Binomial Heap

Binomial Heap - Definition

A binomial heap of n-elements is a collection of binomial trees with the following properties:

- **Each binomial tree is heap-ordered (parent is less than all children)**
- **No two binomial trees in the collection have the same size**
- **Number of trees will be O(lg n)**

Key idea: Union in O(lg n) time

Comparison of Efficiency

Procedure	Binary (worst-case)	Binomial (worst-case)
Make-Heap	$\Theta(1)$	$\Theta(1)$
Insert	$\Theta(\lg n)$	$O(\lg n)$
Minimum	$\Theta(1)$	$O(\lg n)$
Extract-Min	$\Theta(\lg n)$	$\Theta(\lg n)$
Union	$\Theta(n)$	$O(\lg n)$
Decrease-Key	$\Theta(\lg n)$	$\Theta(\lg n)$
Delete	$\Theta(\lg n)$	$\Theta(\lg n)$

Procedure	Binary heap (worst-case)	Binomial heap (worst-case)	Fibonacci heap (amortized)
MAKE-HEAP	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
INSERT	$\Theta(\lg n)$	$O(\lg n)$	$\Theta(1)$
MINIMUM	$\Theta(1)$	$O(\lg n)$	$\Theta(1)$
EXTRACT-MIN	$\Theta(\lg n)$	$\Theta(\lg n)$	$O(\lg n)$
UNION	$\Theta(n)$	$O(\lg n)$	$\Theta(1)$
DECREASE-KEY	$\Theta(\lg n)$	$\Theta(\lg n)$	$\Theta(1)$
DELETE	$\Theta(\lg n)$	$\Theta(\lg n)$	$O(\lg n)$

Figure 19.1 Running times for operations on three implementations of mergeable heaps. The number of items in the heap(s) at the time of an operation is denoted by n.

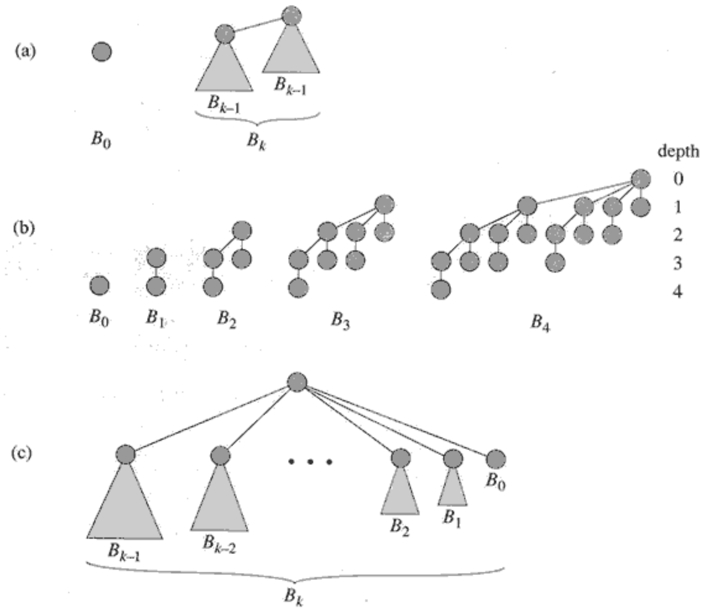


Figure 19.2 **(a)** The recursive definition of the binomial tree B_k. Triangles represent rooted subtrees. **(b)** The binomial trees B_0 through B_4. Node depths in B_4 are shown. **(c)** Another way of looking at the binomial tree B_k.

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

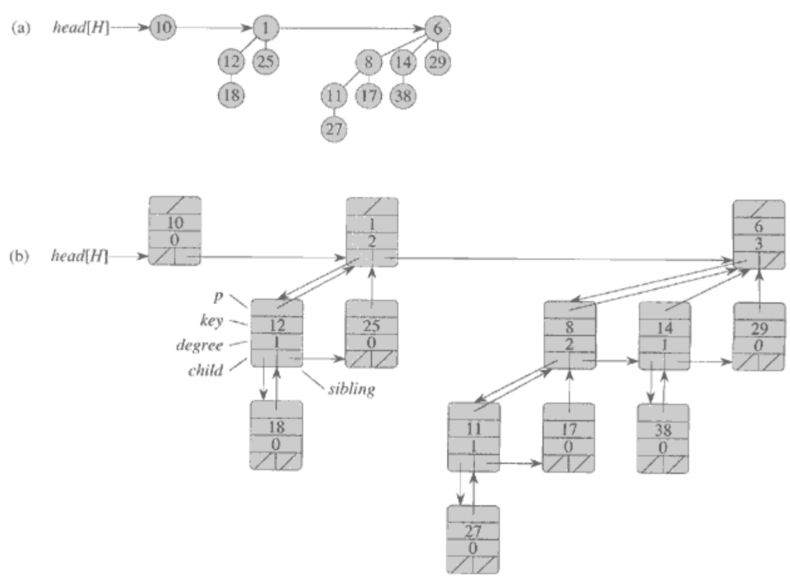
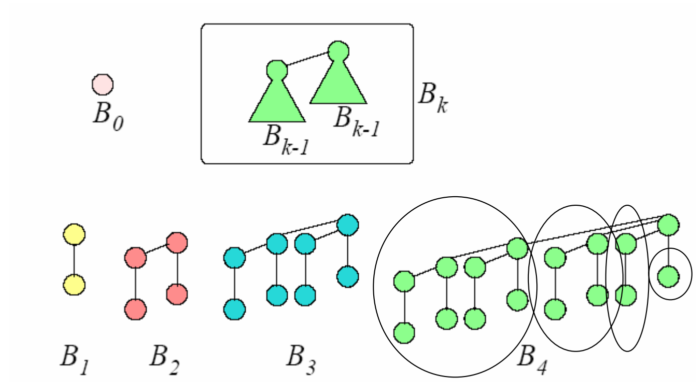


Figure 19.3 A binomial heap H with $n = 13$ nodes. **(a)** The heap consists of binomial trees B_0, B_2, and B_3, which have 1, 4, and 8 nodes respectively, totaling $n = 13$ nodes. Since each binomial tree is min-heap-ordered, the key of any node is no less than the key of its parent. Also shown is the root list, which is a linked list of roots in order of increasing degree. **(b)** A more detailed representation of binomial heap H. Each binomial tree is stored in the left-child, right-sibling representation, and each node stores its degree.

Binomial Trees

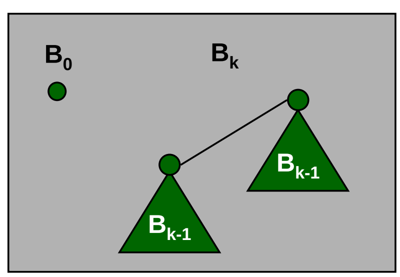


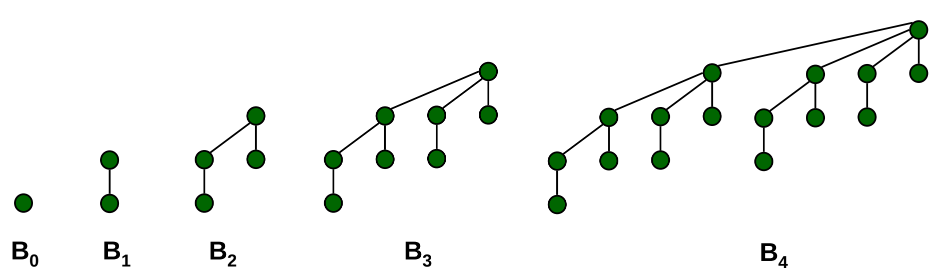
Tree B_k has 2^k nodes.
B_k has height k.
Children of the root of B_k are B_{k-1}, B_{k-2}, …, B_0 from left to right.
Max degree of an n-node binomial tree is lg n.

Binomial Tree

Binomial tree.

- **Recursive definition:**

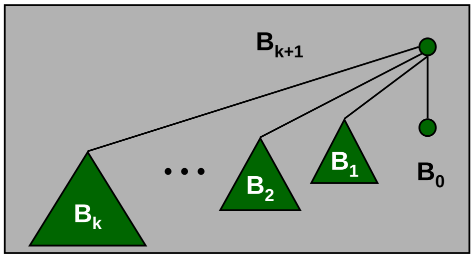




Binomial Tree

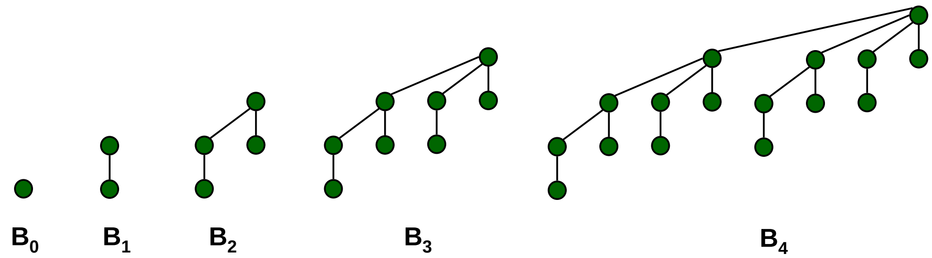
Useful properties of order k binomial tree B_k.

- Number of nodes = 2^k.
- Height = k.
- Degree of root = k.
- Deleting root yields binomial trees B_{k-1}, … , B_0.



Proof.

- By induction on k.

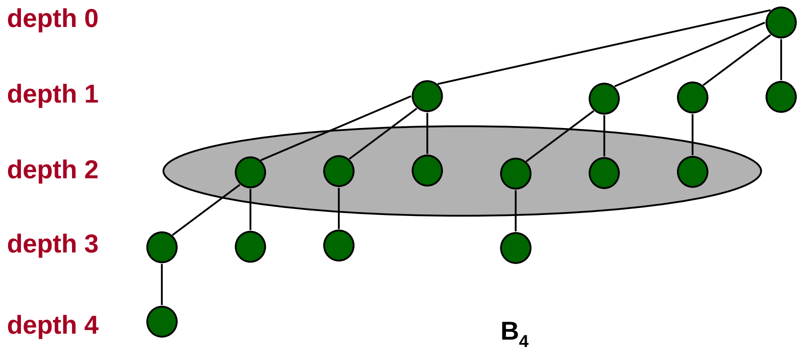


Binomial Tree

A property useful for naming the data structure.

- B_k has $\binom{k}{i}$ nodes at depth i.

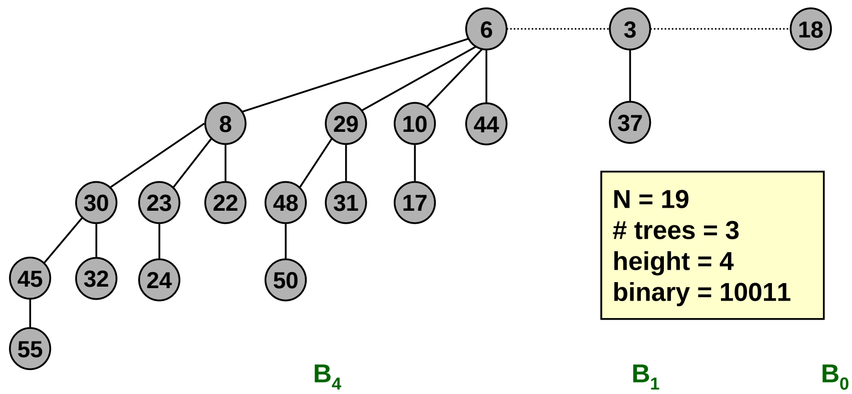
$\binom{4}{2} = 6$



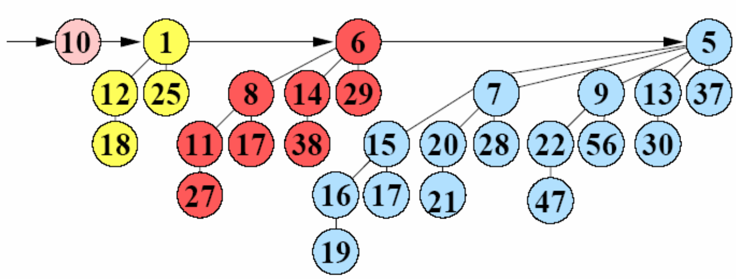
Binomial Heap: Properties

Properties of N-node binomial heap.

- **Min key contained in root of $B_0, B_1, \ldots, B_k$.**
- **Contains binomial tree B_i iff $b_i = 1$ where $b_n \cdot b_2 b_1 b_0$ is binary representation of N.**
- **At most $\lfloor \log_2 N \rfloor + 1$ binomial trees.**
- **Height $\leq \lfloor \log_2 N \rfloor$.**



Example Binomial Heap

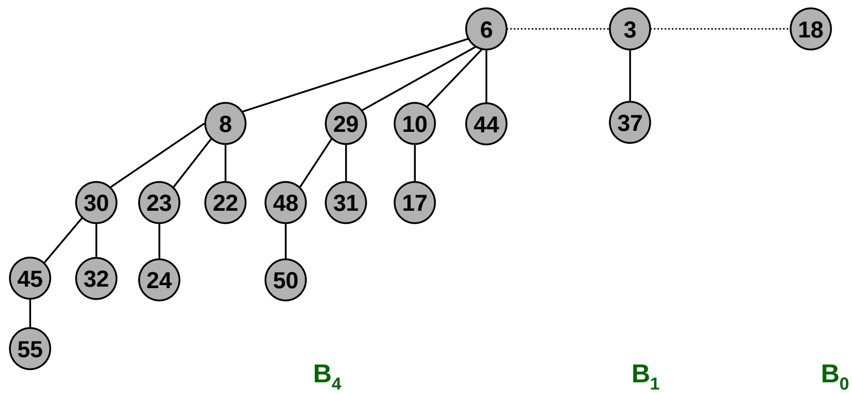


Binomial heap of 29 elements
29 = 11101 in binary.

Binomial Heap

Binomial heap. Vuillemin, 1978.

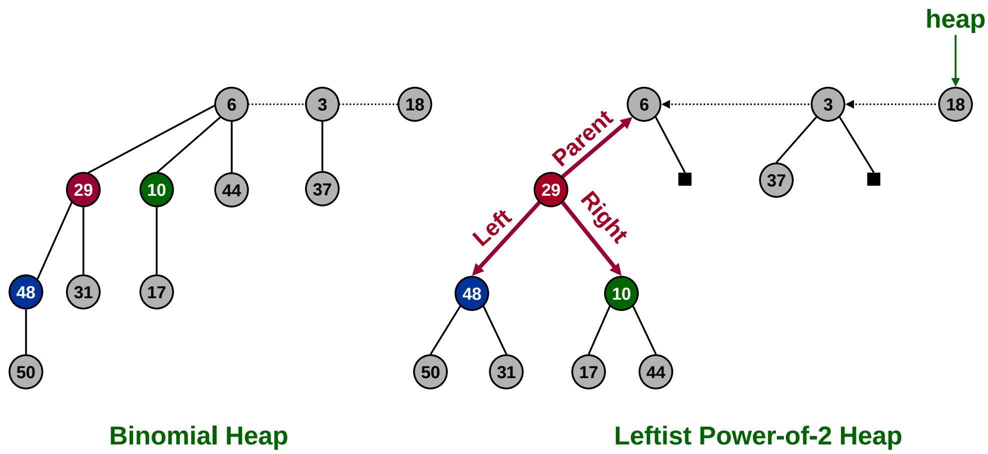
- Sequence of binomial trees that satisfy binomial heap property.
  - each tree is min-heap ordered
  - 0 or 1 binomial tree of order k



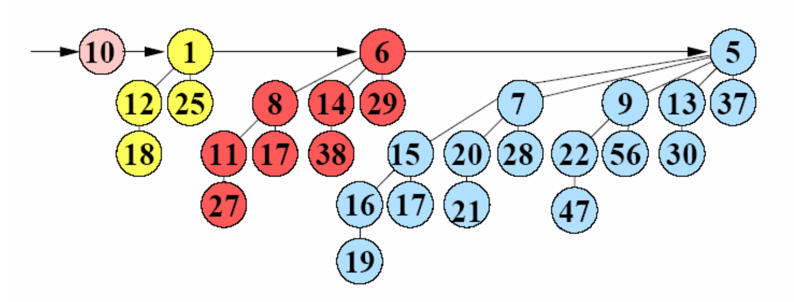
Binomial Heap: Implementation

Implementation.

- Represent trees using left-child, right sibling pointers.
  - three links per node (parent, left, right)
- Roots of trees connected with singly linked list.
  - degrees of trees strictly decreasing from left to right

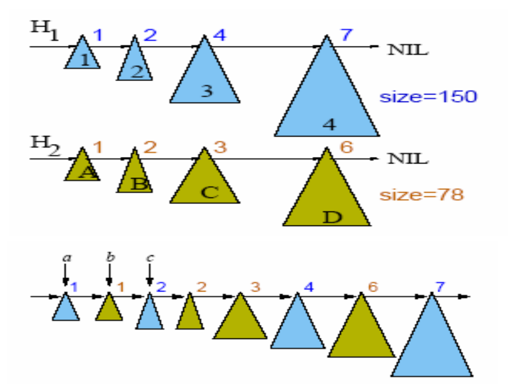


Minimum Operation



Where does the minimum have to be?
How can we find minimum in general?
Running time?

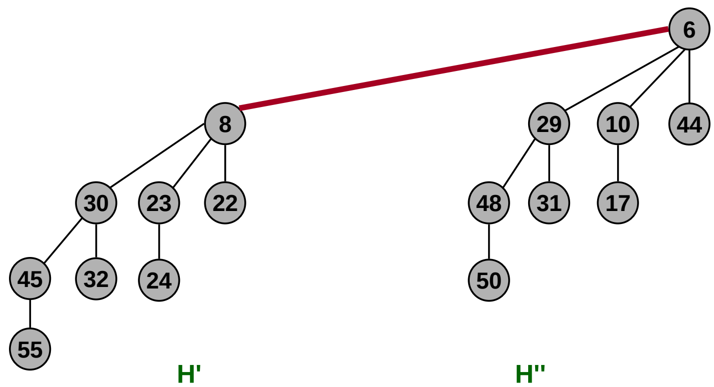
Union of 2 Binomial Heaps



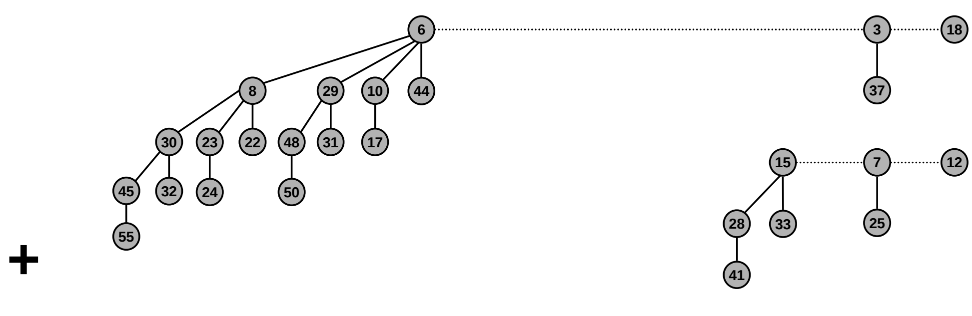
Binomial Heap: Union

Create heap H that is union of heaps H' and H".

- **"Mergeable heaps."**
- **Easy if H' and H" are each order k binomial trees.**
  - **connect roots of H' and H"**
  - **choose smaller key to be root of H**



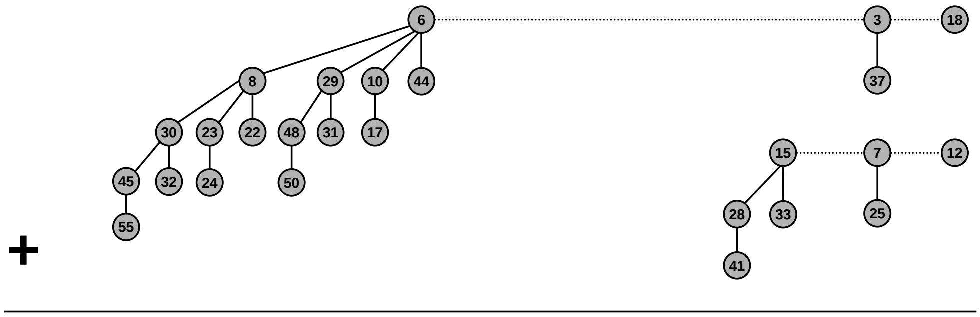
Binomial Heap: Union



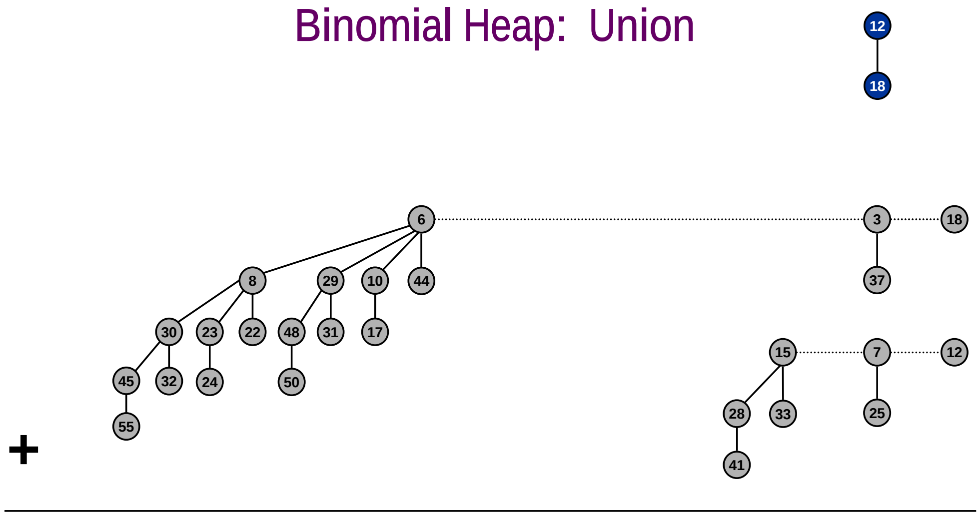
19 + 7 = 26

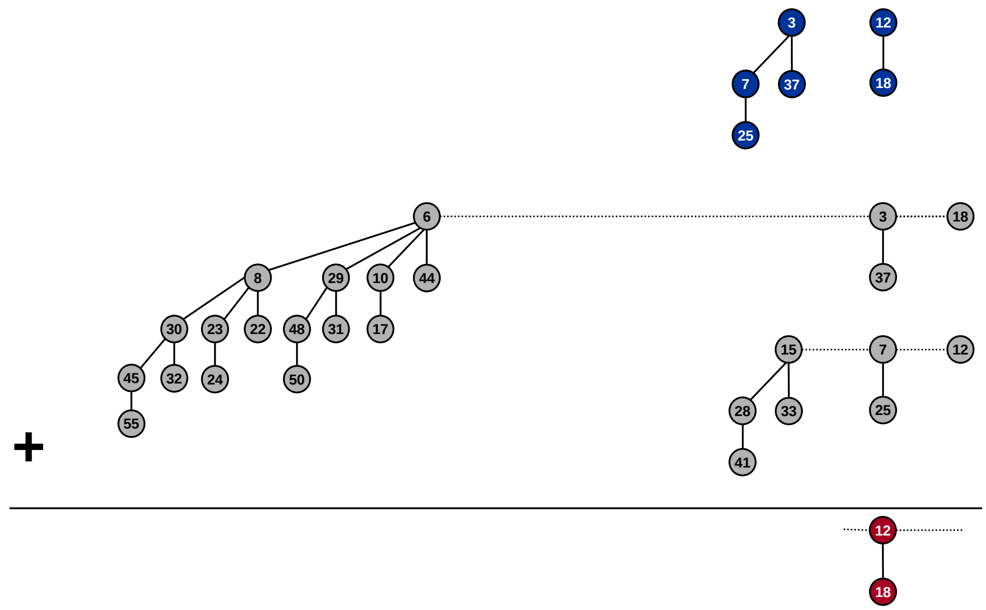
		1	1	1	
	1	0	0	1	1
+	0	0	1	1	1
	1	1	0	1	0

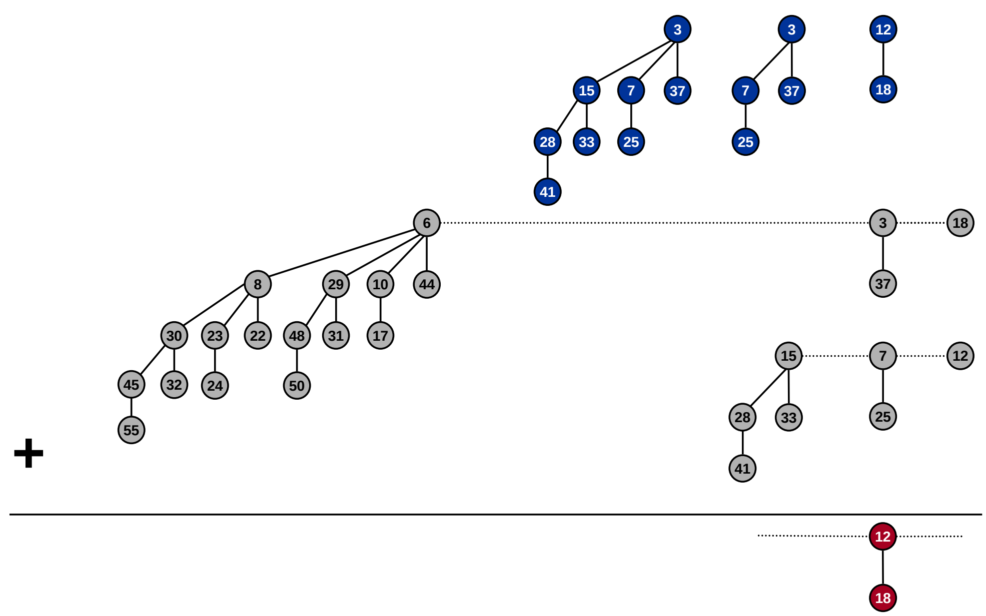
Binomial Heap: Union

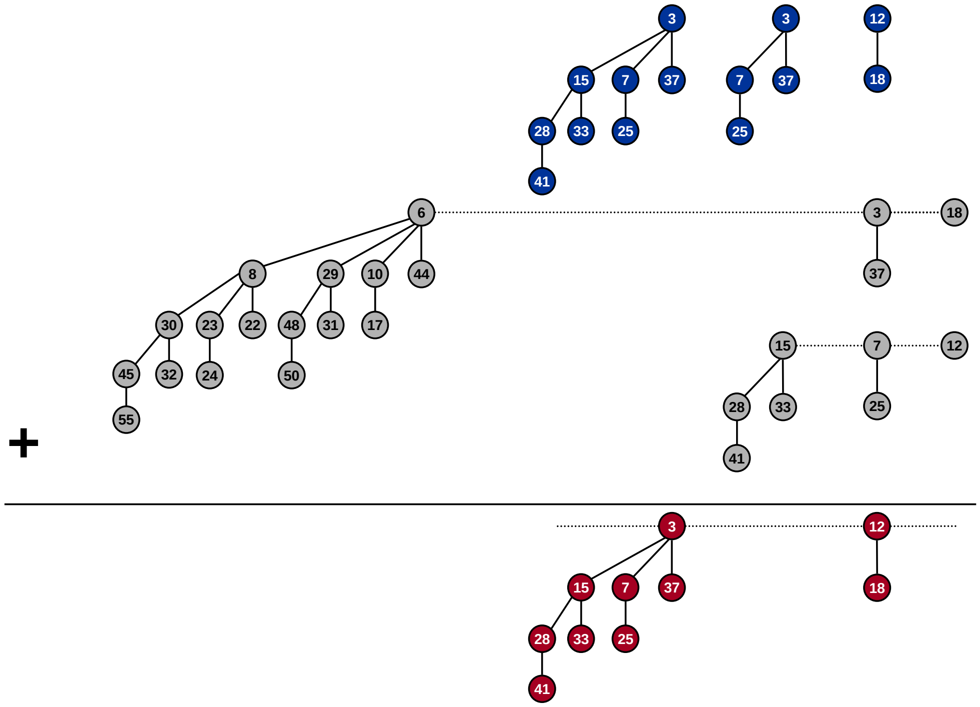


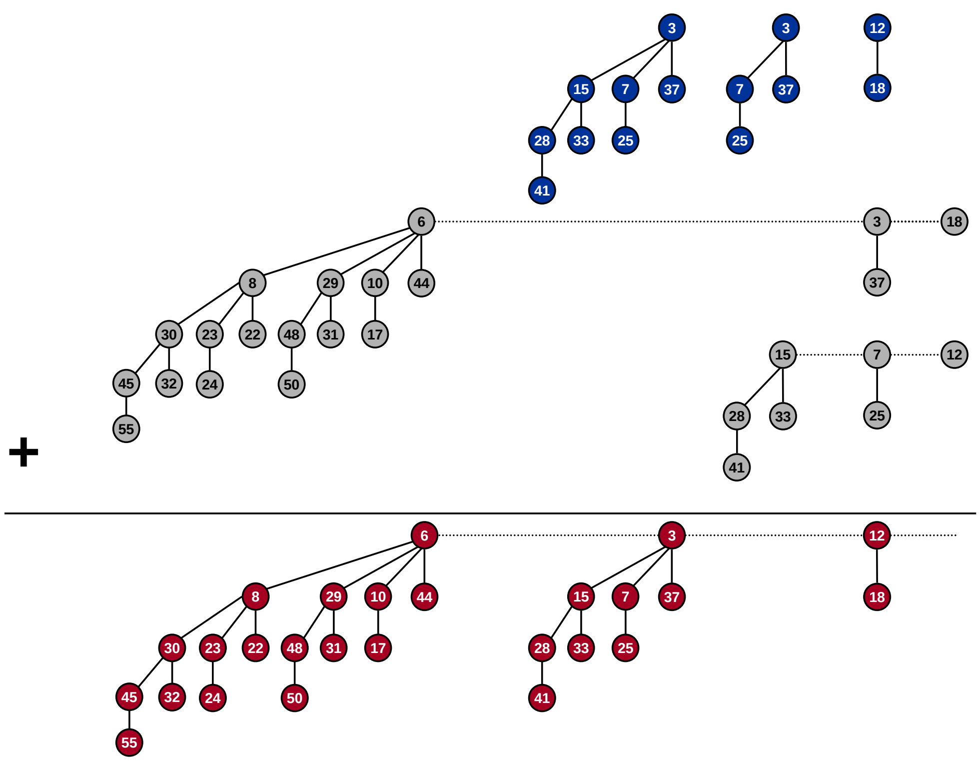
Binomial Heap: Union











Binomial Heap: Union

Create heap H that is union of heaps H' and H''.

- Analogous to binary addition.

Running time. O(log N)

- Proportional to number of trees in root lists $\leq 2(\lfloor \log_2 N \rfloor + 1)$.

19 + 7 = 26

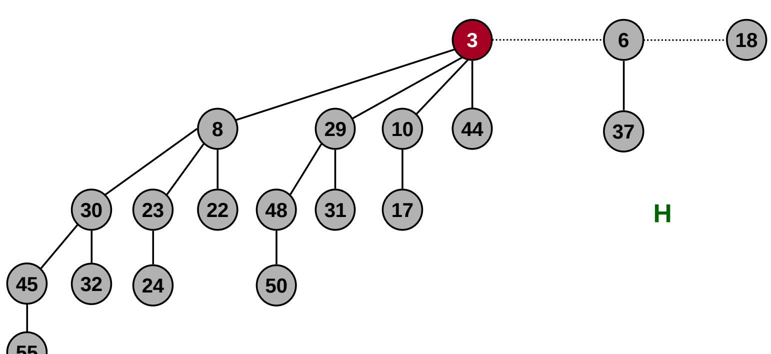
		1	1	1	
	1	0	0	1	1
+	0	0	1	1	1
	1	1	0	1	0

Binomial Heap: Delete Min

Delete node with minimum key in binomial heap H.

- Find root x with min key in root list of H, and delete
- H' ← broken binomial trees
- H ← Union(H', H)

Running time. O(log N)

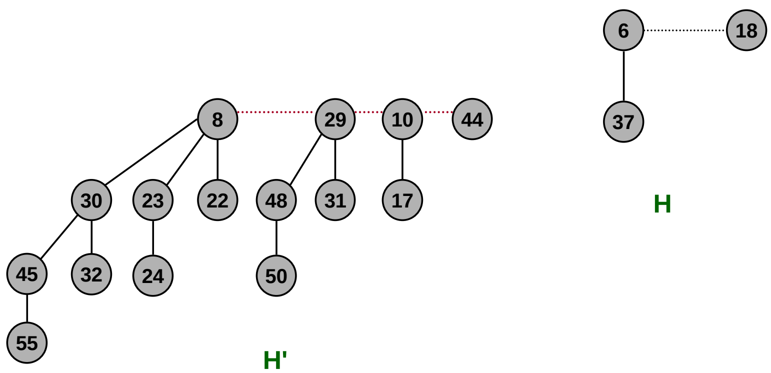


Binomial Heap: Delete Min

Delete node with minimum key in binomial heap H.

- Find root x with min key in root list of H, and delete
- H' ← broken binomial trees
- H ← Union(H', H)

Running time. O(log N)



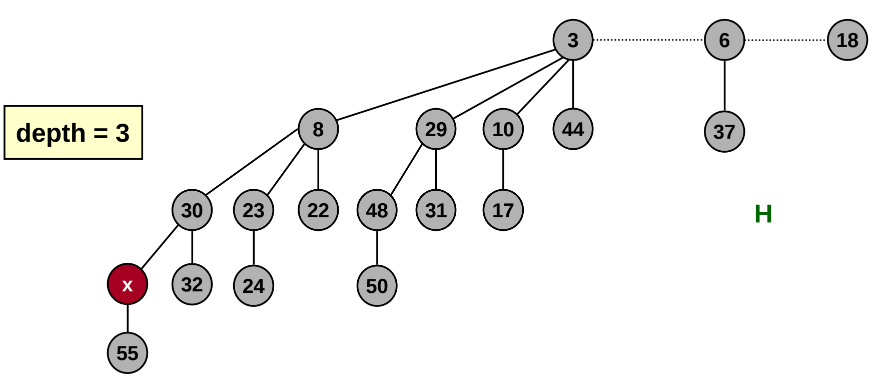
Binomial Heap: Decrease Key

Decrease key of node x in binomial heap H.

- Suppose x is in binomial tree B_k.
- Bubble node x up the tree if x is too small.

Running time. O(log N)

- Proportional to depth of node x $\leq \lfloor \log_2 N \rfloor$.



Binomial Heap: Delete

Delete node x in binomial heap H.

- Decrease key of x to $-\infty$.
- Delete min.

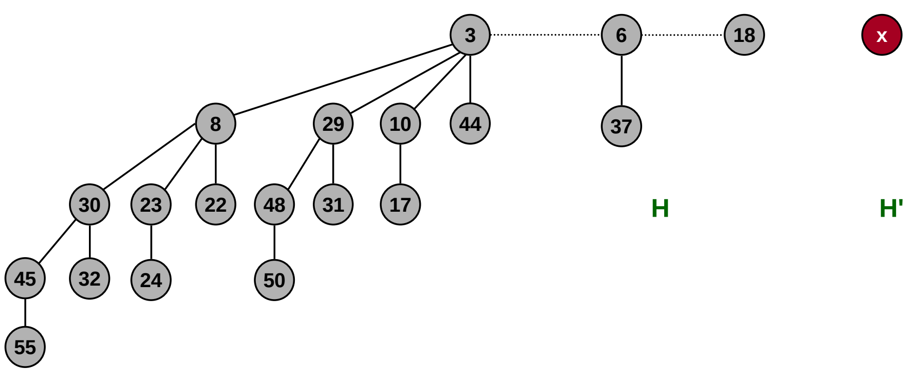
Running time. O(log N)

Binomial Heap: Insert

Insert a new node x into binomial heap H.

- H' $\leftarrow$ MakeHeap(x)
- H $\leftarrow$ Union(H', H)

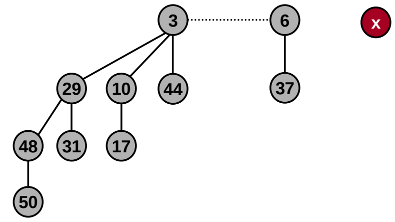
Running time. O(log N)



Binomial Heap: Sequence of Inserts

Insert a new node x into binomial heap H.

- If N = `.......0`, then only 1 steps.
- If N = `......01`, then only 2 steps.
- If N = `.....011`, then only 3 steps.
- If N = `....0111`, then only 4 steps.



Inserting 1 item can take $\Omega(\log N)$ time.

- If N = `11...111`, then $\log_2$ N steps.

But, inserting sequence of N items takes O(N) time!

- (N/2)(1) + (N/4)(2) + (N/8)(3) + . . . $\leq$ 2N
- Amortized analysis.
- Basis for getting most operations down to constant time.

$$\sum_{n=1}^{N} \frac{n}{2^n} = 2 - \frac{N}{2^N} - \frac{1}{2^{N-1}} \leq 2$$