $\binom{k}{d}$ nodes are at depth d.

Binomial Heaps

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.



Example: heap of 13 Binomial elements

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

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

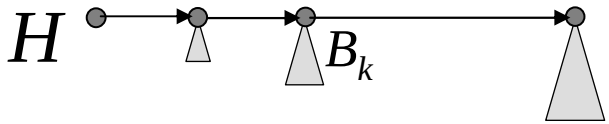
Pointers to: children, parent, left sibbling, next binm. tree

List length, vertex degree, tree depth: all ≤ $O(\log n)$

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.



Operations:

1. *Create* one-elem. bin.heap: $O(1)$
2. *Extractmin*(imum): $O(\log n)$
3. *Merge* two binom. heaps: $O(\log n)$
4. *Insert* element: $O(\log n)$
5. *DecreaseKey*: $O(\log n)$
6. *Delete*: $O(\log n)$

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

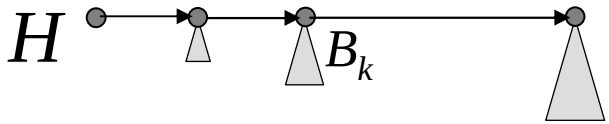
no *search* operation: entries via reference/link/"handle"

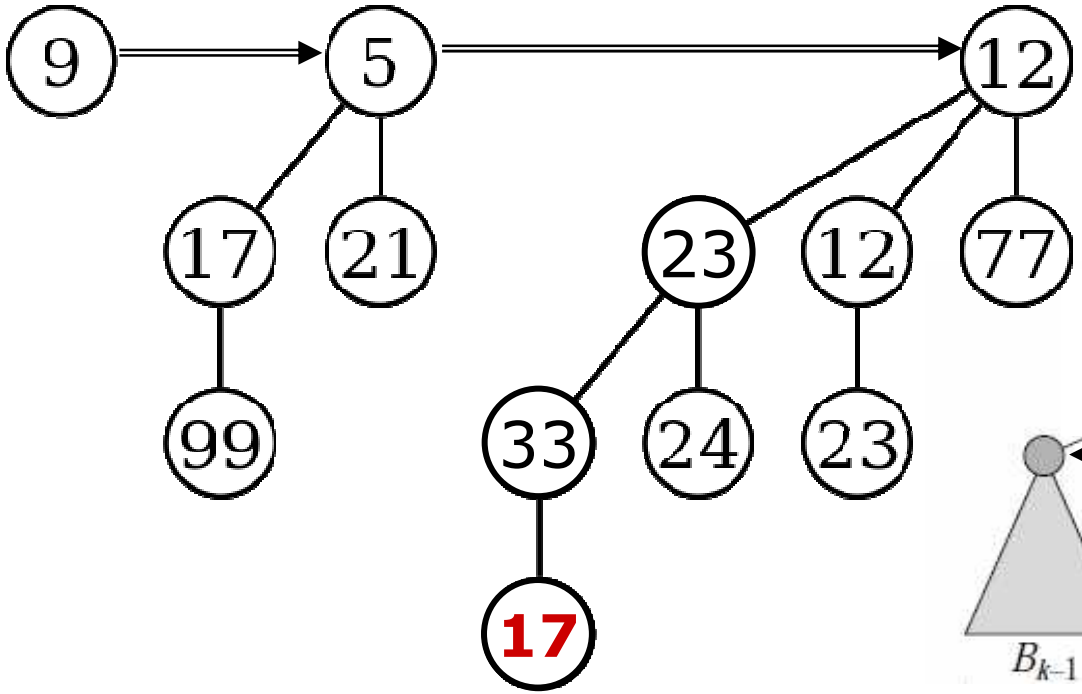
List length, vertex degree, tree depth: all ≤ $O(\log n)$

ExtractMin and DecreaseKey

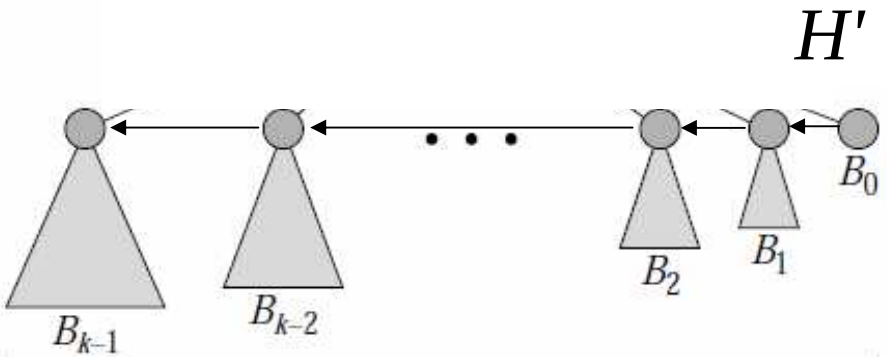
A *binomial tree* is an ordered tree defined recursively.

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





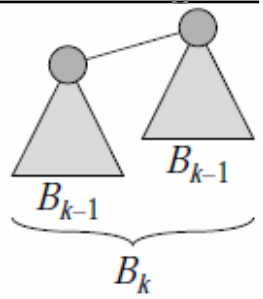
Require and maintain each B to be *heap-ordered*:
key(node) ≤ key(children)



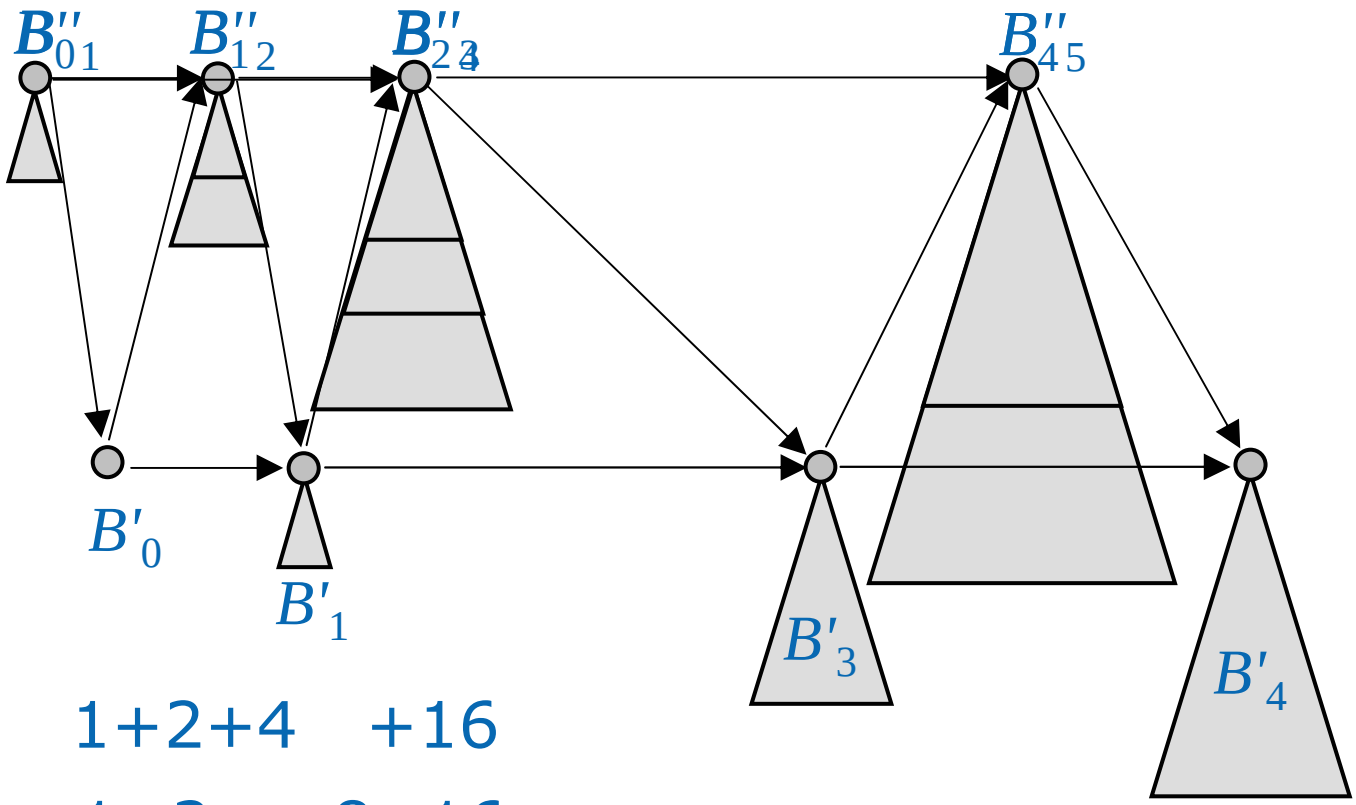
List length, vertex degree, tree depth: all ≤ $O(\log n)$

Merging two Binomial Heaps

- Merging two Binomial *Trees* B_k: cost 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)

1+2+4 +16
+ 1+2 +8+16
= 2 +16+32√

List length, vertex degree, tree depth: all ≤ O(log n)

§3 Recap

- Abstract Data Types
  - Hide hardware/implementation/data structure
  - Recap: basic / derived/ linked data structures
- AVL Trees:
  - definition, properties,
  - operations/maintenance, deficiency
- Binomial Trees, Binomial Heaps:
  - definition, operations, analysis
  - ExtractMin, DecreaseKey, Merge