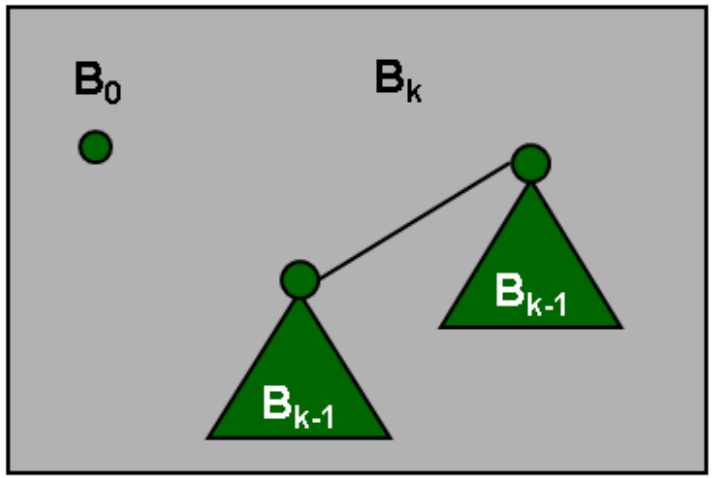
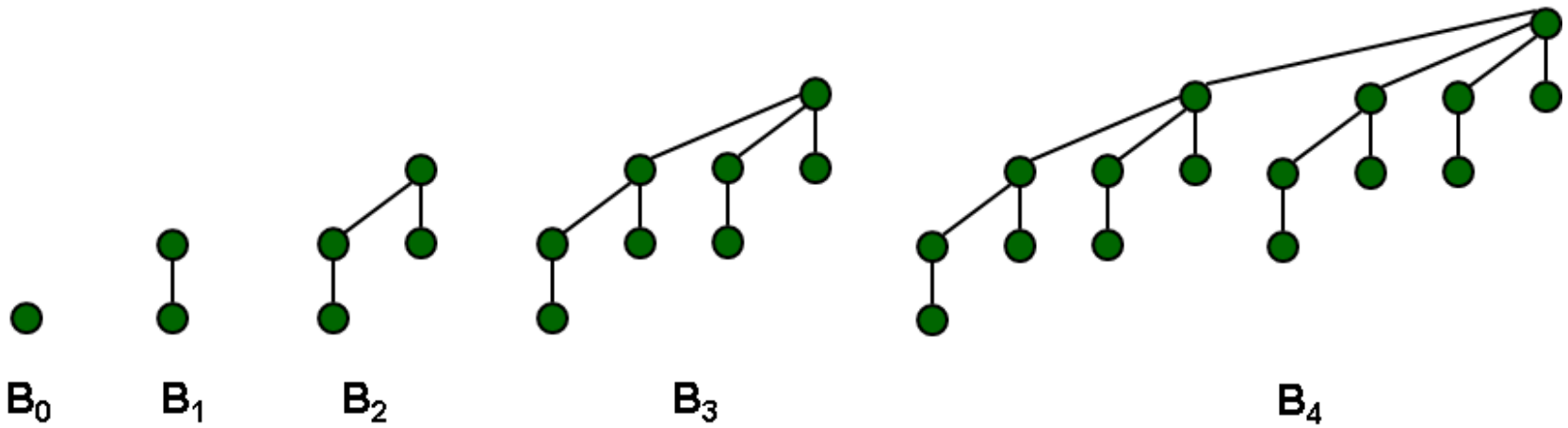
Binomial Heaps

Binomial Tree

Binomial tree.

- **Recursive definition:**

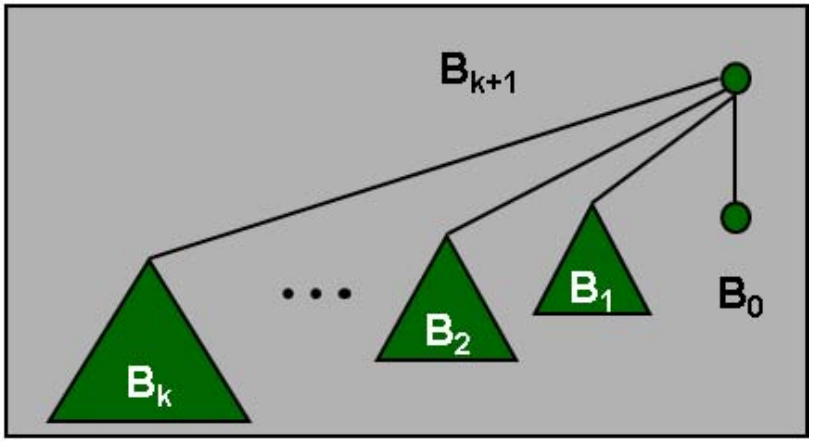




Binomial Tree

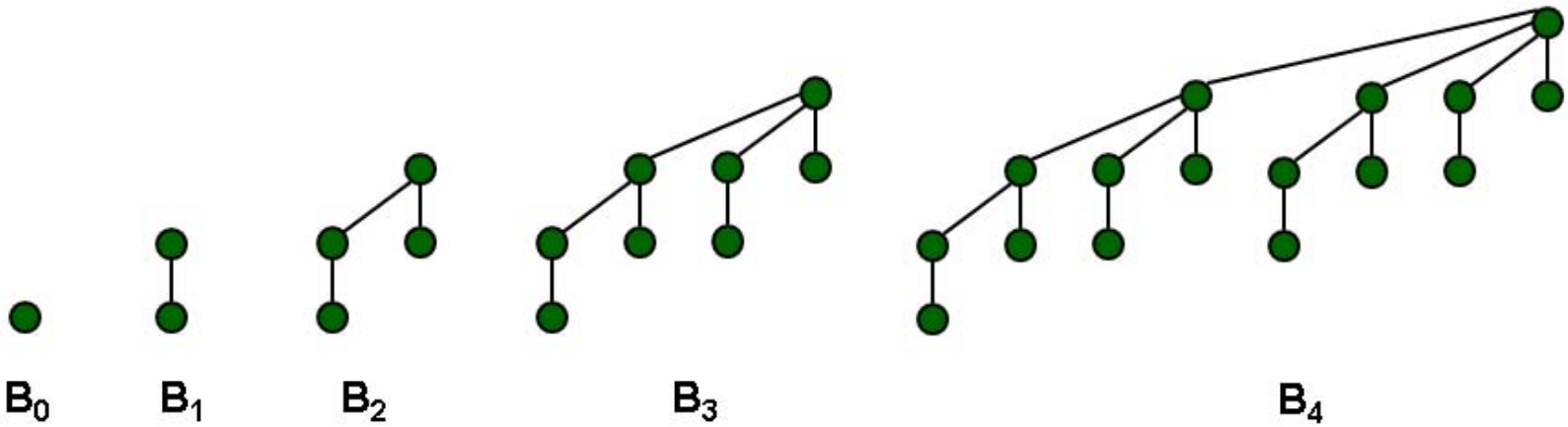
Useful properties of order k binomial tree B_k.

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



Proof.

- By induction on k.

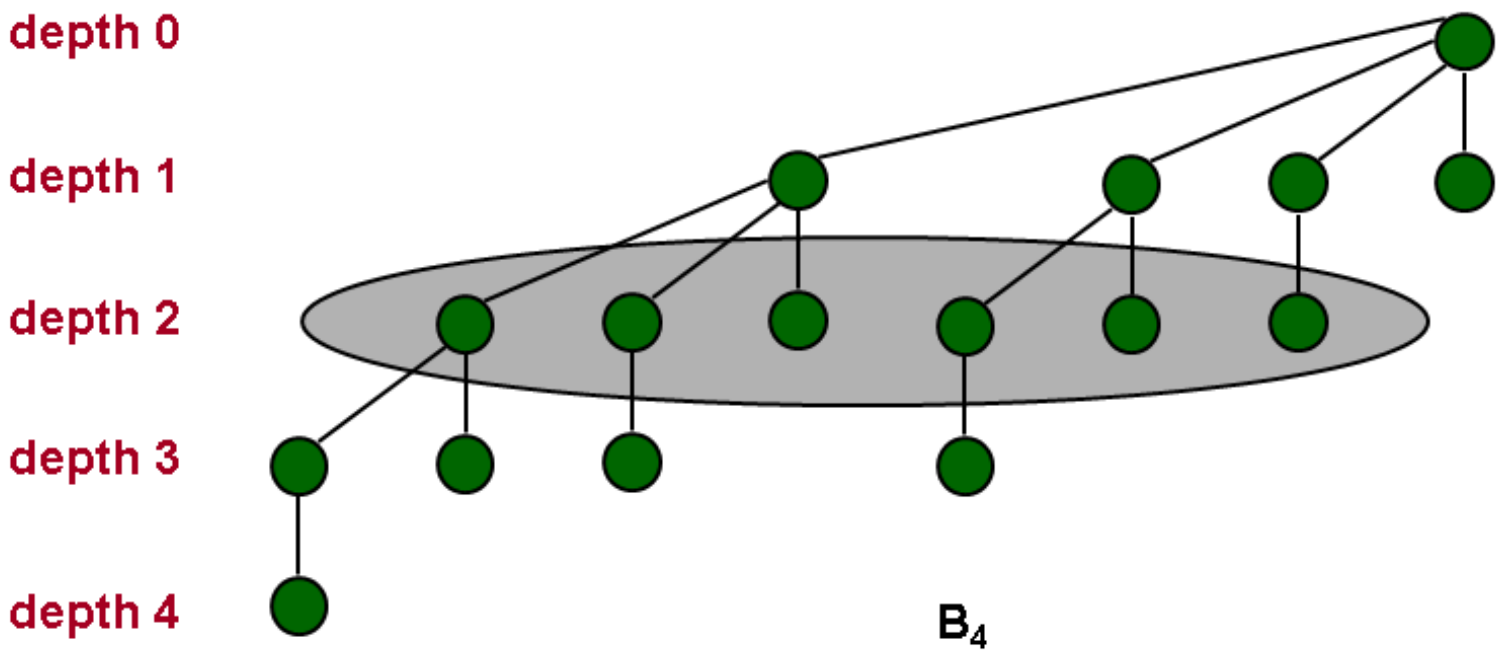


Binomial Tree

A property useful for naming the data structure.

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

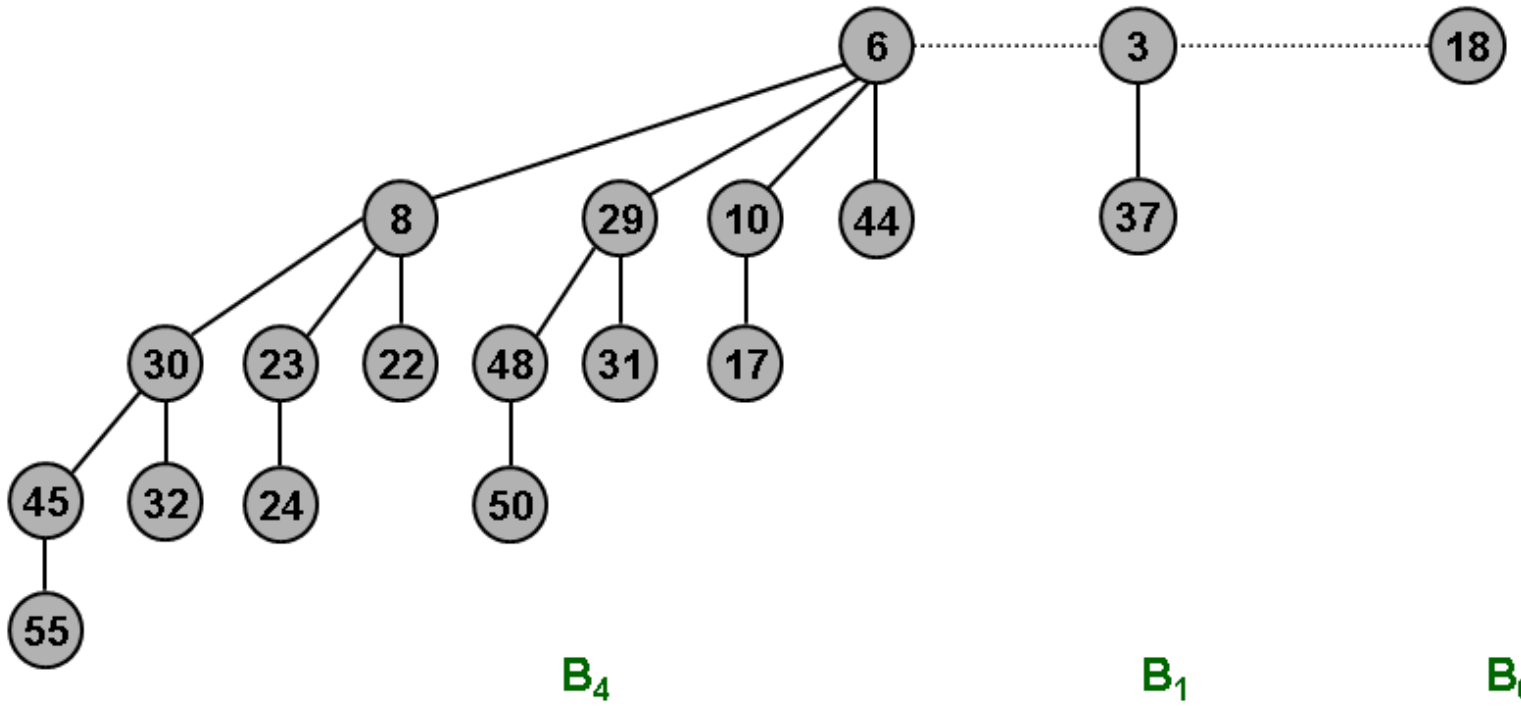
$$\binom{4}{2} = 6$$



Binomial Heap

Binomial heap. Vuillemin, 1978.

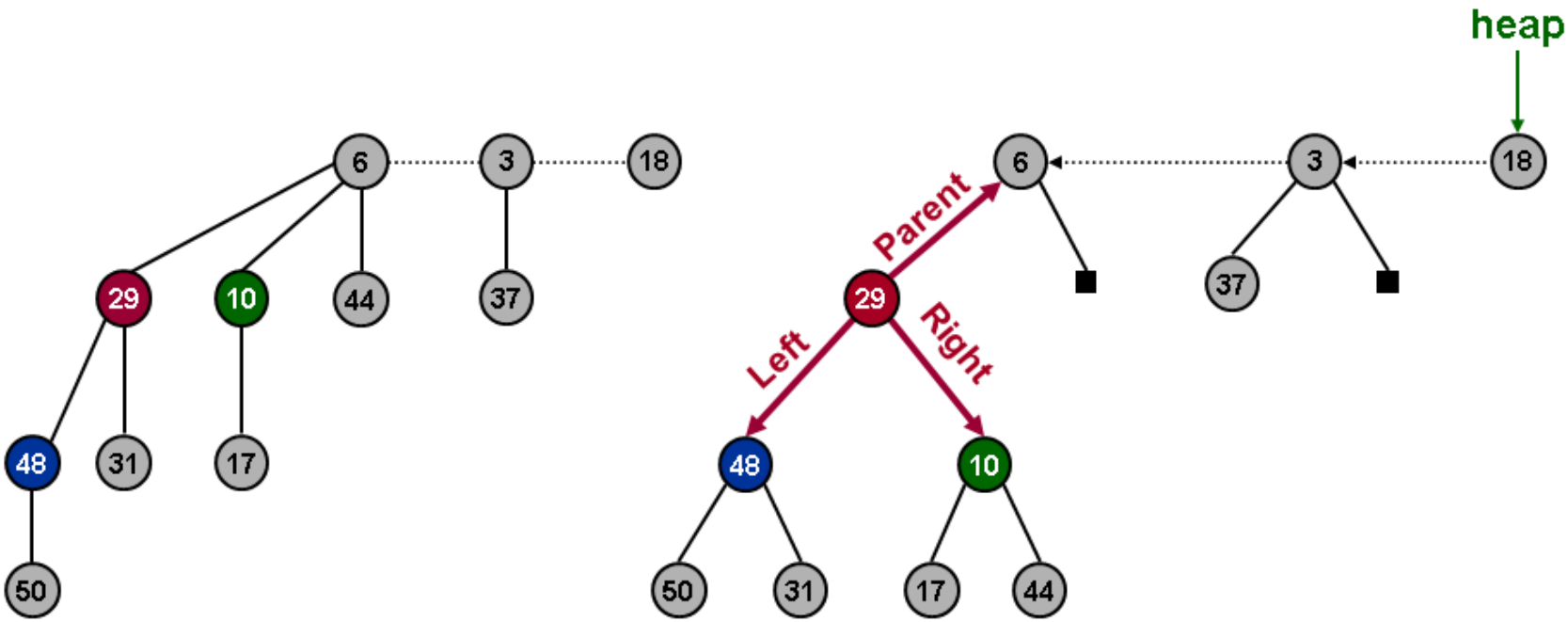
- Sequence of binomial trees that satisfy binomial heap property.
  - each tree is min-heap ordered
  - 0 or 1 binomial tree of order k



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



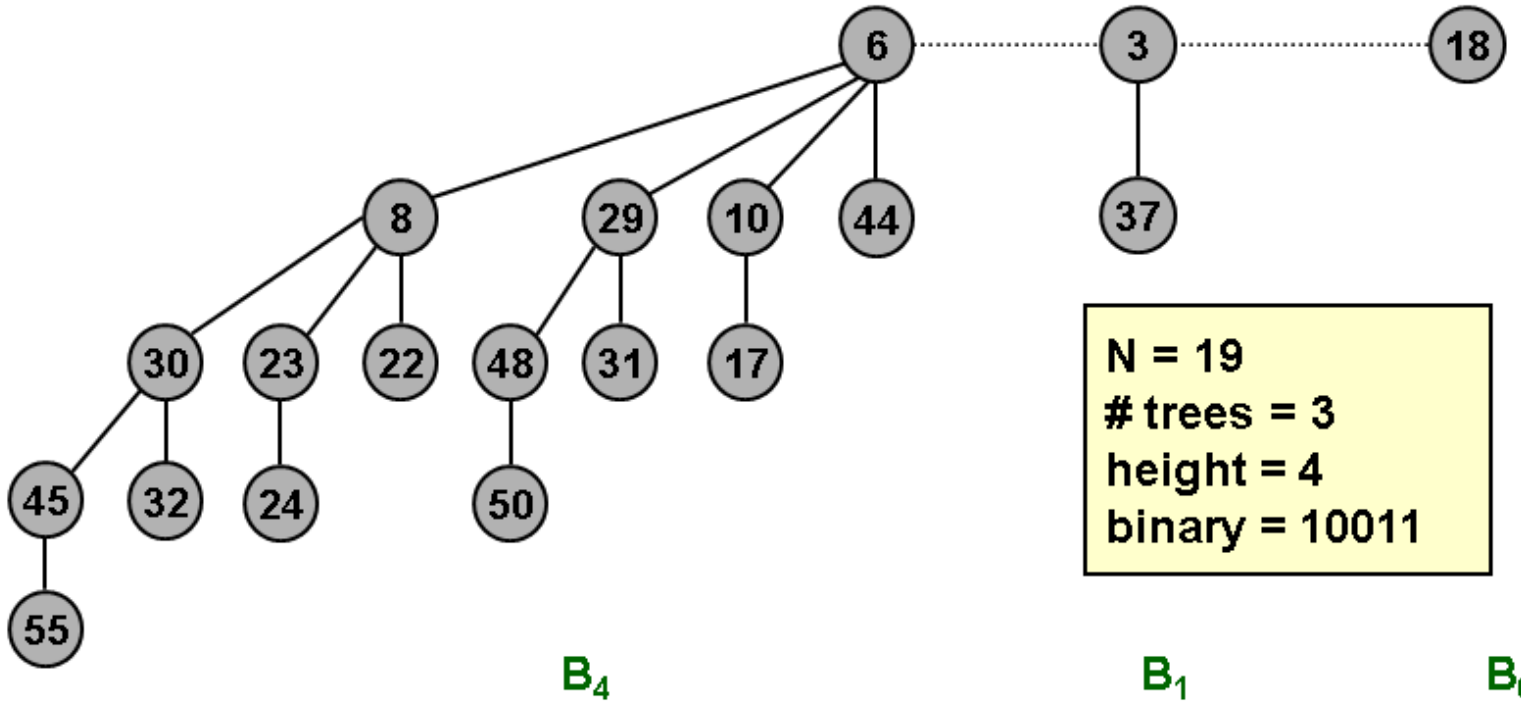
Binomial Heap

Leftist Power-of-2 Heap

Binomial Heap: Properties

Properties of N-node binomial heap.

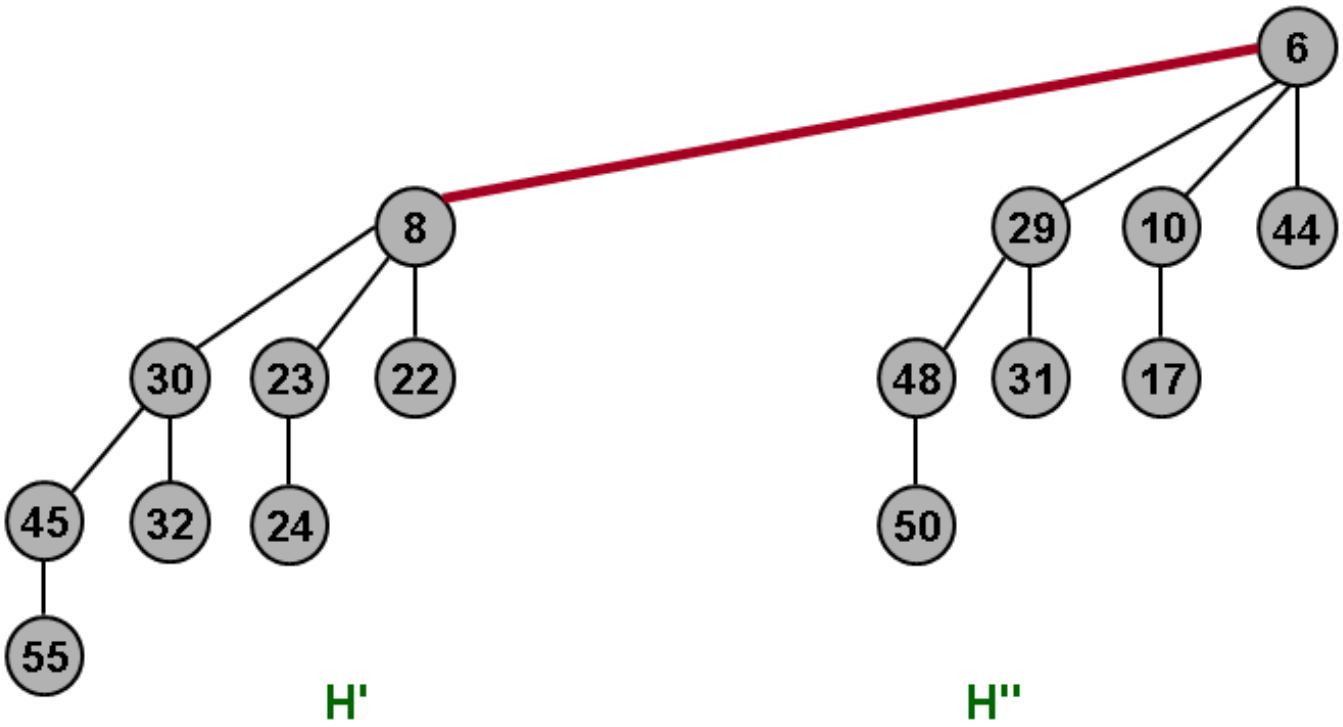
- Min key contained in root of $B_0, B_1, \ldots, B_k$.
- Contains binomial tree B_i iff $b_i = 1$ where $b_n \cdot b_2 b_1 b_0$ is binary representation of N.
- At most $\lfloor \log_2 N \rfloor + 1$ binomial trees.
- Height $\leq \lfloor \log_2 N \rfloor$.



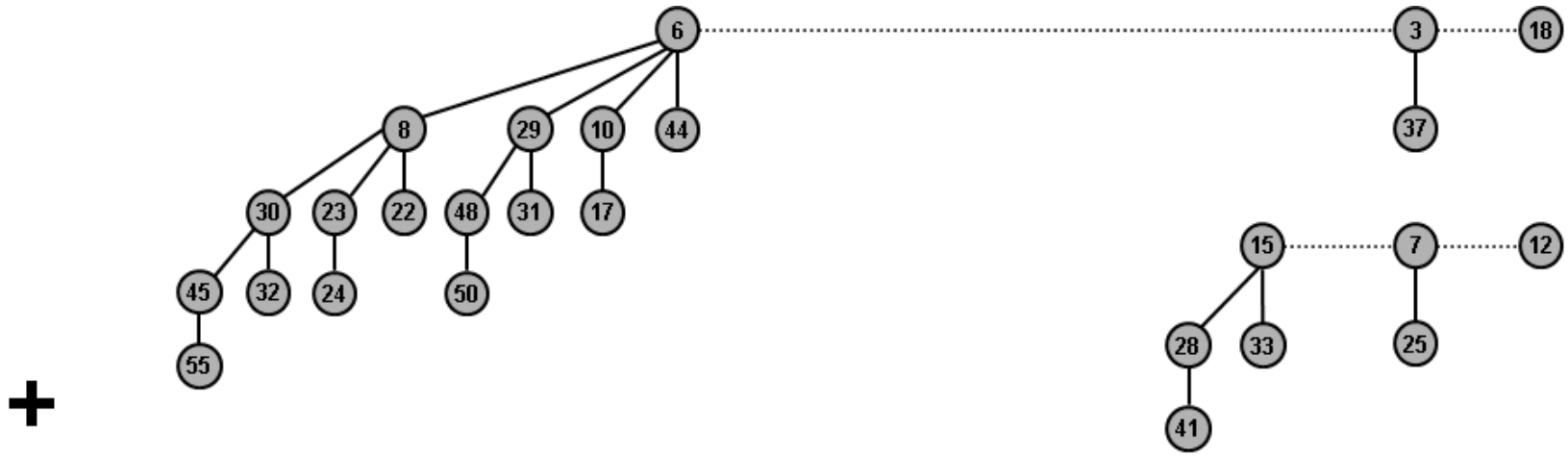
Binomial Heap: Union

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



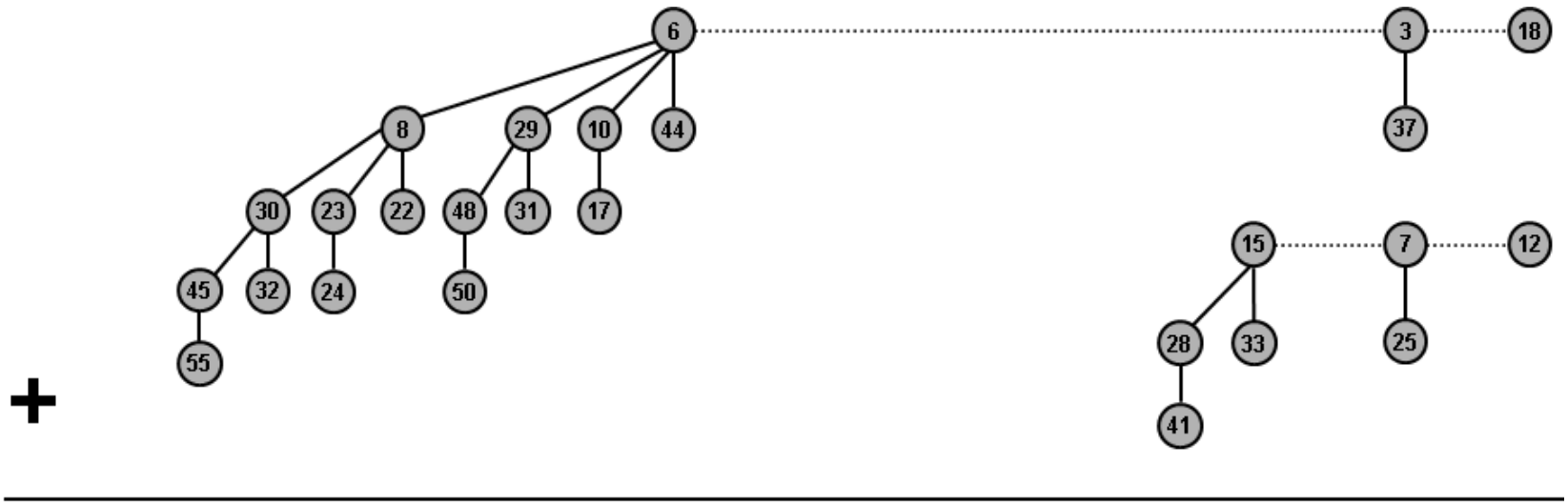
Binomial Heap: Union



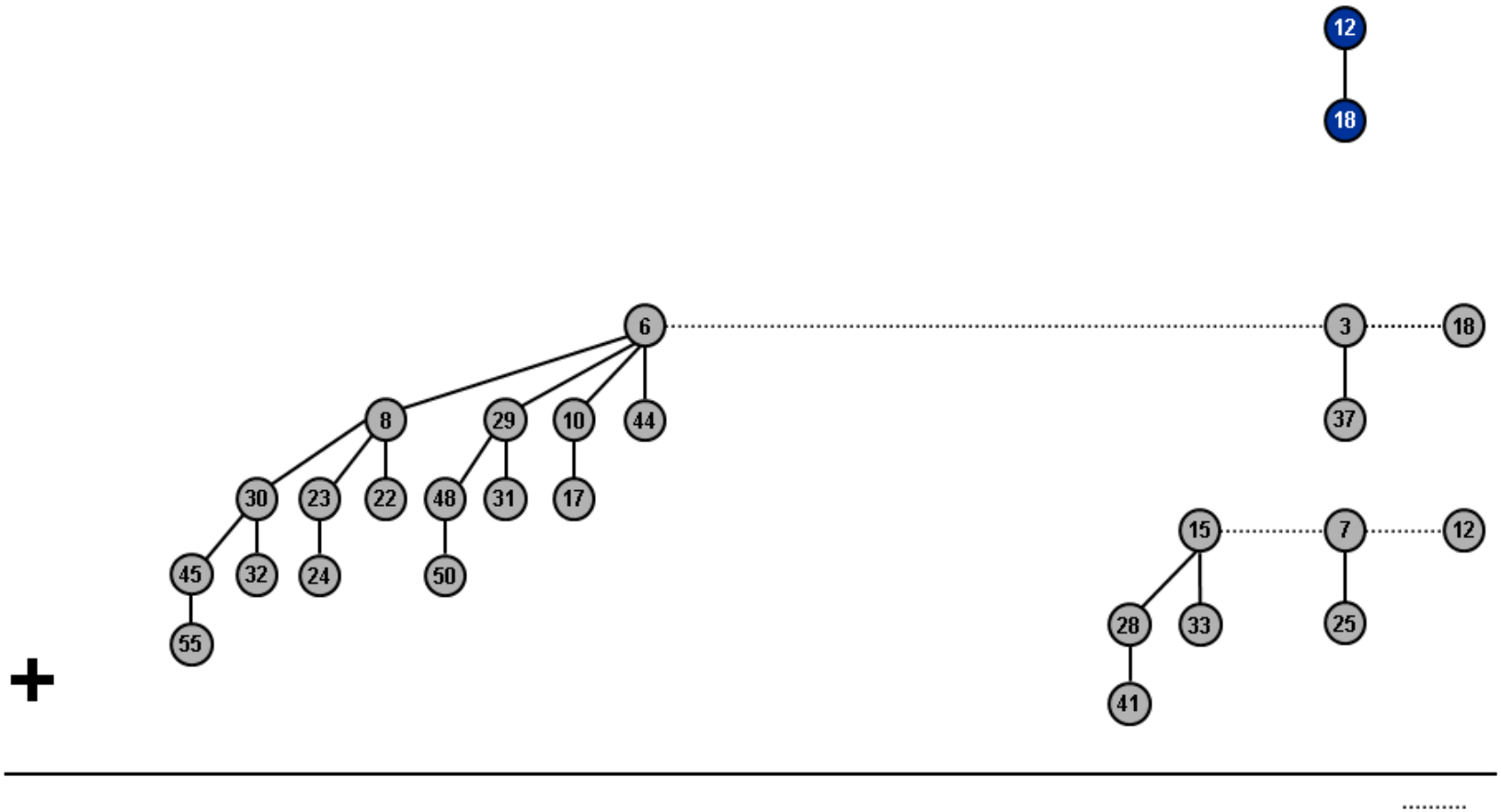
19 + 7 = 26

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

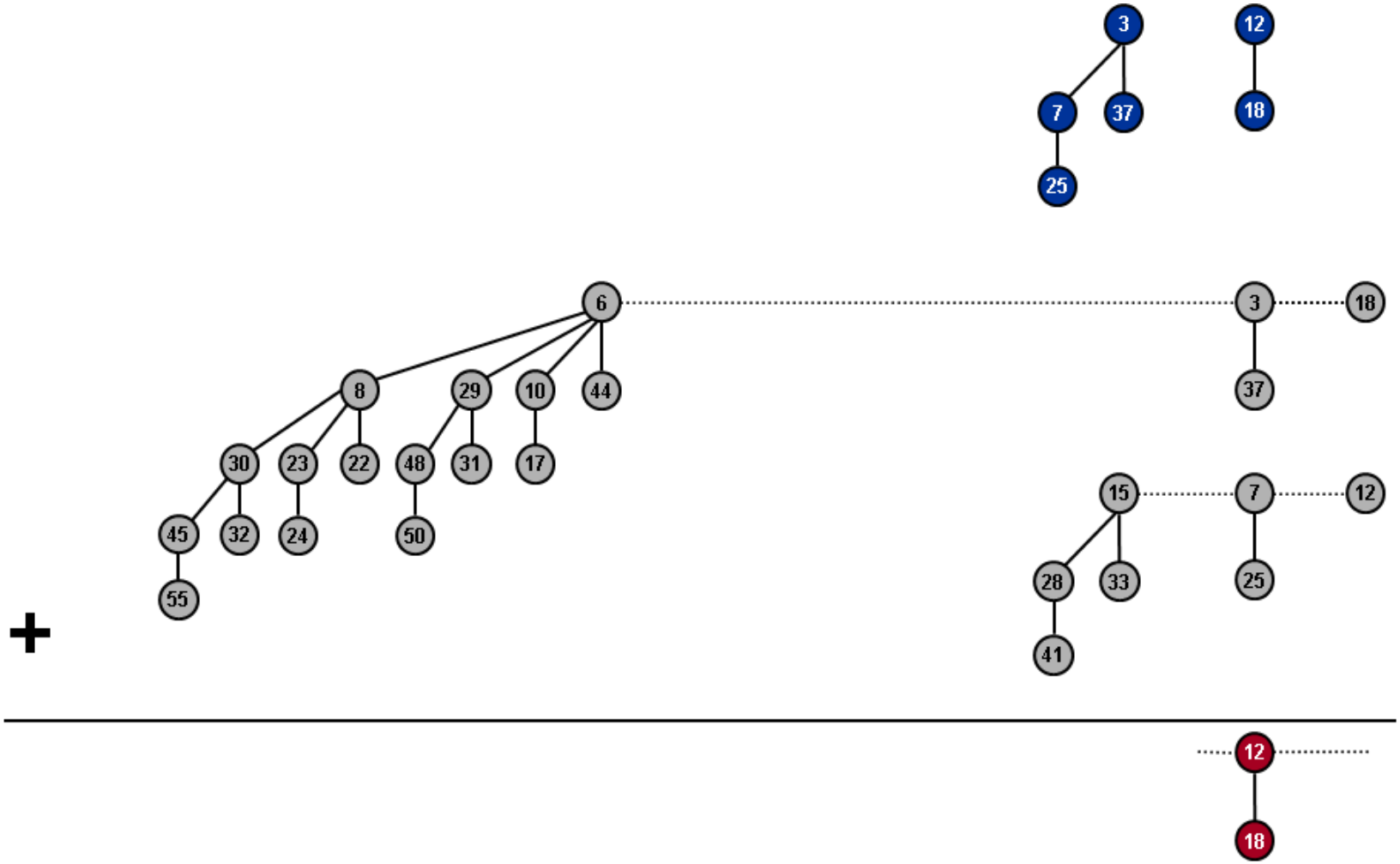
Binomial Heap: Union



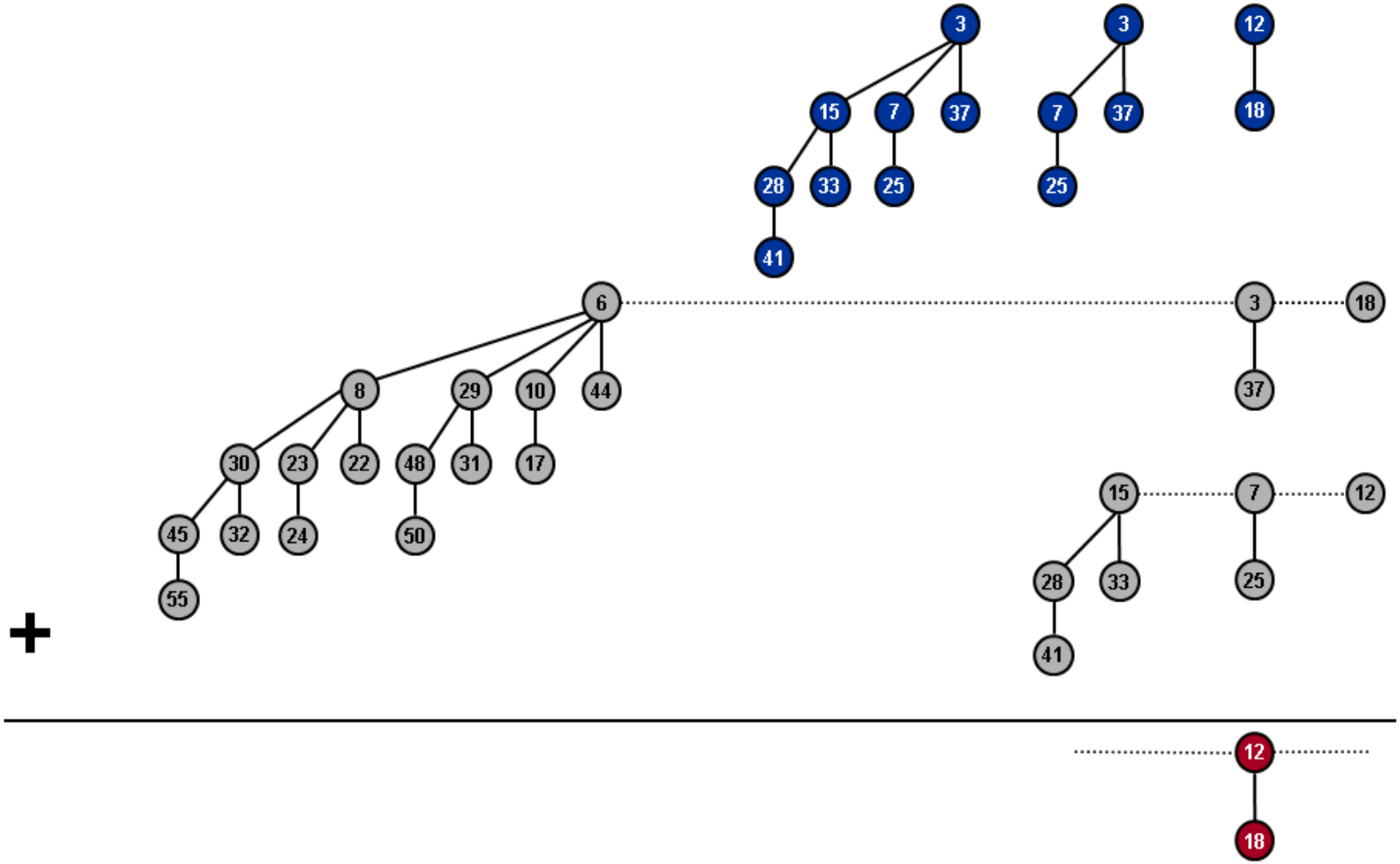
Binomial Heap: Union



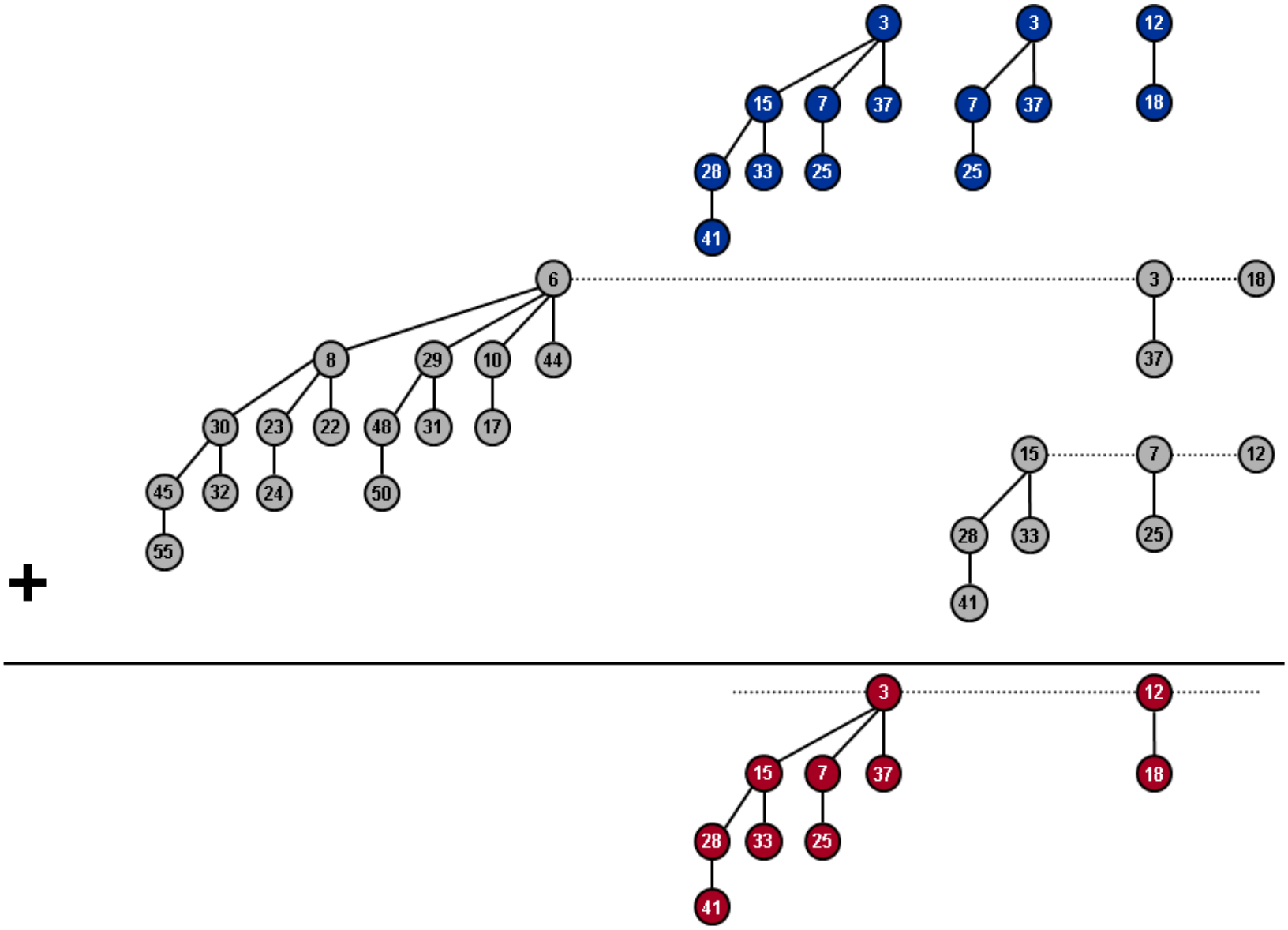
Binomial Heap: Union



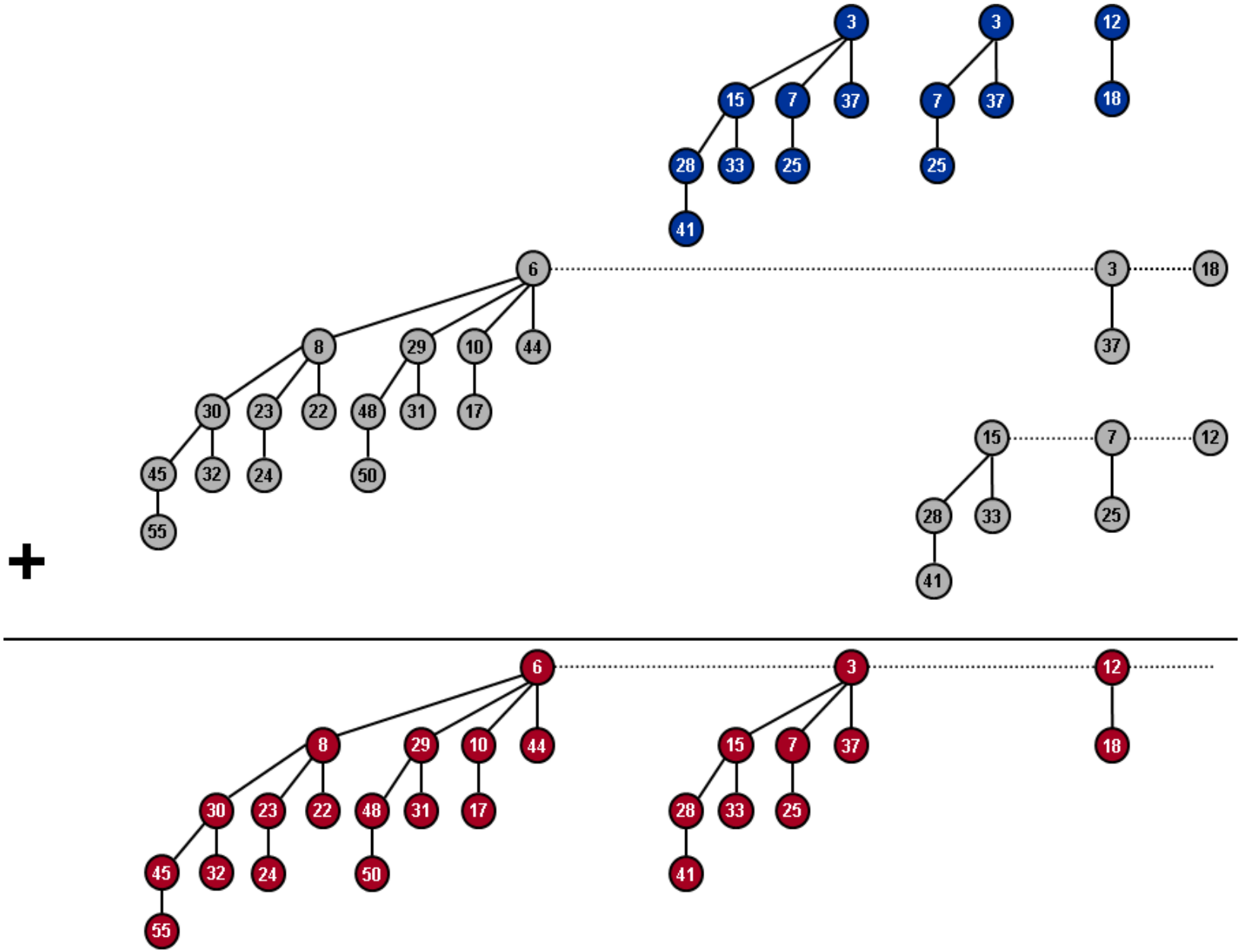
Binomial Heap: Union



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

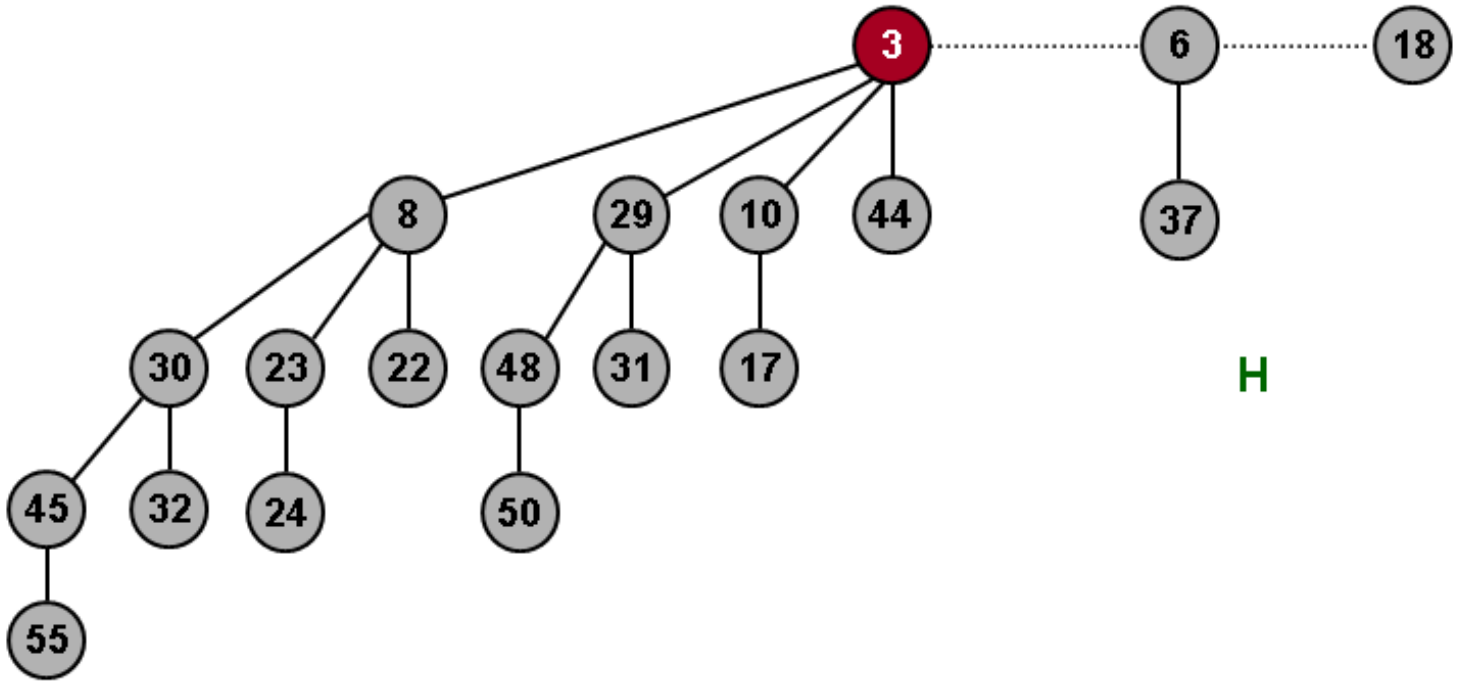
		1	1	1	
	1	0	0	1	1
+	0	0	1	1	1
	1	1	0	1	0

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' $\leftarrow$ broken binomial trees
- H $\leftarrow$ Union(H', H)

Running time. O(log N)

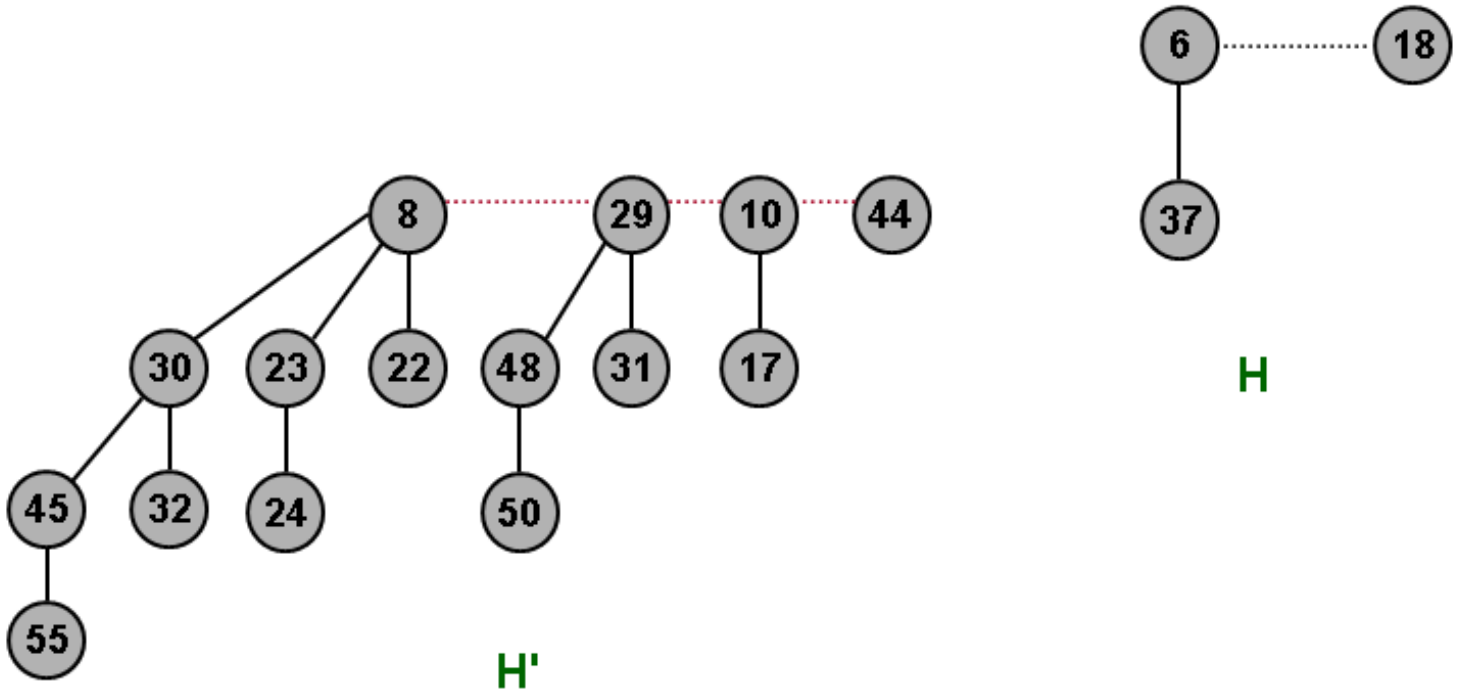


Binomial Heap: Delete Min

Delete node with minimum key in binomial heap H.

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

Running time. O(log N)



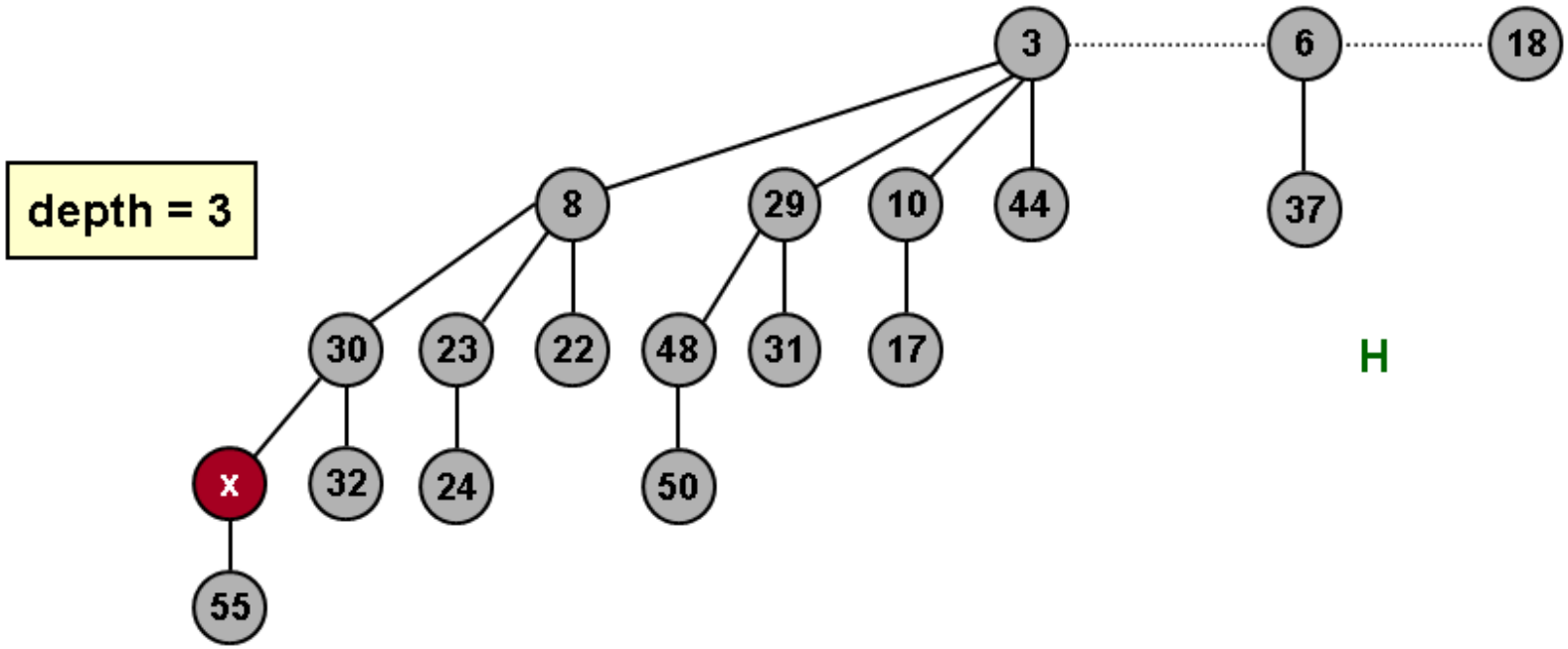
Binomial Heap: Decrease Key

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

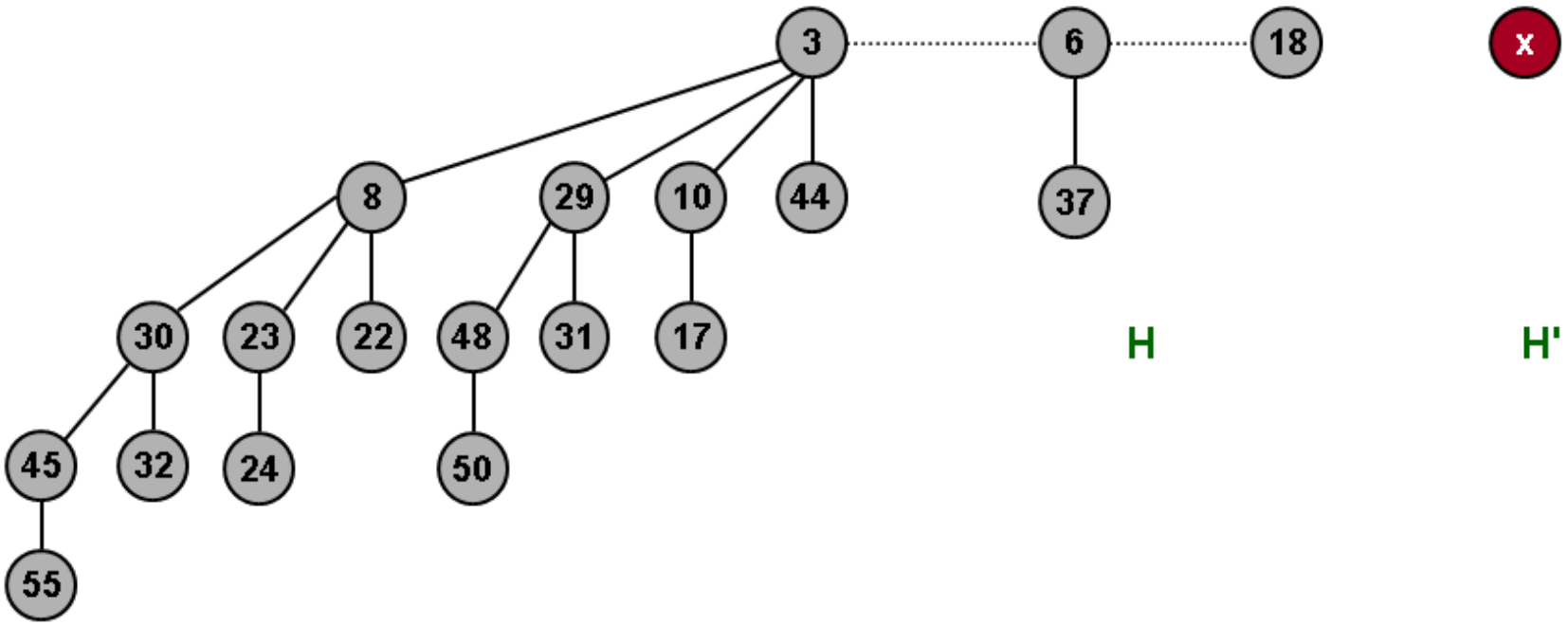
- Delete node x in binomial heap H.
  - Decrease key of x to -∞
  - Delete min.
- Running time. O(log N)

Binomial Heap: Insert

Insert a new node x into binomial heap H.

- H' ← MakeHeap(x)
- H ← Union(H', H)

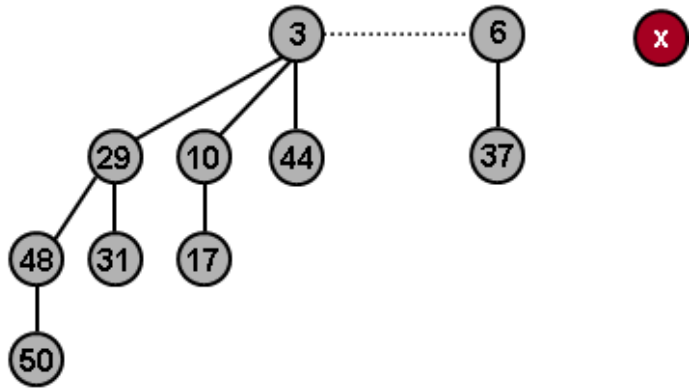
Running time. O(log N)



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 Ω(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.

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

Dijkstra's Algorithm (Single source shortest path problem)

```
Algorithm Dijkstra(V, E, w, s)
    Q := MakeQueue
    dist[s] := 0
    Insert(Q, s, 0)
    for v ∈ V \ { s } do
        dist[v] := +∞
        Insert(Q, v, +∞)
    while Q ≠ ∅ do
        v := DeleteMin(Q)
        foreach u : (v, u) ∈ E do
            if u ∈ Q and dist[v]+w(v, u) < dist[u] then
                dist[u] := dist[v]+w(v, u)
                DecreaseKey(u, dist[u])
```

n x Insert + *n* x DeleteMin + *m* x DecreaseKey

Binary heaps / Binomial queues : $O((n + m) \cdot \log n)$

Priority Queues

	Binomial Queues [Vuillemin 78]	Fibonacci Heaps [Fredman, Tarjan 84]	Run-Relaxed Heaps [Driscoll, Gabow, Shrairman, Tarjan 88]	[Brodal 96]
Insert	1	1	1	1
Meld	1	1	-	1
Delete	$\log n$	$\log n$	$\log n$	$\log n$
DeleteMin	$\log n$	$\log n$	$\log n$	$\log n$
DecreaseKey	$\log n$	1	1	1
		Amortized	Worst-case	

$\Downarrow$

Dijkstra's Algorithm $O(m + n \cdot \log n)$

(and Minimum Spanning Tree $O(m \cdot \log^* n)$)

Empty, FindMin, Size, MakeQueue – O(1) worst-case time