Binomial Heaps

Data Structures and Algorithms
Andrei Bulatov

Priority Queues

- Supports the following operations.
  - Insert element x.
  - Return min element.
  - Return and delete minimum element.
  - Decrease key of element x to k.
- Applications.
  - Dijkstra's shortest path algorithm.
  - Prim's MST algorithm.
  - Event-driven simulation.
  - Huffman encoding.
  - Heapsort.
  - . . .

Dijkstra's Algorithm

```
set S:={s} and  d(s):=0
while S≠V do
  pick a node v not from S such that the value
```

$$d'(v) := \min_{e=(u,v),u\in S}\{d(u)+len(e)\}$$

```
  is minimal
  set S:=S∪{v}, and  d(v):=d'(v)
endwhile
```

Dijkstra's Algorithm: PQ Style

```
call PQinit
set S:=V
set key(s):=0
PQinsert(s)
for each v∈V-{s}
   key(v):=∞
   call PQinsert(v)

while not PQisempty
   set v:=Pqdelmin
   set S:=S-{v}
   for each w∈Q such that (v,w)∈E
      if key(w)>key(v)+len(v,w) then
         call PQdecrease(w,key(v)+len(v,w))
```

Priority Queues

		Heaps			
Operation	Linked List	Binary	Binomial	Fibonacci *	Relaxed
make-heap	1	1	1	1	1
insert	1	log N	log N	1	1
find-min	N	1	log N	1	1
delete-min	N	log N	log N	log N	log N
union	1	N	log N	1	1
decrease-key	1	log N	log N	1	1
delete	N	log N	log N	log N	log N
is-empty	1	1	1	1	1

Dijkstra/Prim
1 make-heap
|V| insert
|V| delete-min
|E| decrease-key

$O(|V|^2)$

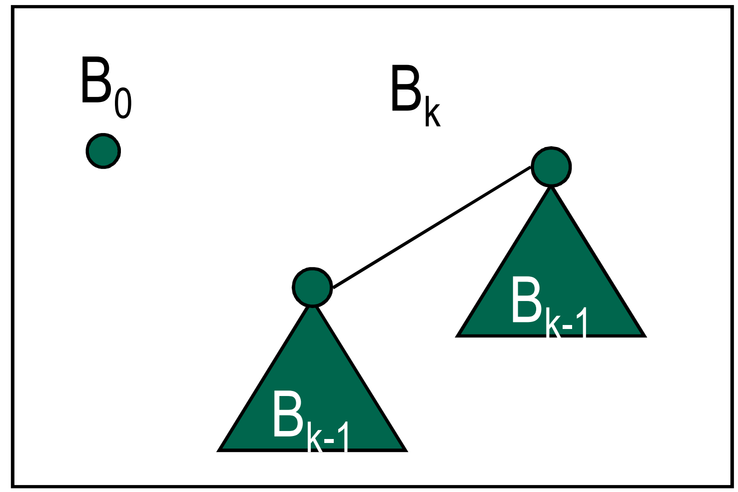
$O(|E| \log |V|)$

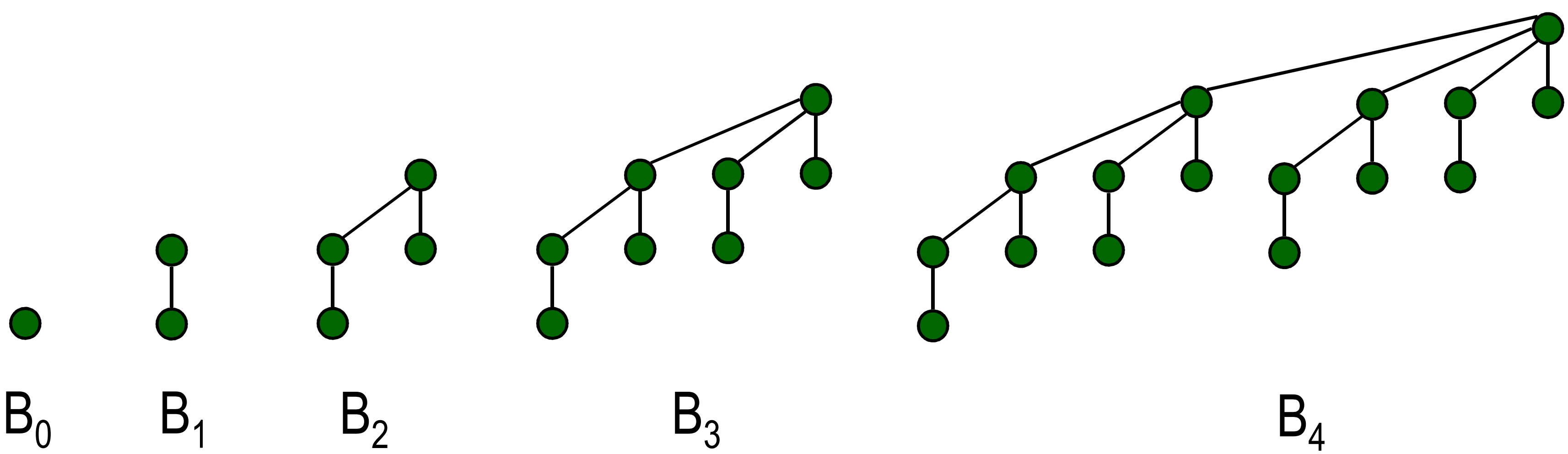
$O(|E| + |V| \log |V|)$

Binomial Tree

Recursive definition:

- B_0 is a single node
- B_k is obtained from attaching one copy of B_{k-1} as the leftmost child of another B_{k-1}





Binomial Tree

Lemma

For a binomial tree B_k.

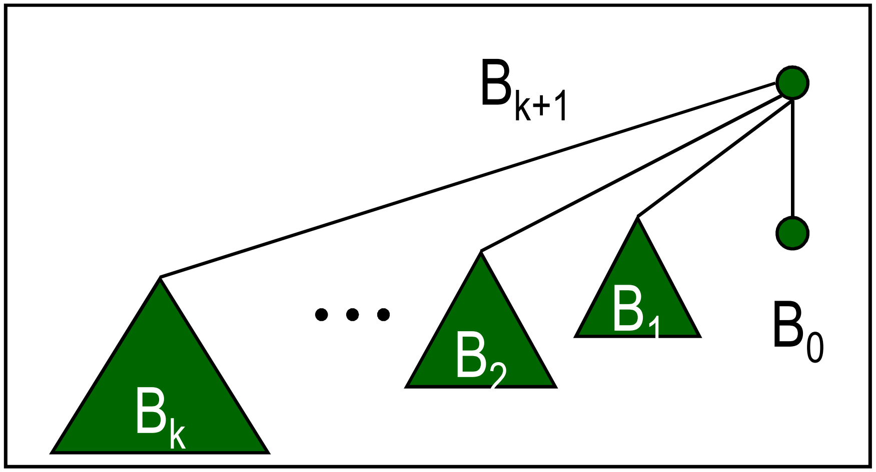
(a) Number of nodes equals 2^k.

(b) Height equals k.

(c) Degree of root equals k.

(d) Deleting root yields binomial trees $B_{k-1}, \ldots, B_0$.

(e) B_k has $\binom{k}{i}$ nodes at depth i.

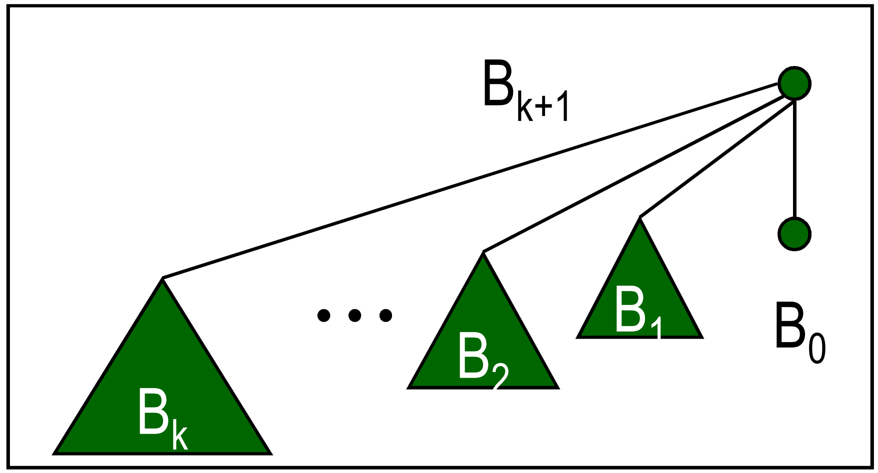


Binomial Tree

Proof Induction on k

Base case: For B_0 all claims are obvious.

Induction Step: Suppose the lemma is true for B_{k-1}



(a) $|B_k| = |B_{k-1}| + |B_{k-1}| = 2^k$

(b), (c), (d) Exercise

(d) Denote the number of nodes of B_k at depth i by N(k,i)

Then $N(k,i) = N(k-1, i) + N(k-1, i-1)$

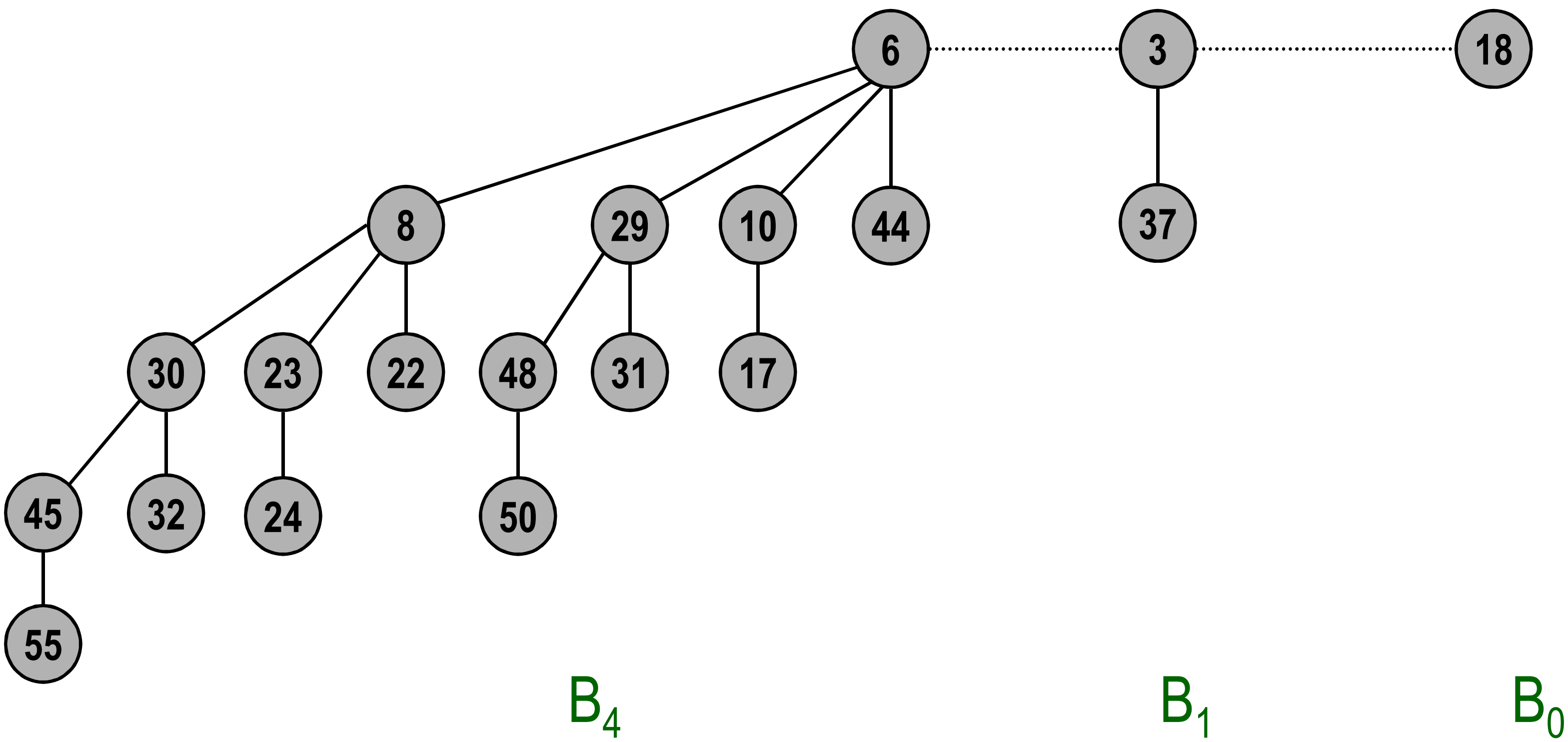
$$= \binom{k-1}{i} + \binom{k-1}{i-1}$$

$$= \binom{k}{i} \quad \longleftarrow \quad \text{Pascal's identity}$$

Binomial Heap

Binomial heap.

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



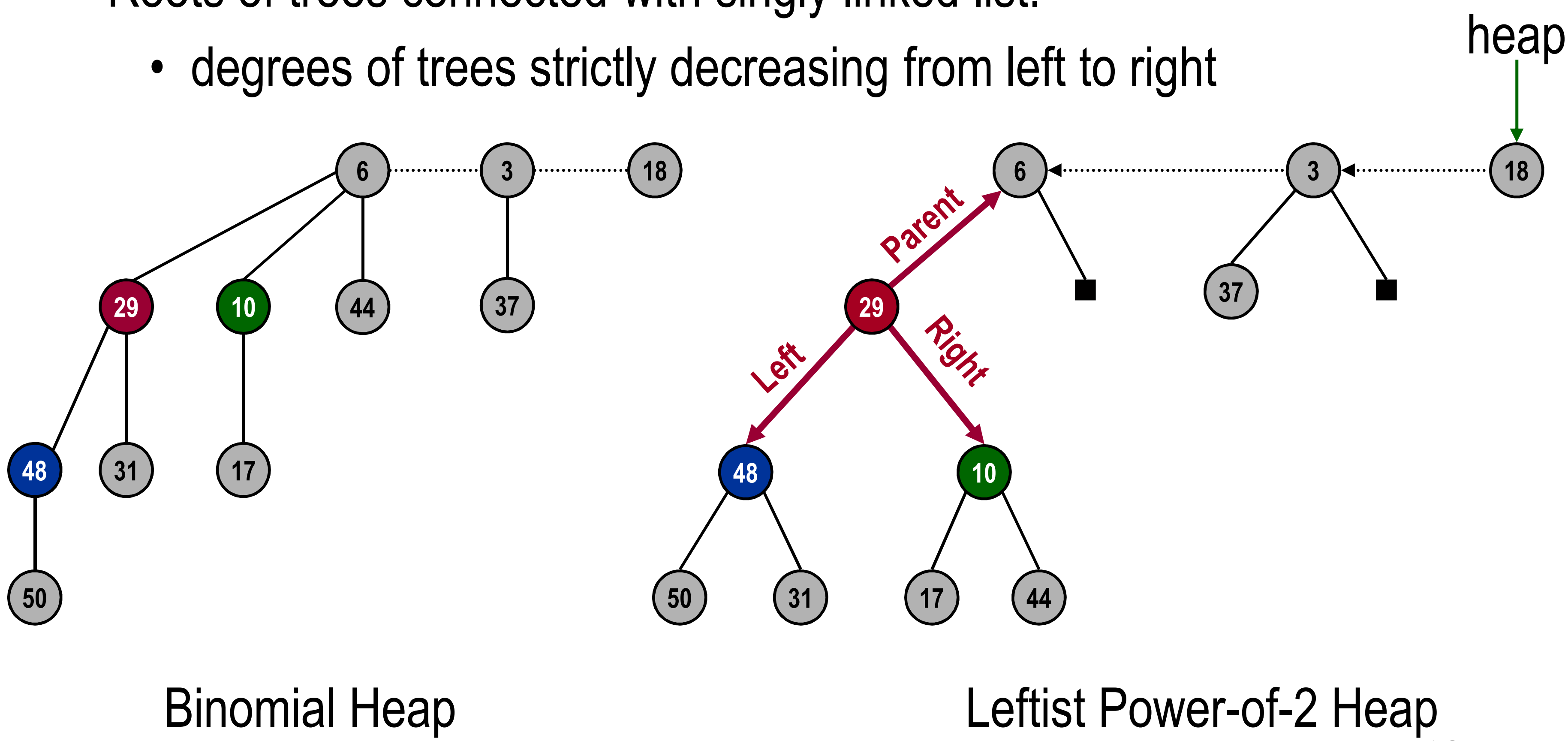
Binomial Heap: Implementation

Represent trees using left-child, right-child pointers.

- three links per node (parent, left, right)

Roots of trees connected with singly linked list.

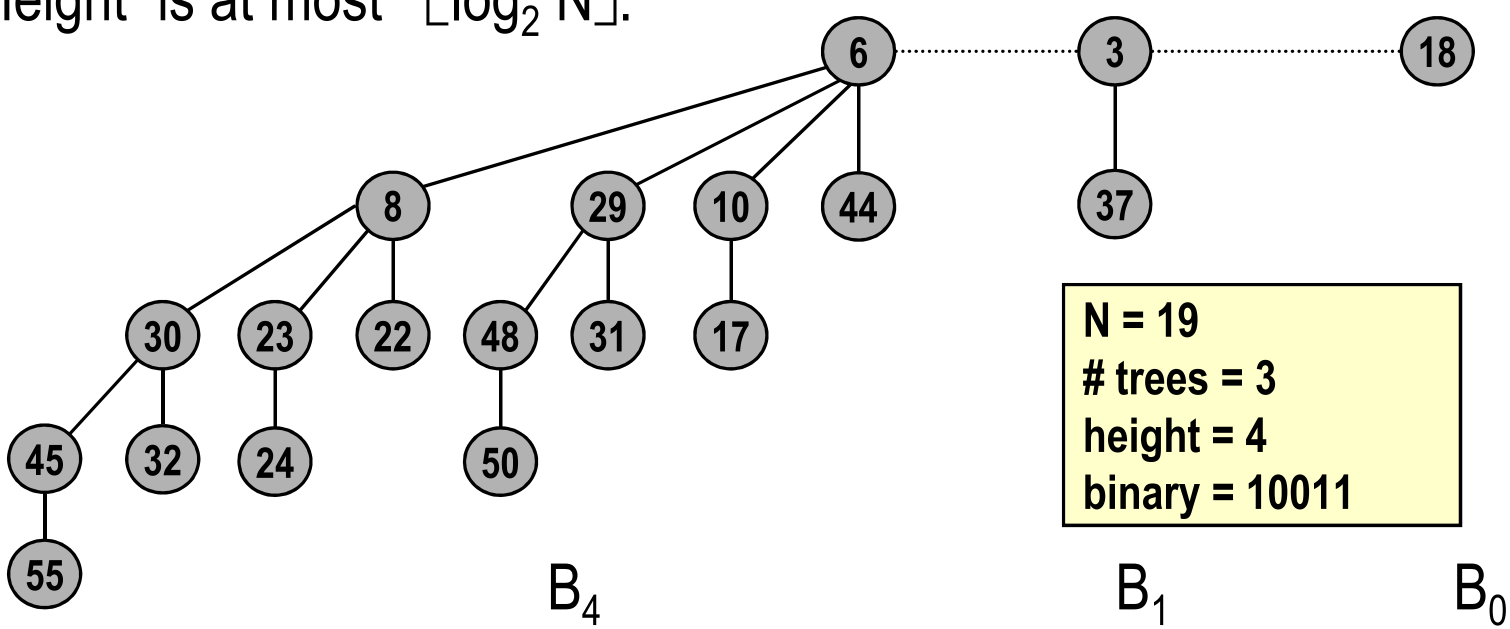
- degrees of trees strictly decreasing from left to right



Binomial Heap: Properties

Properties of N-node binomial heap.

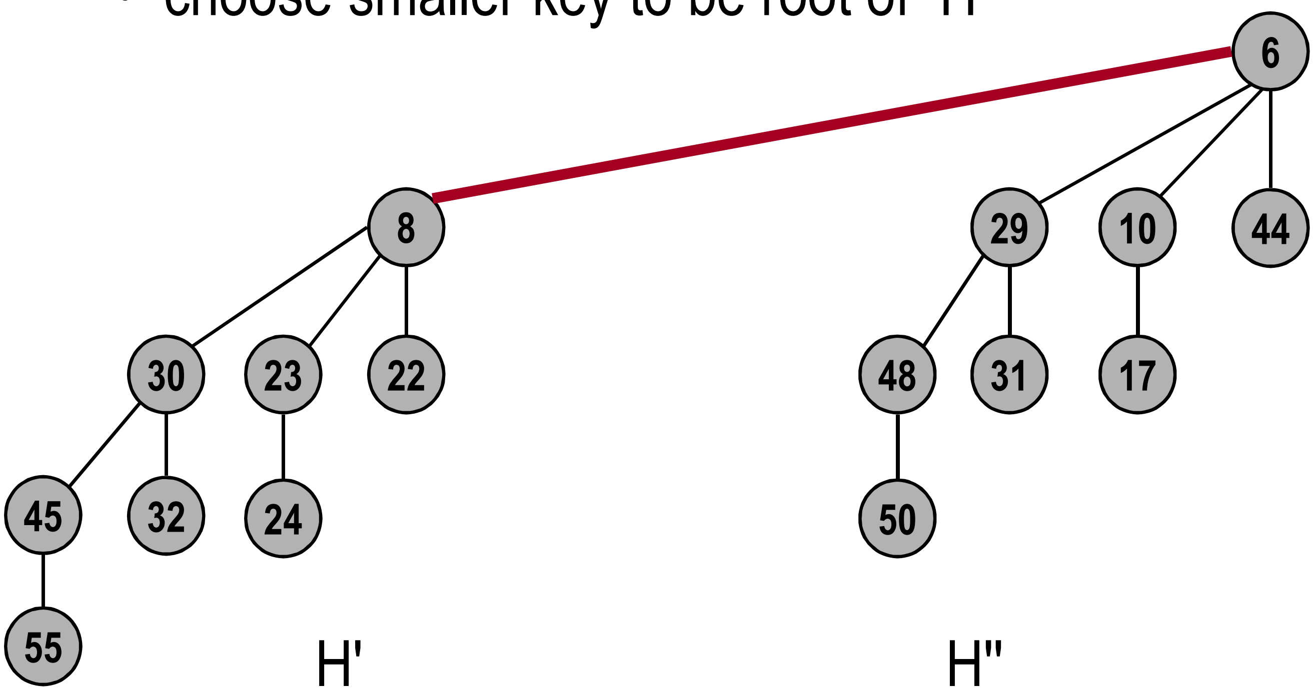
- Min key contained in root of $B_0, B_1, \ldots, B_k$.
- Contains binomial tree B_i iff $b_i = 1$ where $b_n b_2 b_1 b_0$ is binary representation of N.
- At most $\lfloor \log_2 N \rfloor + 1$ binomial trees.
- Height is at most $\lfloor \log_2 N \rfloor$.



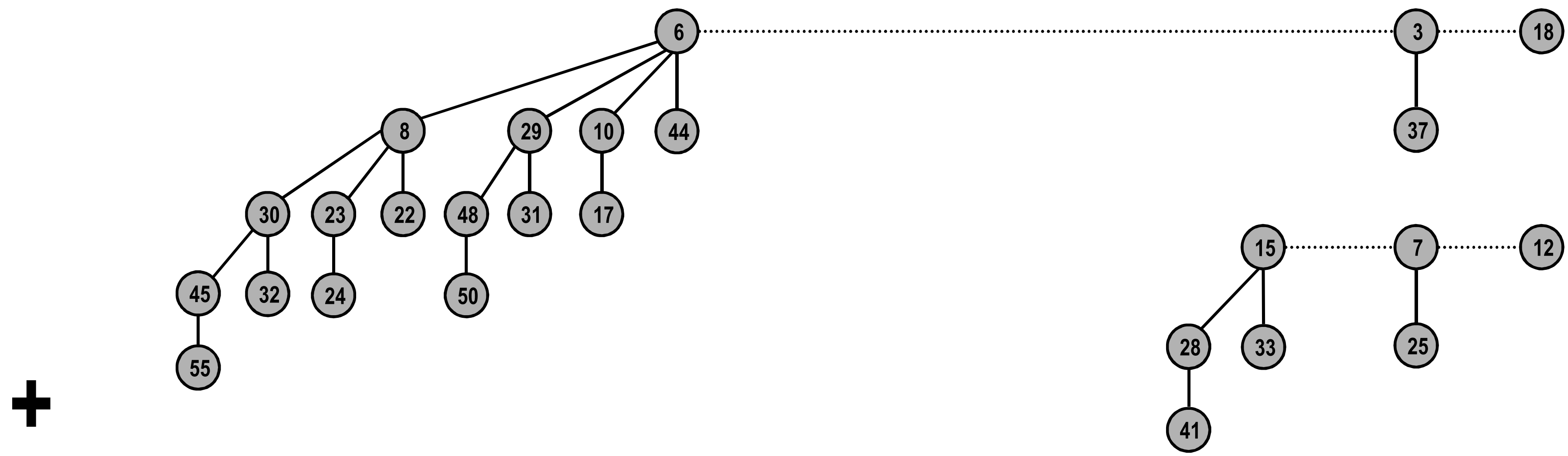
Binomial Heap: Union

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



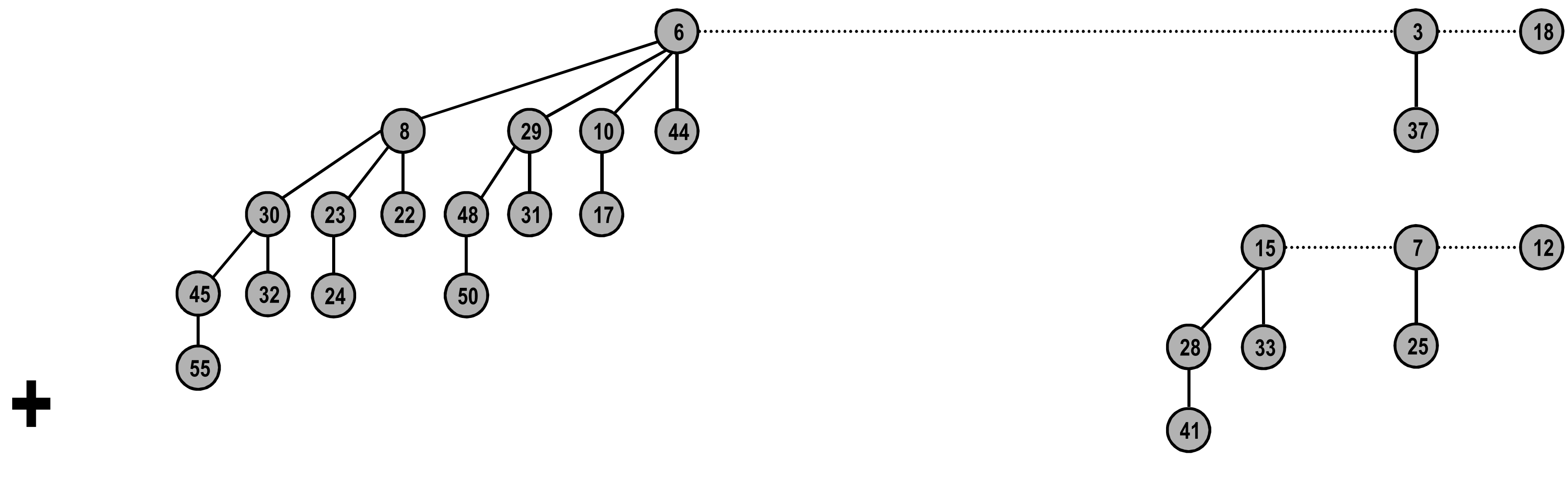
Binomial Heap: Union



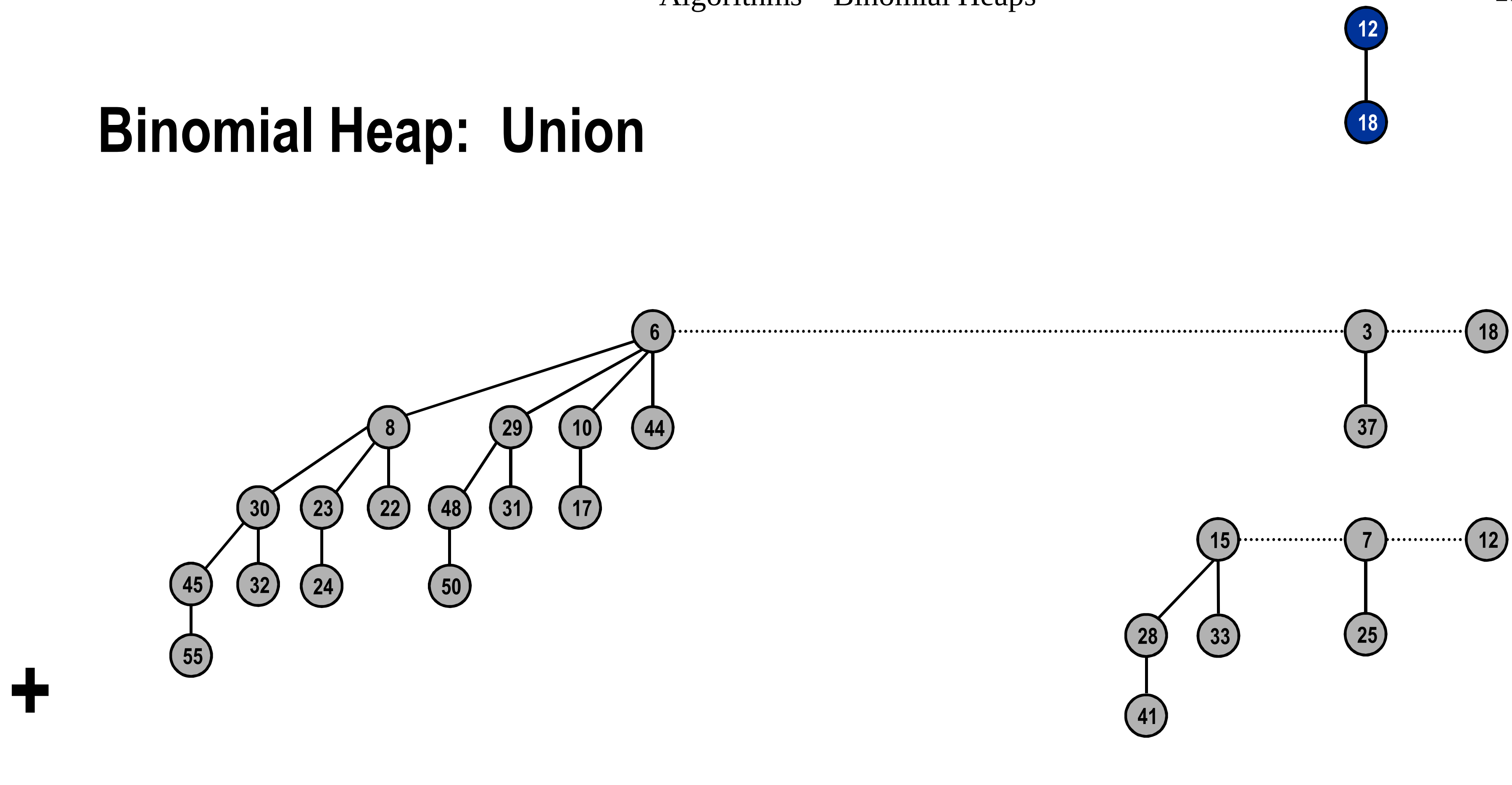
19 + 7 = 26

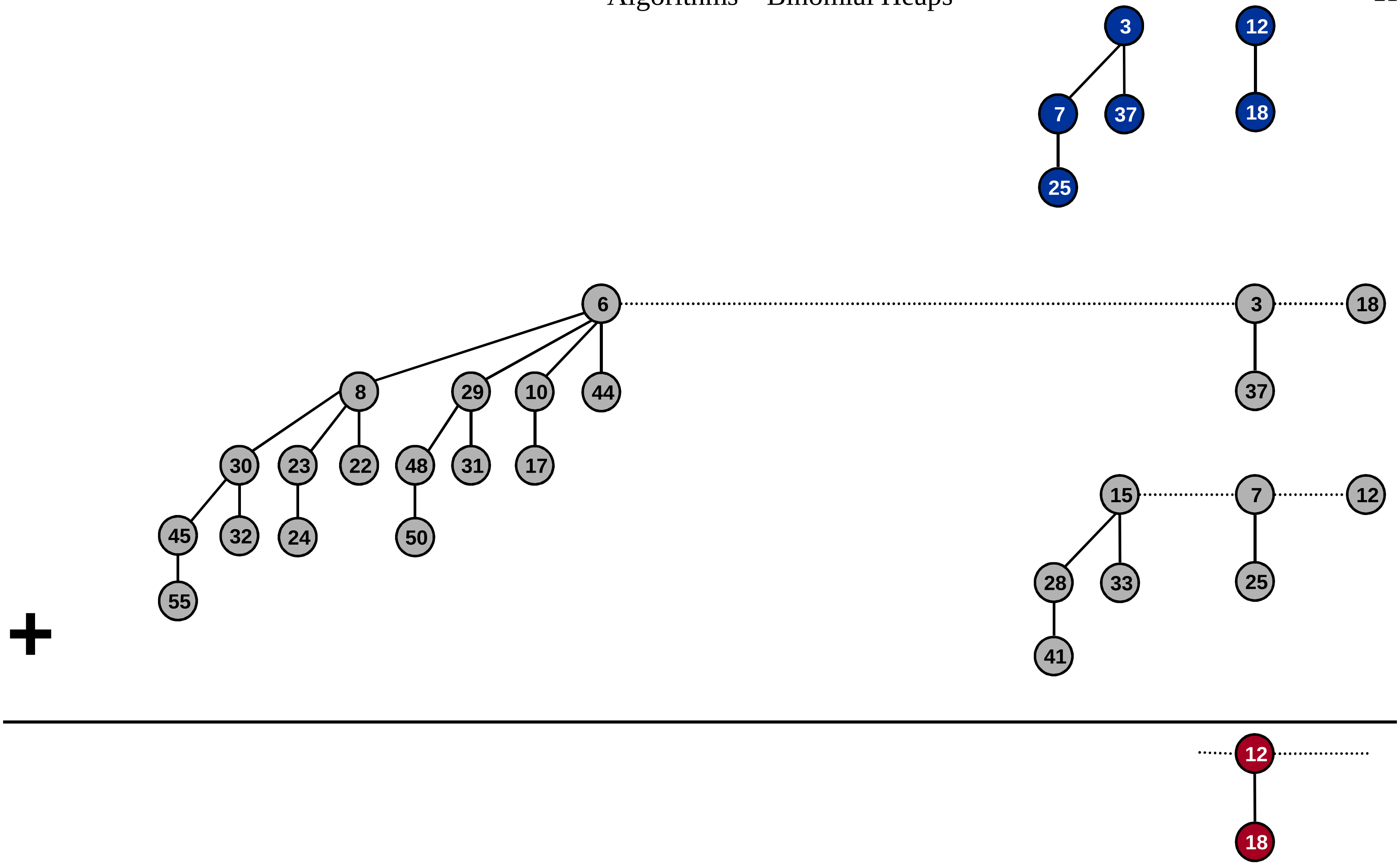
		1	1	1	
	1	0	0	1	1
+	0	0	1	1	1
	1	1	0	1	0

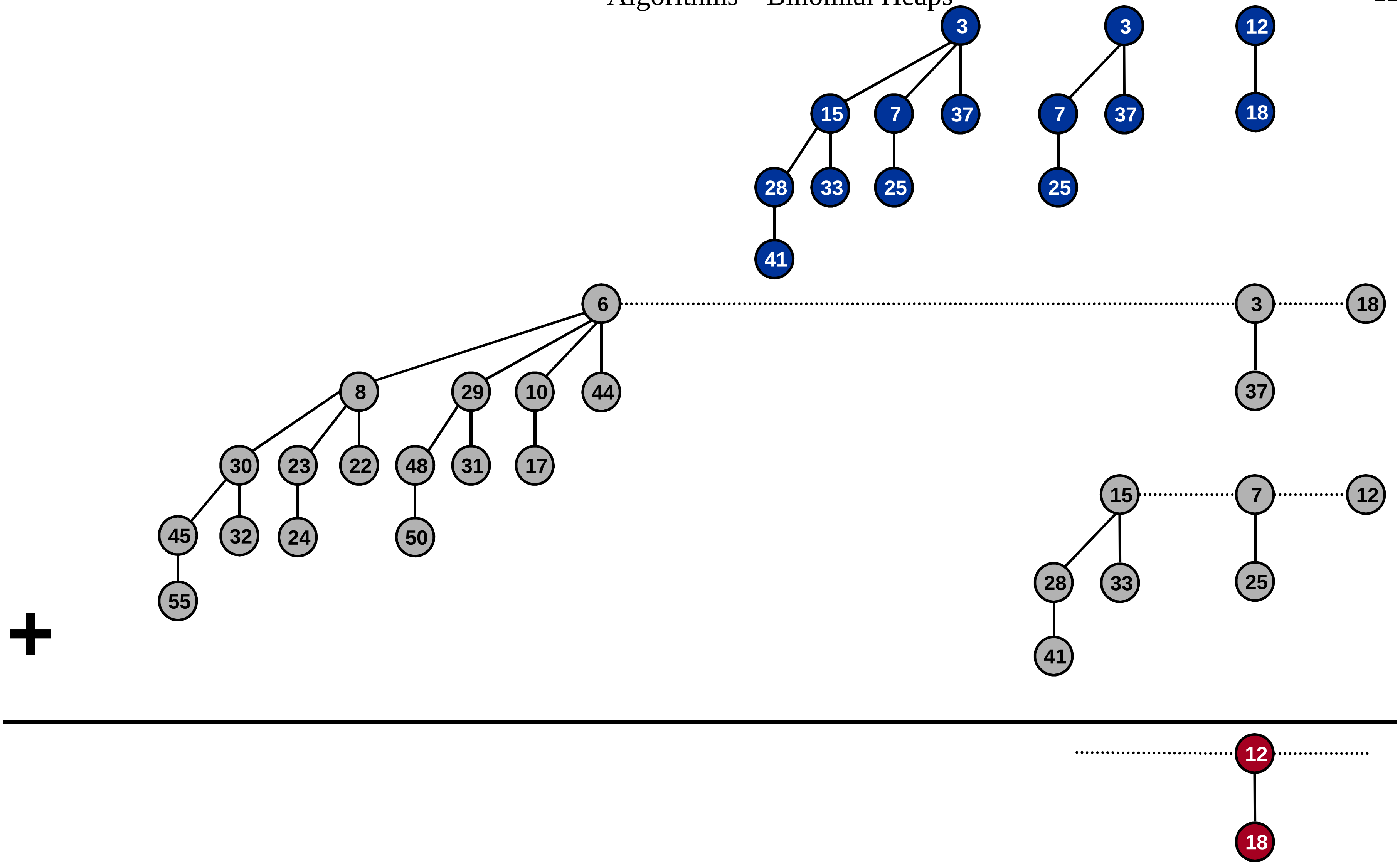
Binomial Heap: Union

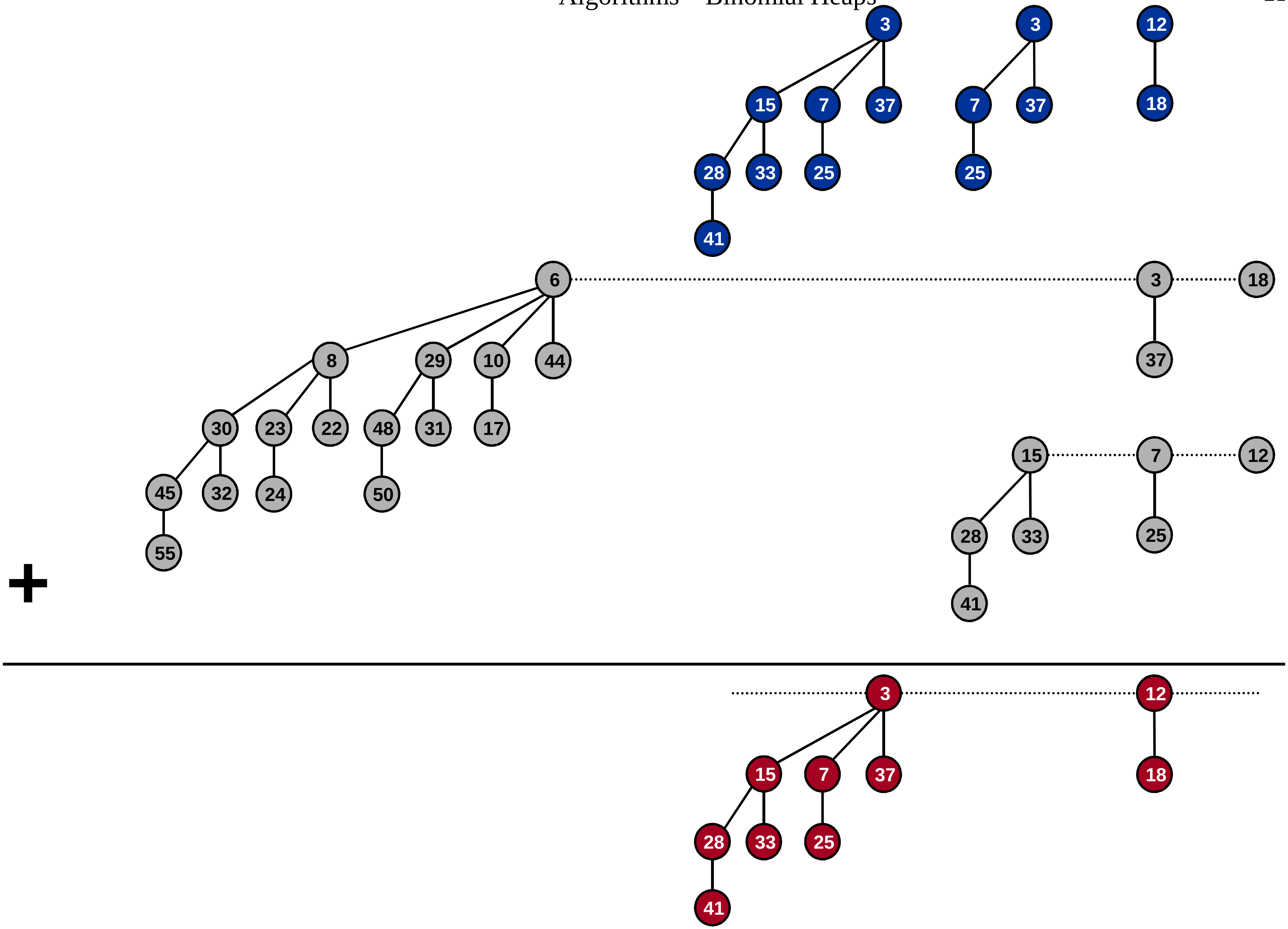


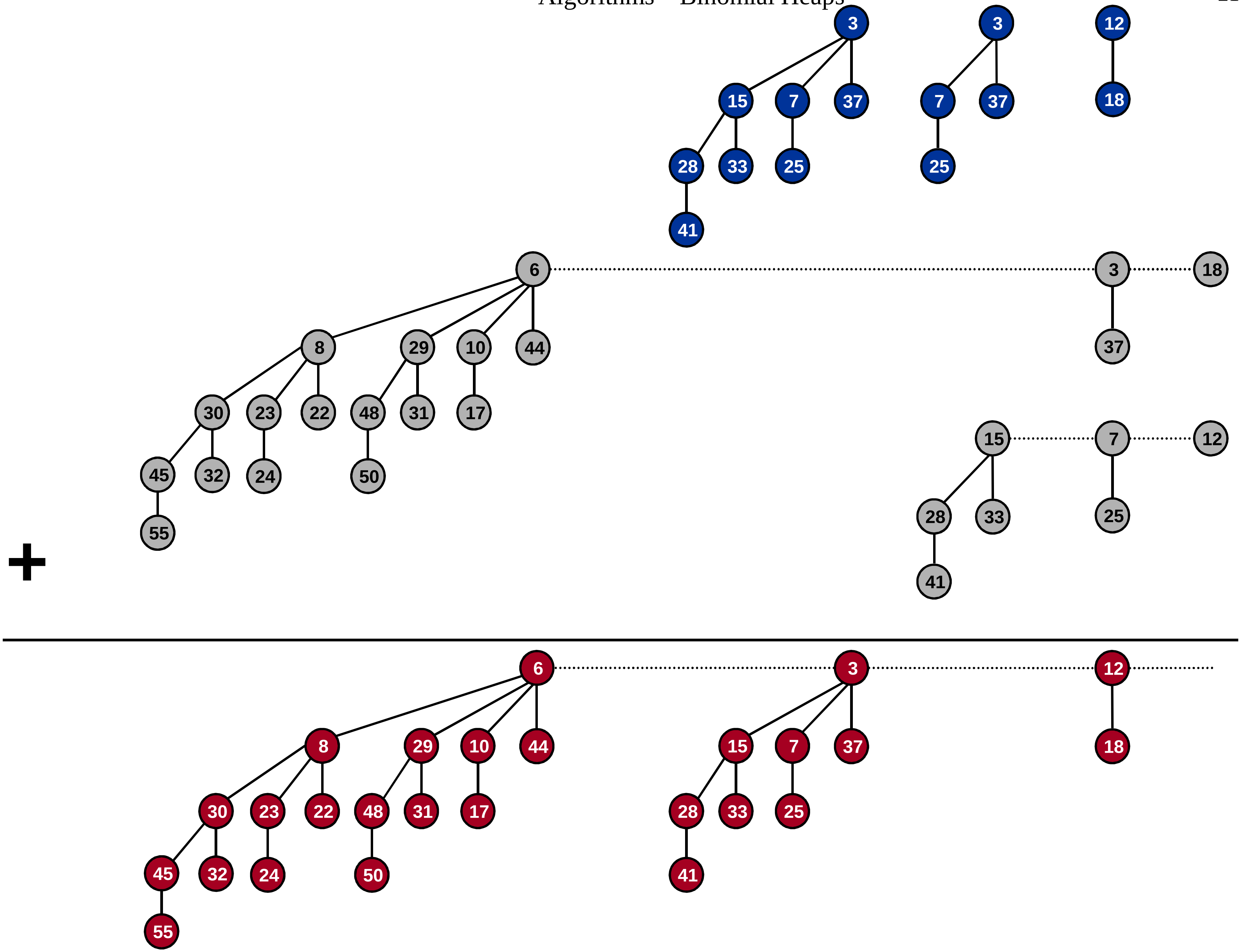
Binomial Heap: Union




3
7
37
25
12
18
6
8
29
10
44
30
23
22
48
31
17
45
32
24
50
55
3
18
37
15
7
12
28
33
25
41
+
12
18


3
15
7
37
28
33
25
41
3
7
37
25
12
18
6
8
29
10
44
30
23
22
48
31
17
45
32
24
50
55
3
18
37
15
7
12
28
33
25
41
+
12
18


3
3
12
15
7
37
7
37
18
28
33
25
25
41
6
3
18
8
29
10
44
37
30
23
22
48
31
17
15
7
12
45
32
24
50
28
33
25
55
41
+
3
12
15
7
37
18
28
33
25
41


3
15
7
37
28
33
25
41
3
7
37
25
12
18
6
8
29
10
44
30
23
22
48
31
17
45
32
24
50
55
3
37
18
15
7
12
28
33
25
41
+
6
8
29
10
44
30
23
22
48
31
17
45
32
24
50
55
3
15
7
37
28
33
25
41
12
18

Union: Running Time

Theorem

Union can be executed in $O(n)$ time

Proof

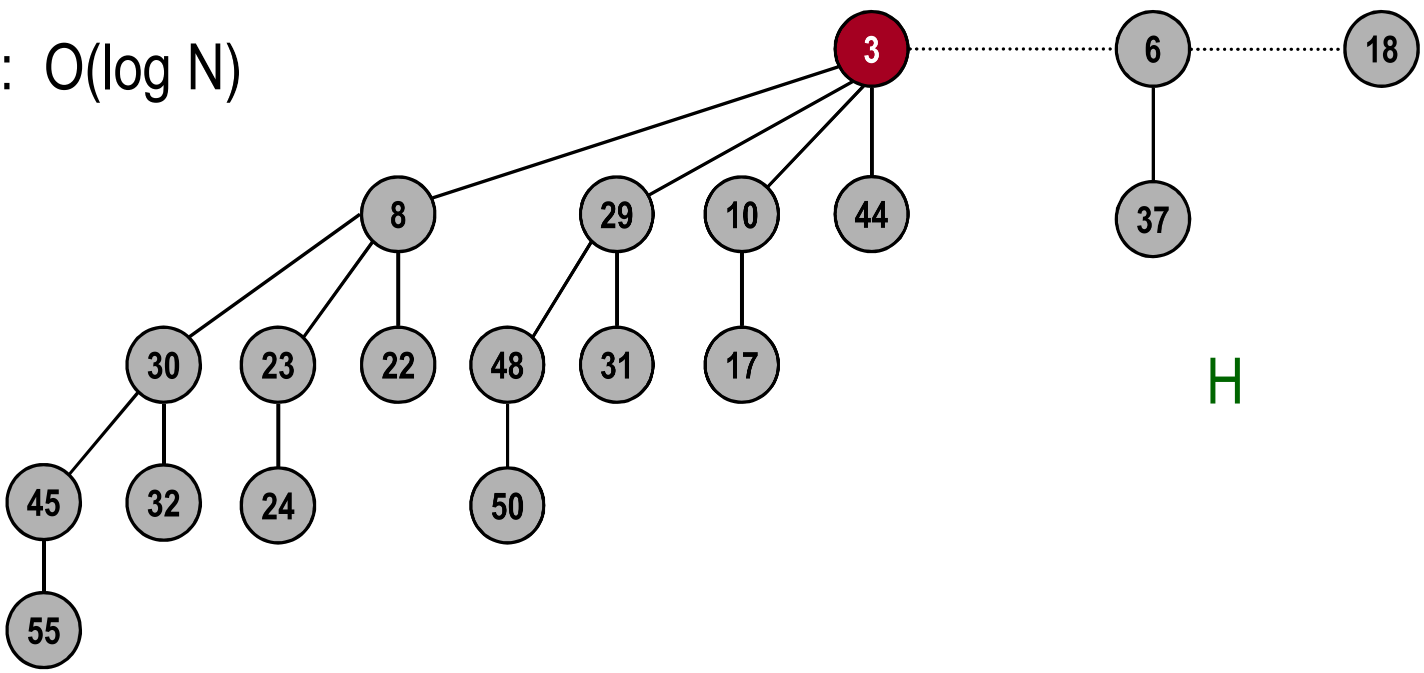
The running time is proportional to the number of trees in root lists, which is at most $2(\lfloor \log_2 N \rfloor + 1)$.

Delete Minimal

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)

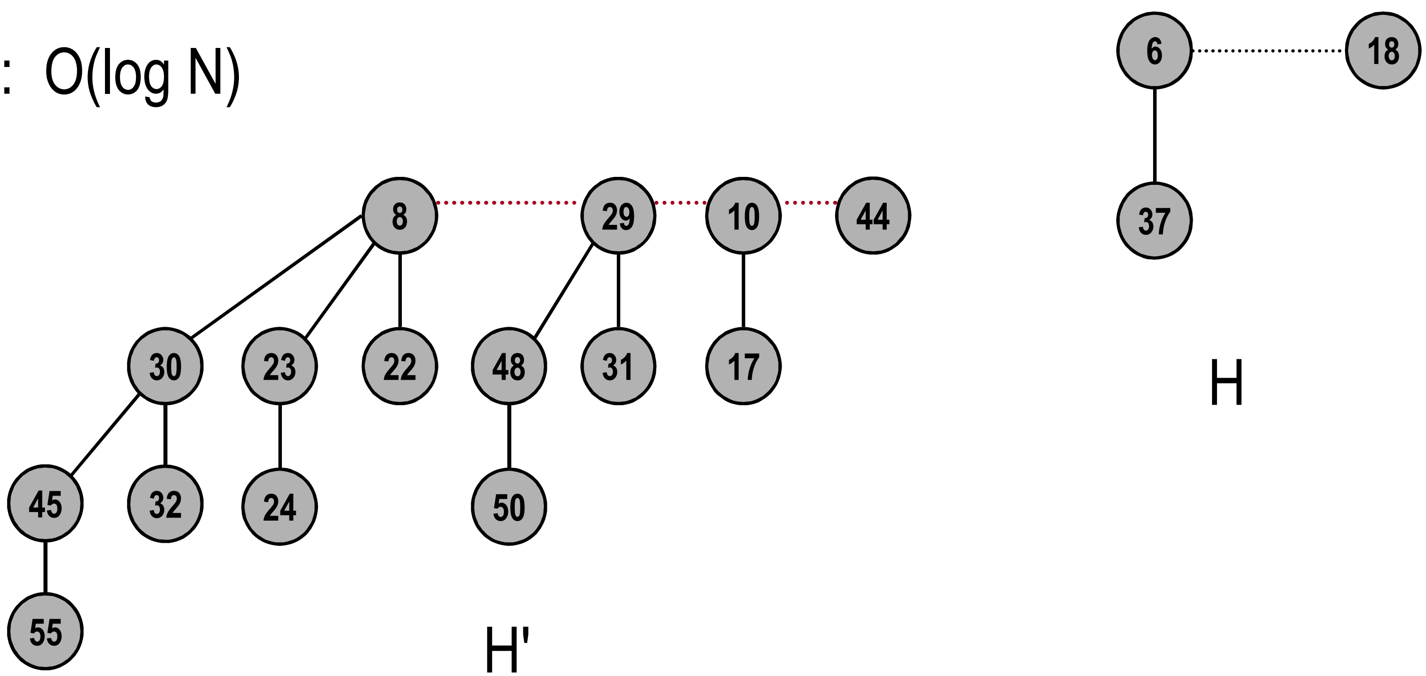


Delete Minimal

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)



Decrease Key

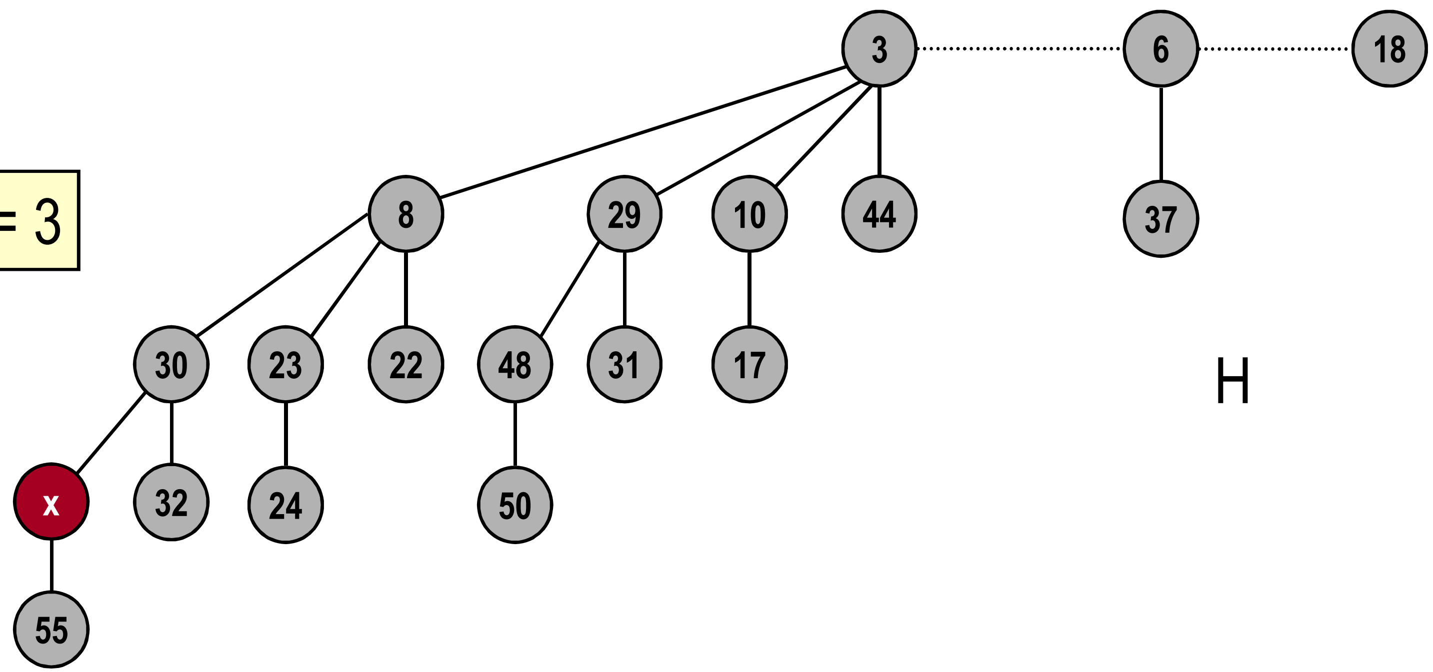
Decrease key of node x in binomial heap H.

- 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 $x \leq \lfloor \log_2 N \rfloor$.

depth = 3



Delete

Delete node x in binomial heap H.

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

Running time: O(log N)

Insert

Insert a new node x into binomial heap H.

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

Running time: O(log N)