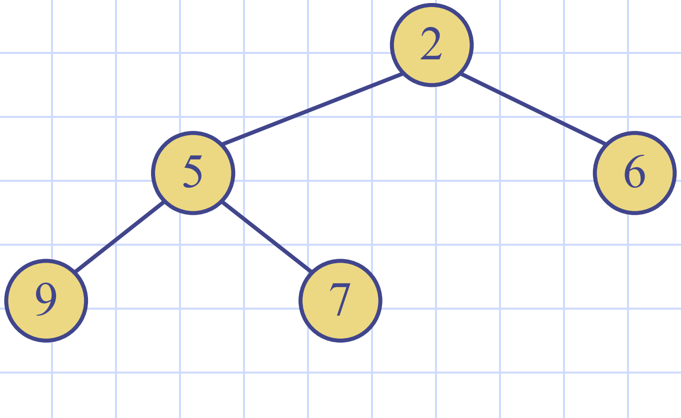
Heaps



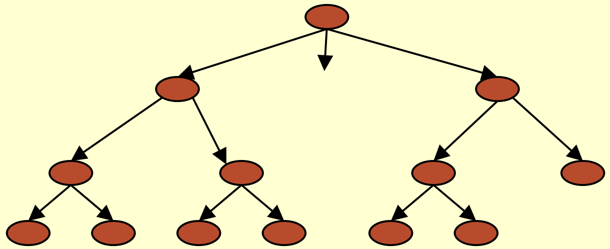
Definition of a heap

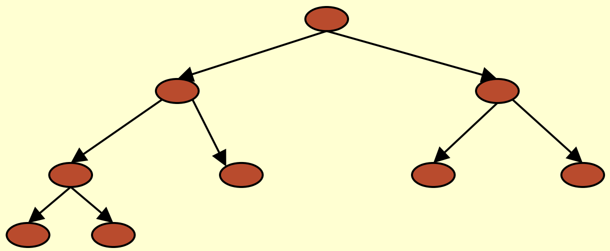
- A heap is a tree that satisfies the *heap property*.
  - *Minimum Heap Property:* The value stored at each node is less than or equal to the values stored at its children.
  - OR Maximum Heap Property: for greater

Heap **Structure** Property

- All levels but the last (leaf) are full and the leaf nodes are left oriented, i.e. filled from left to right.

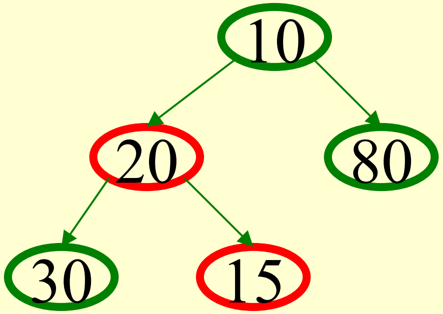
Examples:



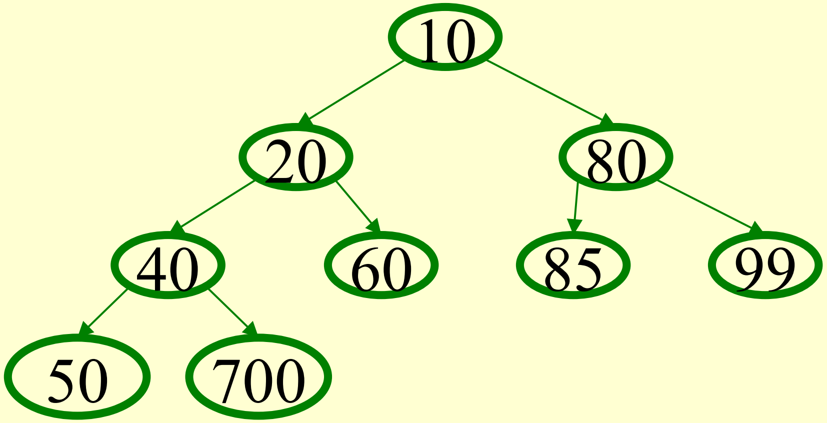


Heap **Order** Property

Heap order property: For every non-root node X, the value in the parent of X is less than (or equal to) the value in X.

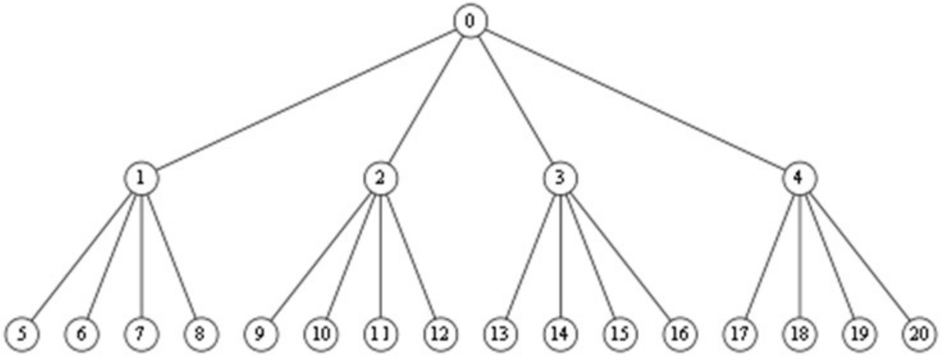


not a heap

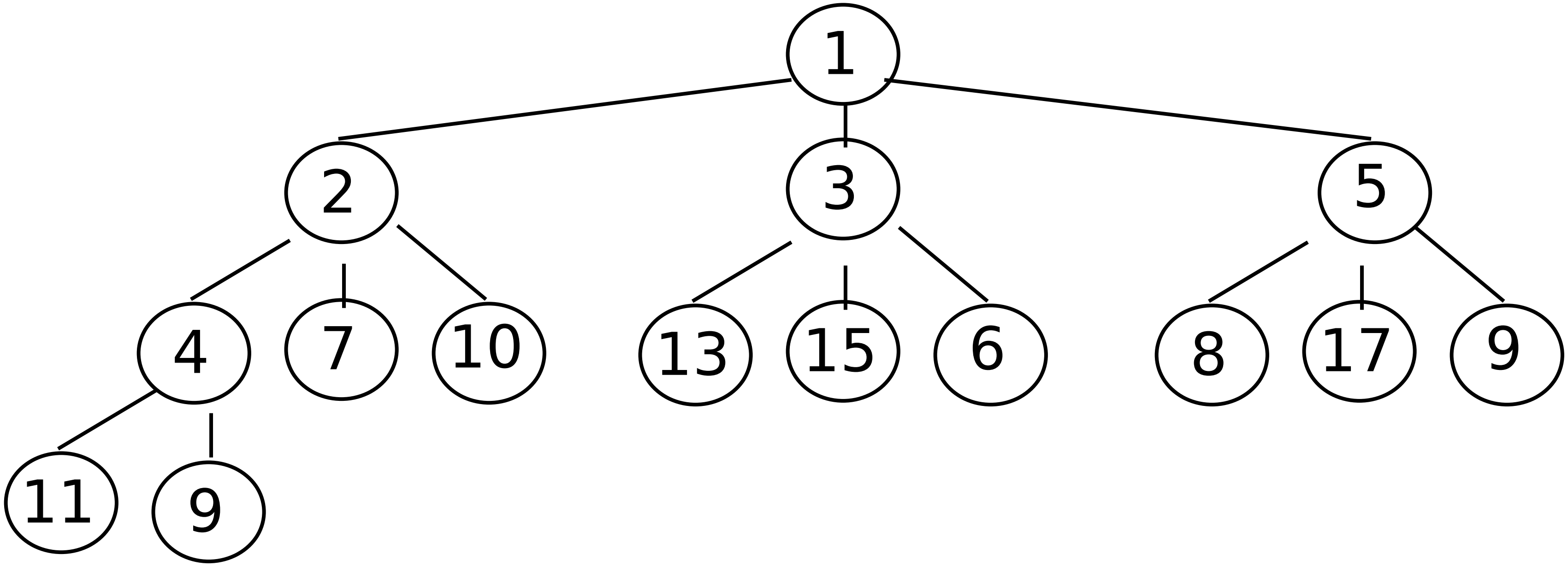


HEAPS

- **Example of a 4-heap**



d-Heap



An example of 3-heap

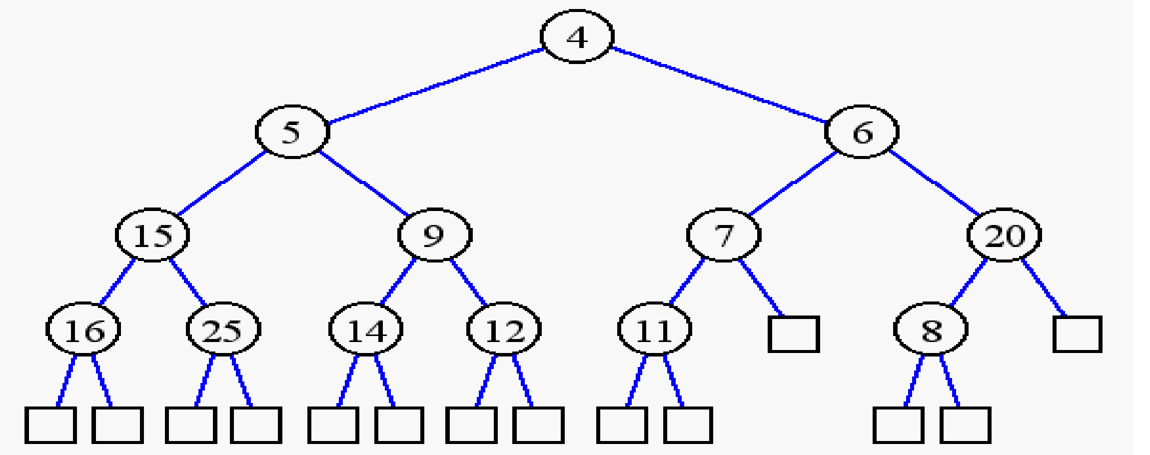
Definition of a heap

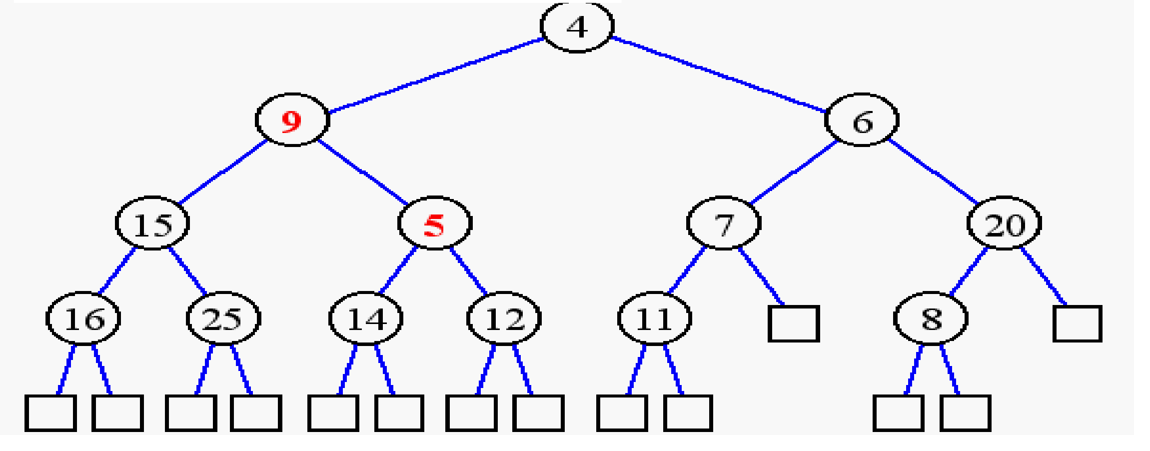
- **A heap That has D children is called D-Heap.**
- **The most popular heap is 2-Heap,**
  - **Also called binary Heap Tree**

Definition of a heap

- **Difference between Binary Search Tree and Binary Heap Tree**

Heap or Not a Heap?



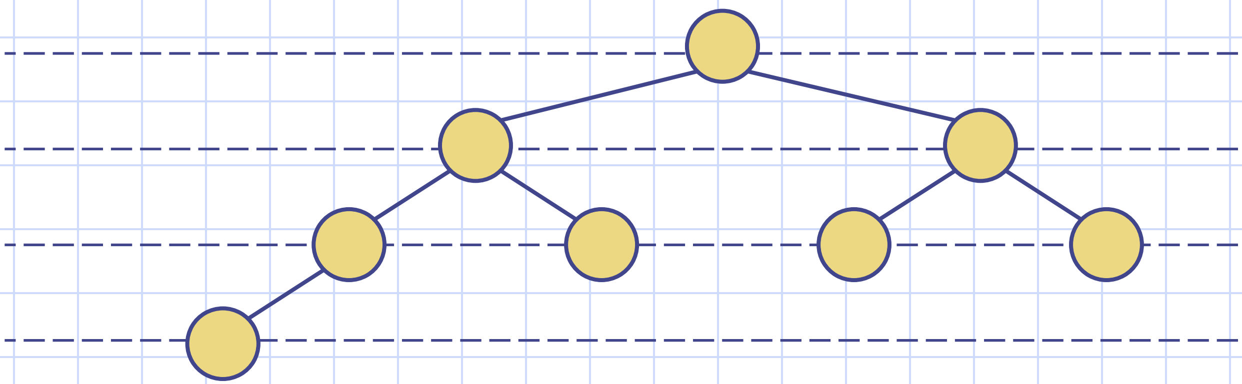


Height of a Heap

- Theorem: A heap storing n keys has height $O(\log n)$

  Proof: (we apply the complete binary tree property)
  - Let h be the height of a heap storing n keys
  - Since there are 2^i keys at depth $i = 0, \ldots, h - 1$ and at least one key at depth h, we have $n \geq 1 + 2 + 4 + \ldots + 2^{h-1} + 1$
  - Thus, $n \geq 2^h$, i.e., $h \leq \log n$

depth	keys
0	1
1	2
h-1	2^{h-1}
h	1

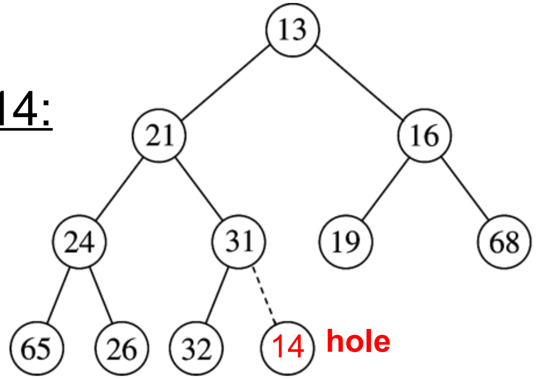


Operations on Heap

- Insert
- Delete-Min

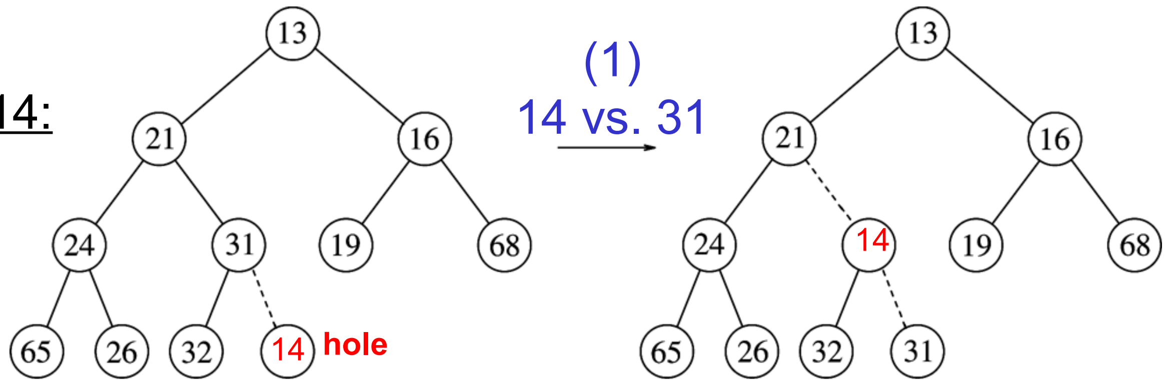
Heap Insert: Example

Insert 14:



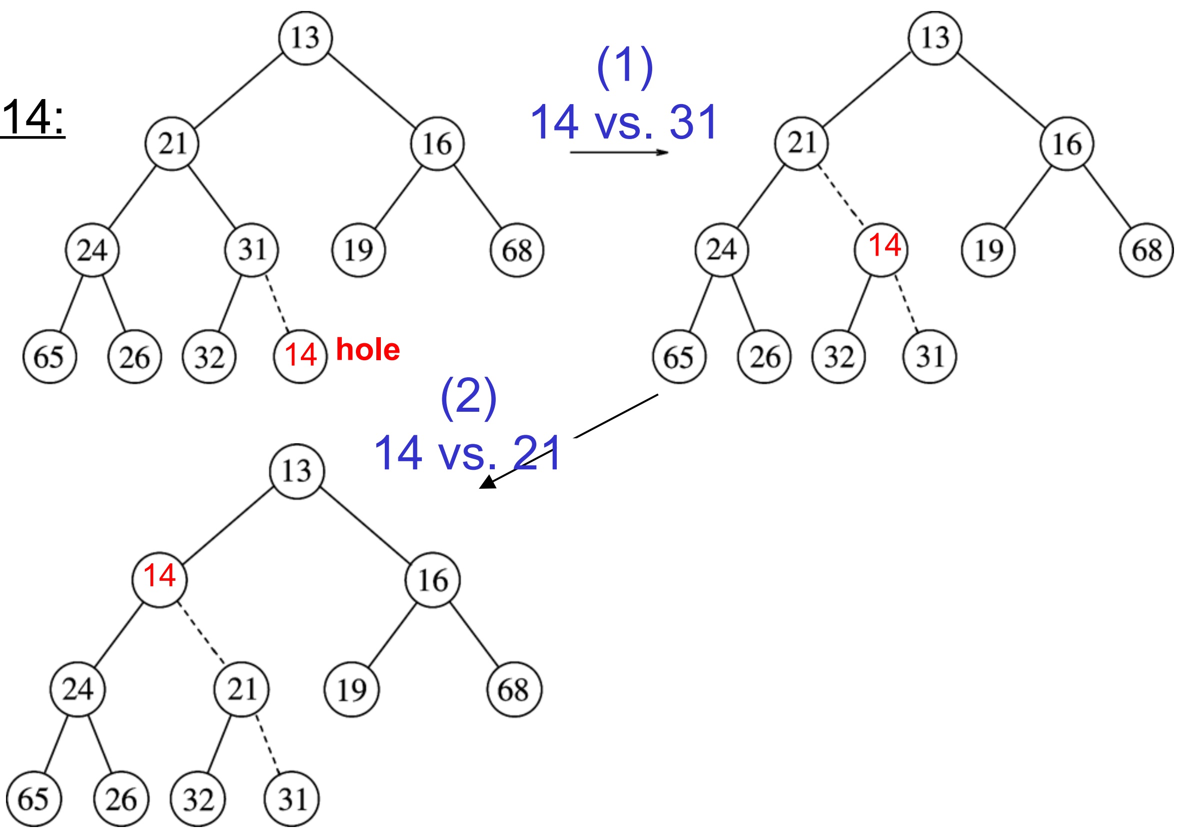
Heap Insert: Example

Insert 14:



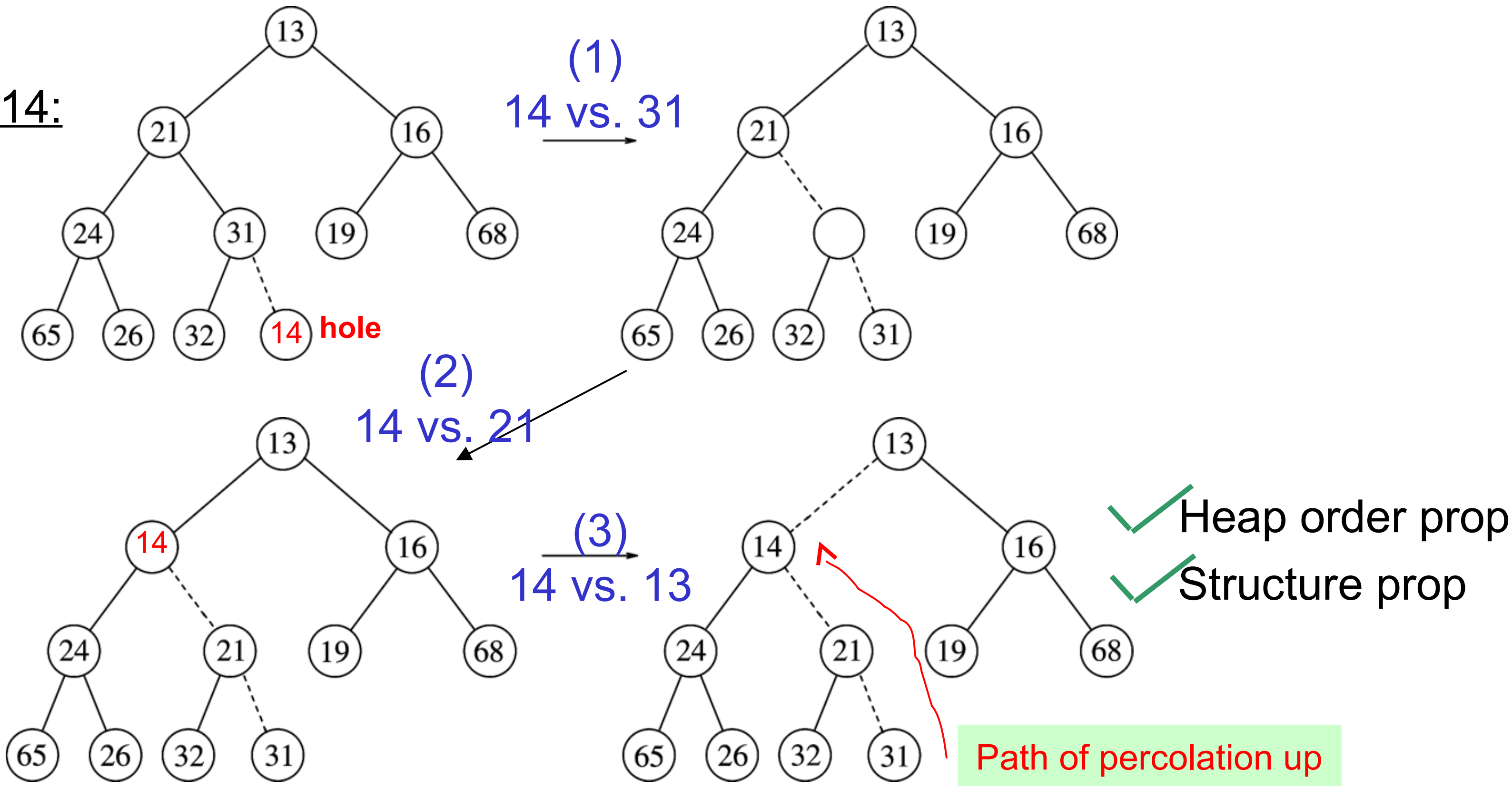
Heap Insert: Example

Insert 14:



Heap Insert: Example

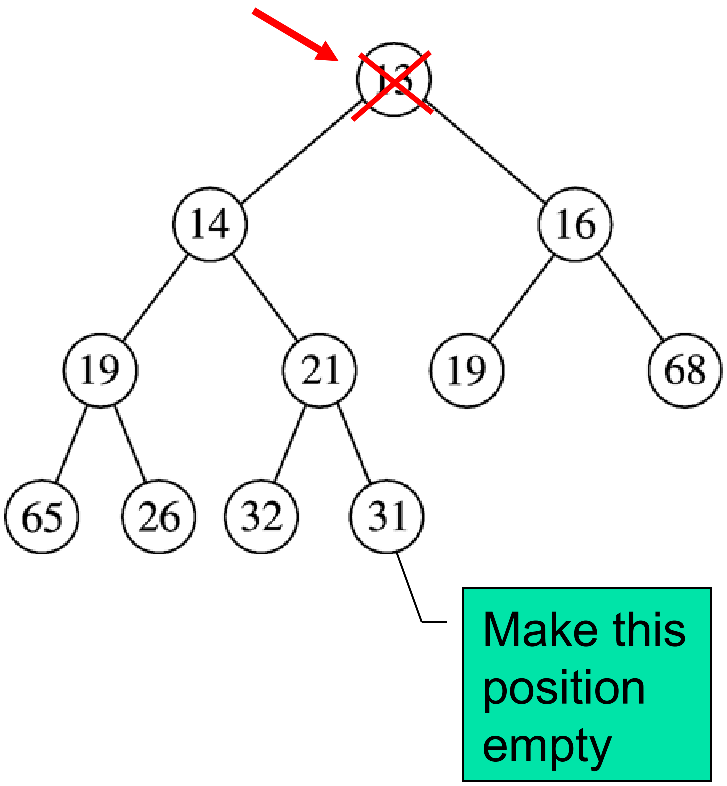
Insert 14:



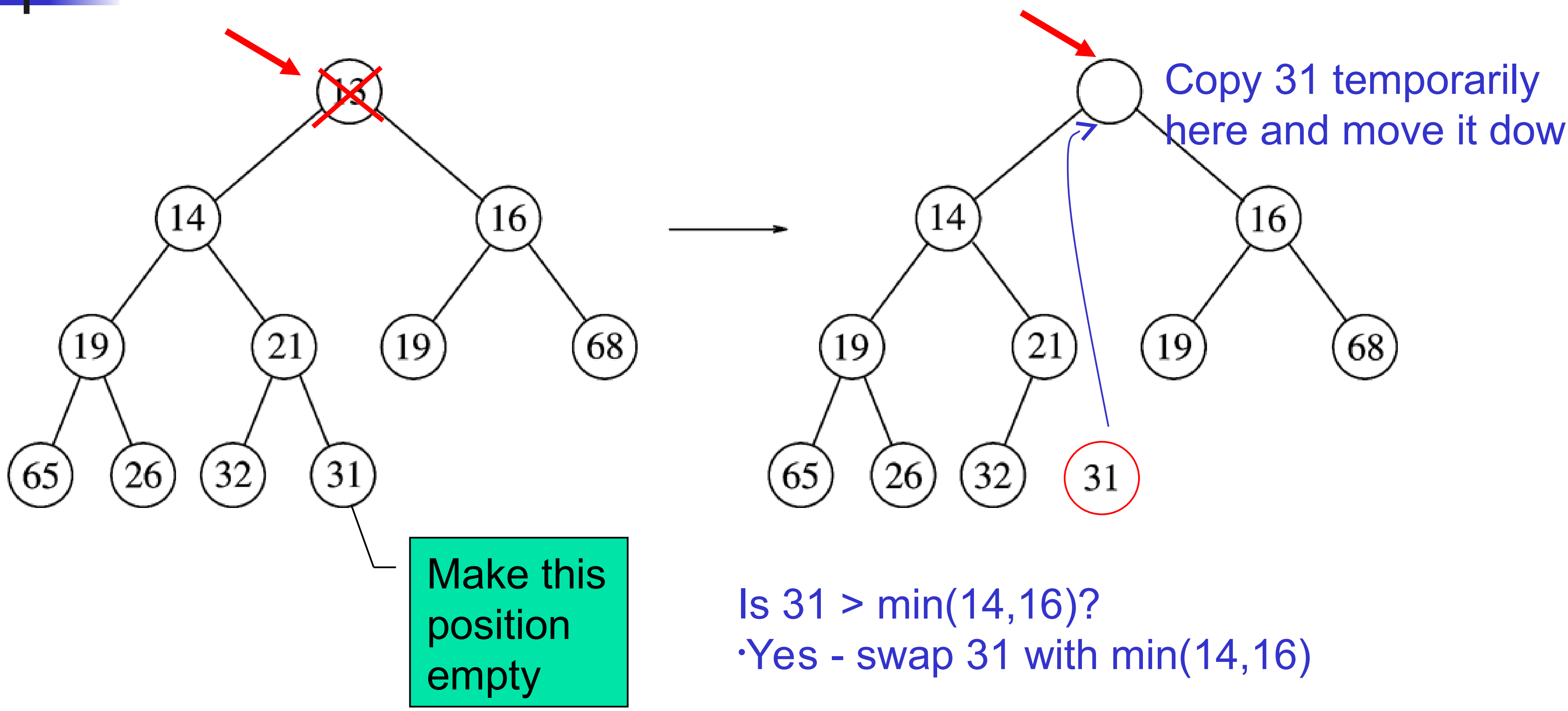
Heap DeleteMin

- Minimum element is always at the root
- Heap decreases by one in size
- Move last element into hole at root
- ***Percolate down*** while heap-order property not satisfied

Heap DeleteMin: Example

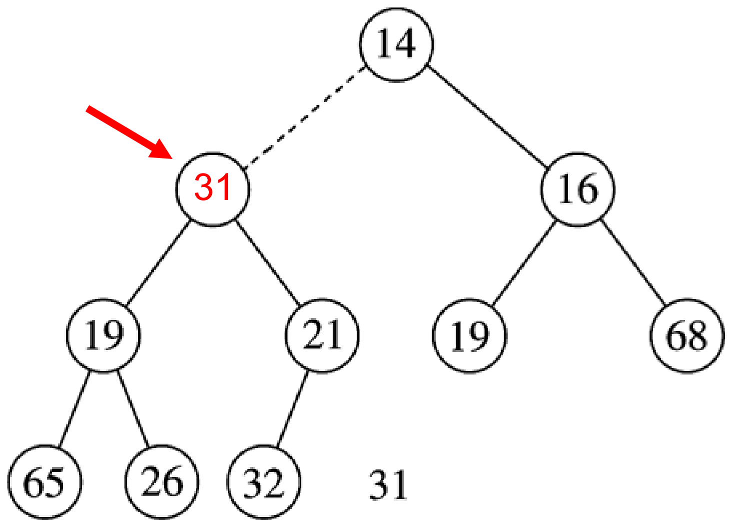


Heap DeleteMin: Example



Is 31 > min(14,16)?
·Yes - swap 31 with min(14,16)

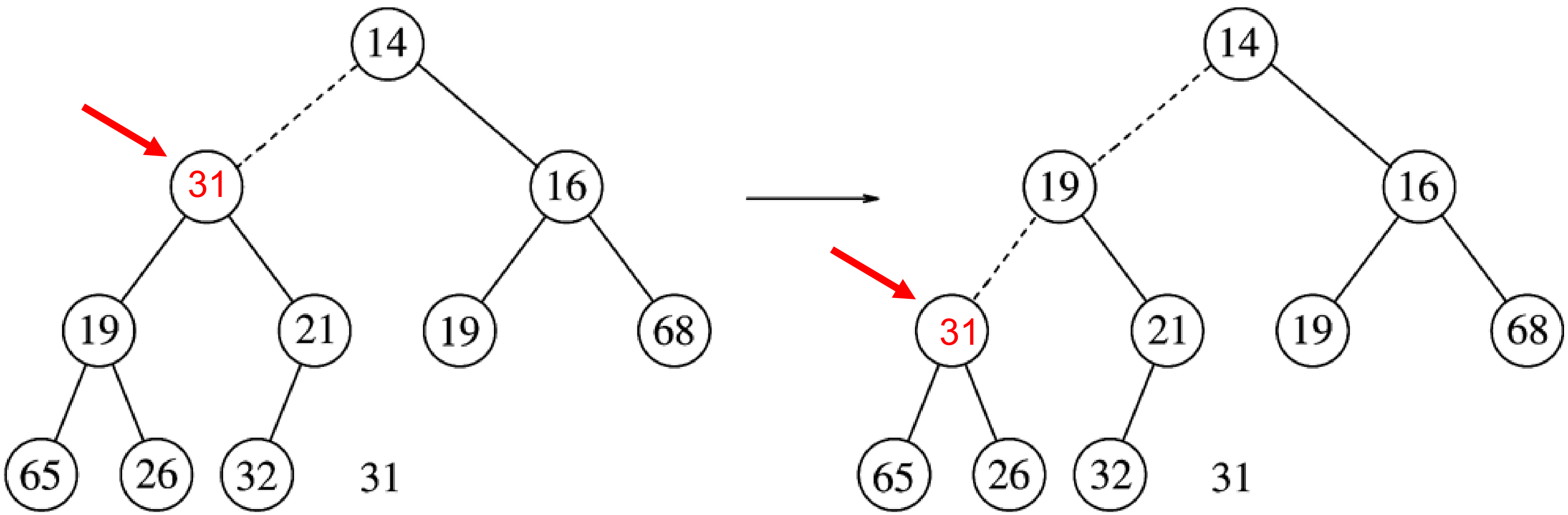
Heap DeleteMin: Example



Is 31 > min(19,21)?

- Yes - swap 31 with min(19,21)

Heap DeleteMin: Example

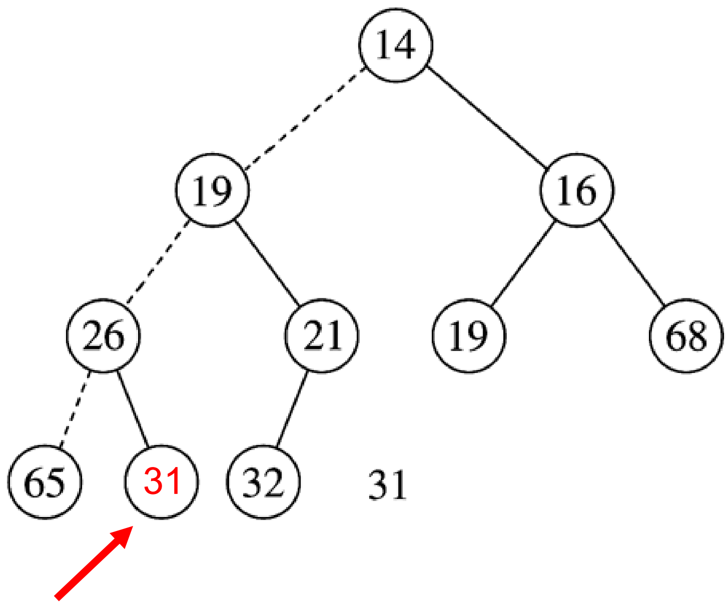


Is 31 > min(19,21)?
- Yes - swap 31 with min(19,21)

Is 31 > min(65,26)?
- Yes - swap 31 with min(65,26)

Percolating down…

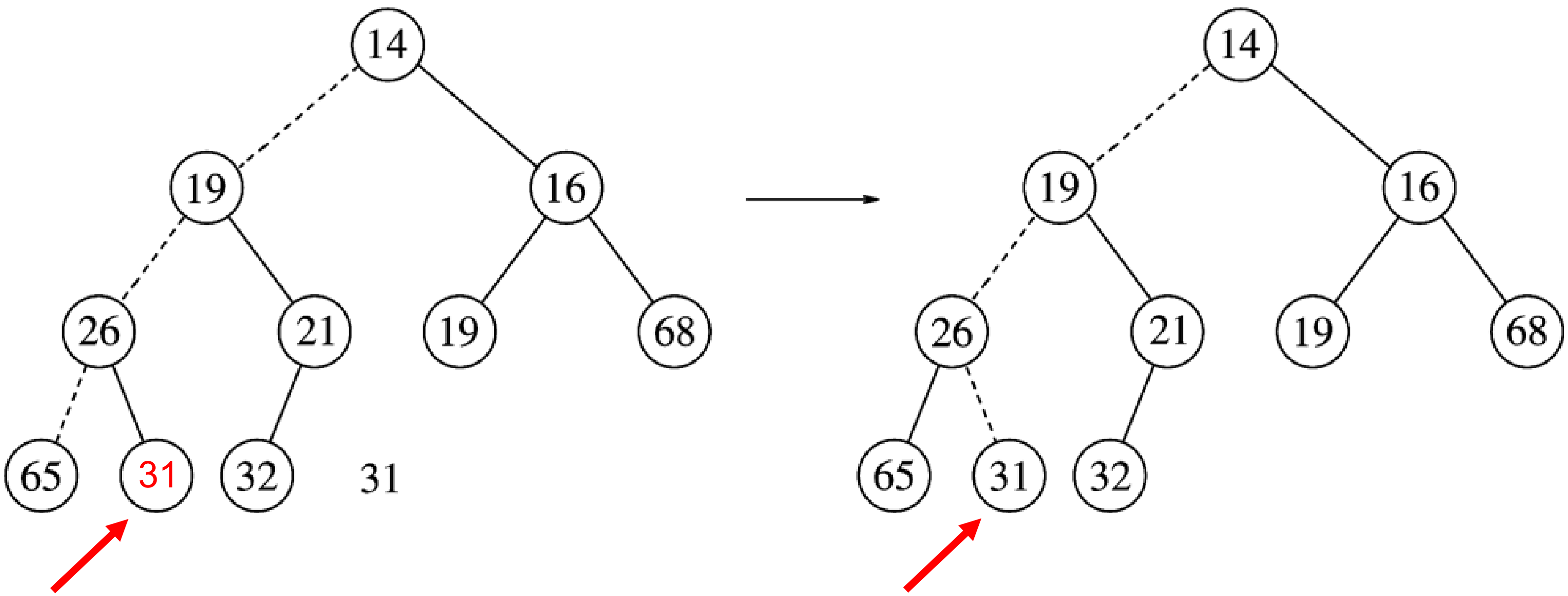
Heap DeleteMin: Example



Percolating down…

Percolating down…

Heap DeleteMin: Example

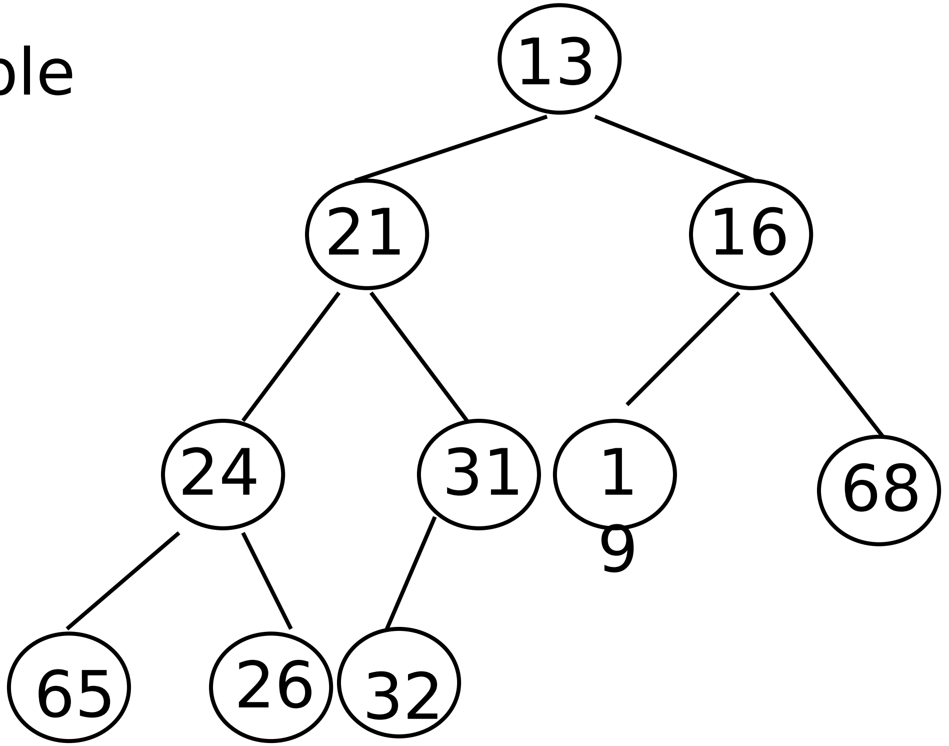


✓ Heap order prop

✓ Structure prop

Basic Heap Operations

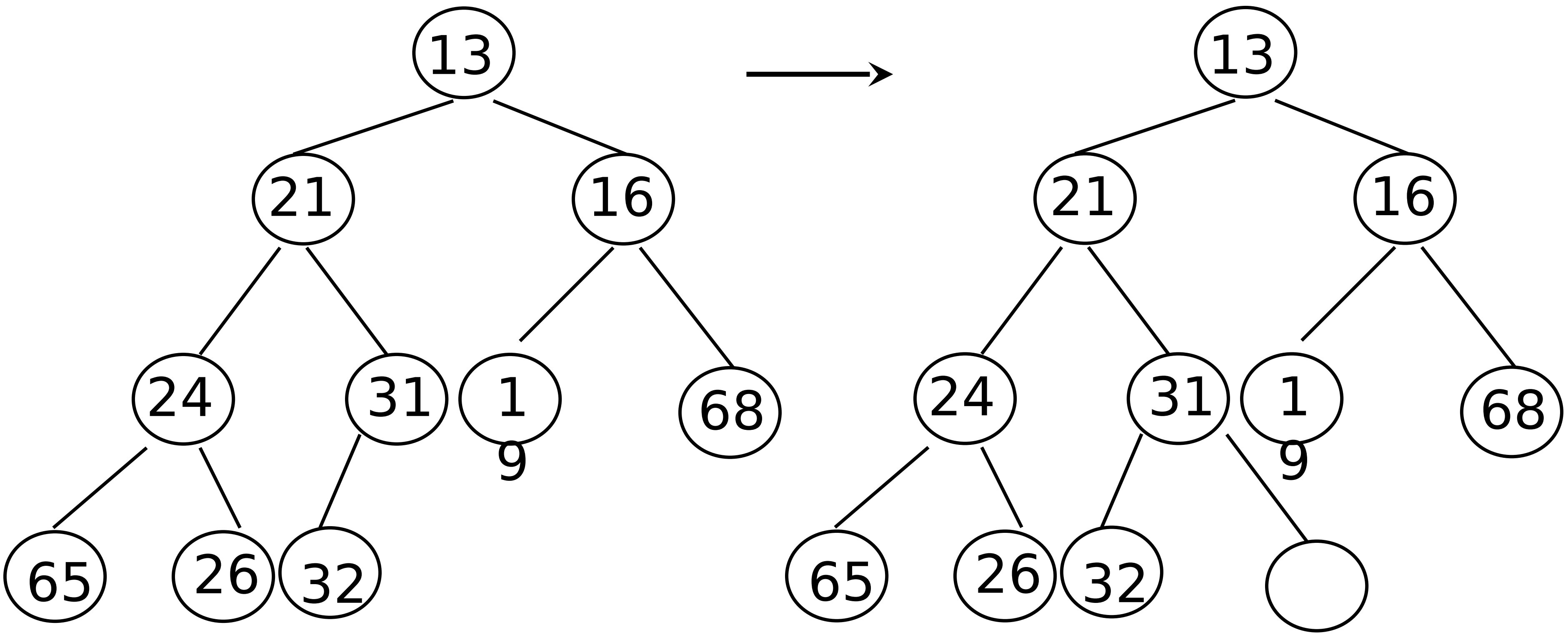
Example
:



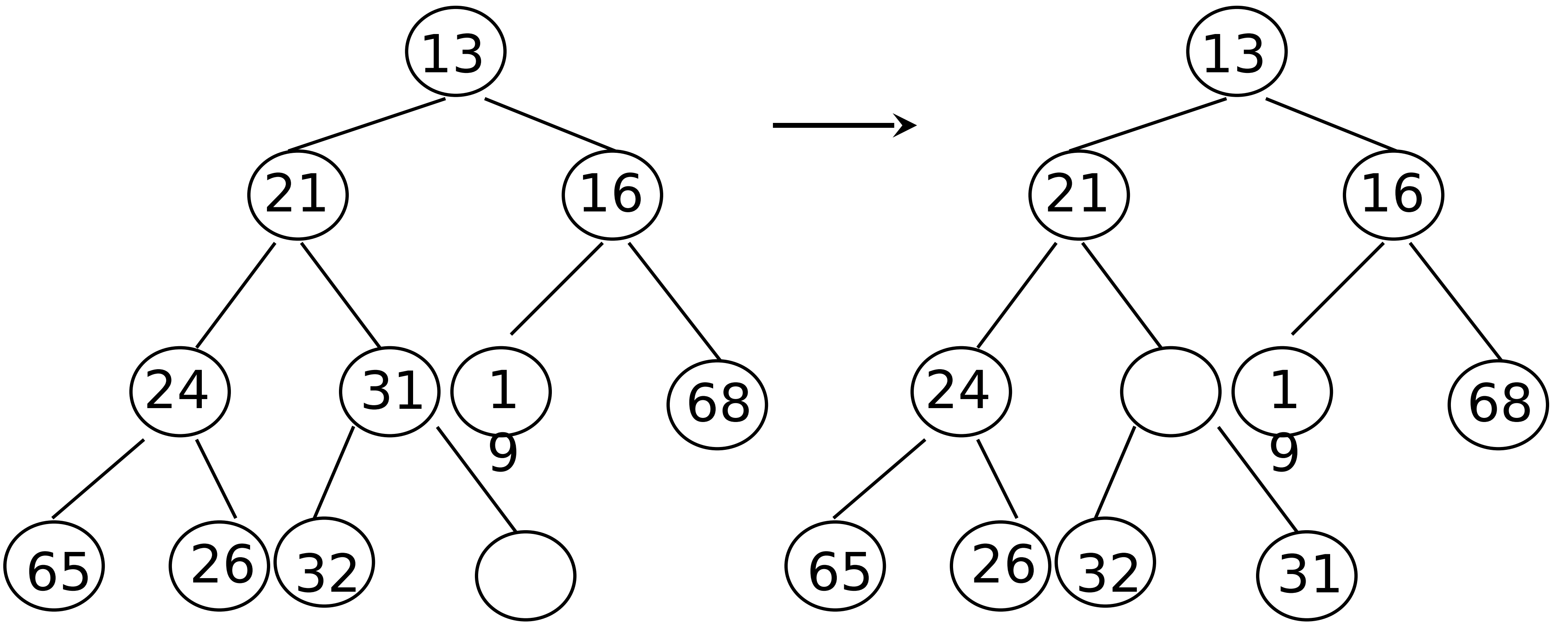
Insert 14

Original Tree

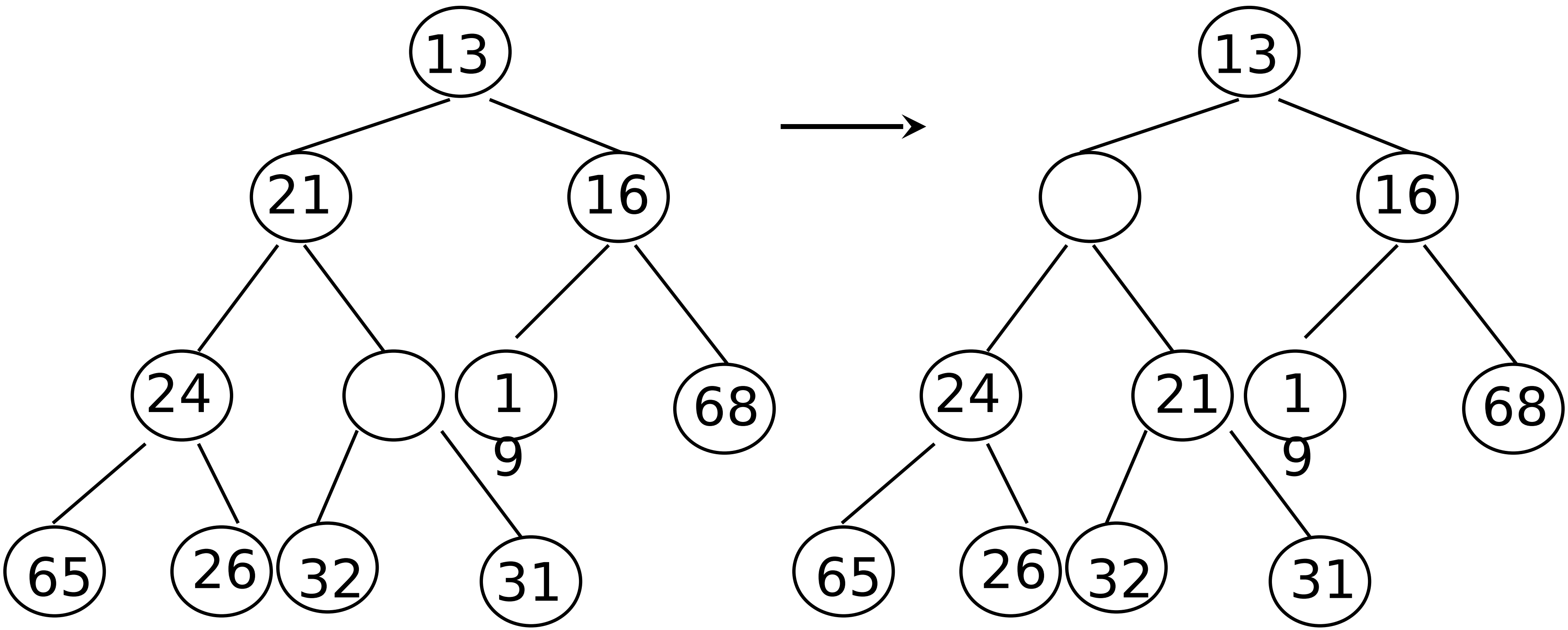
Basic Heap Operations



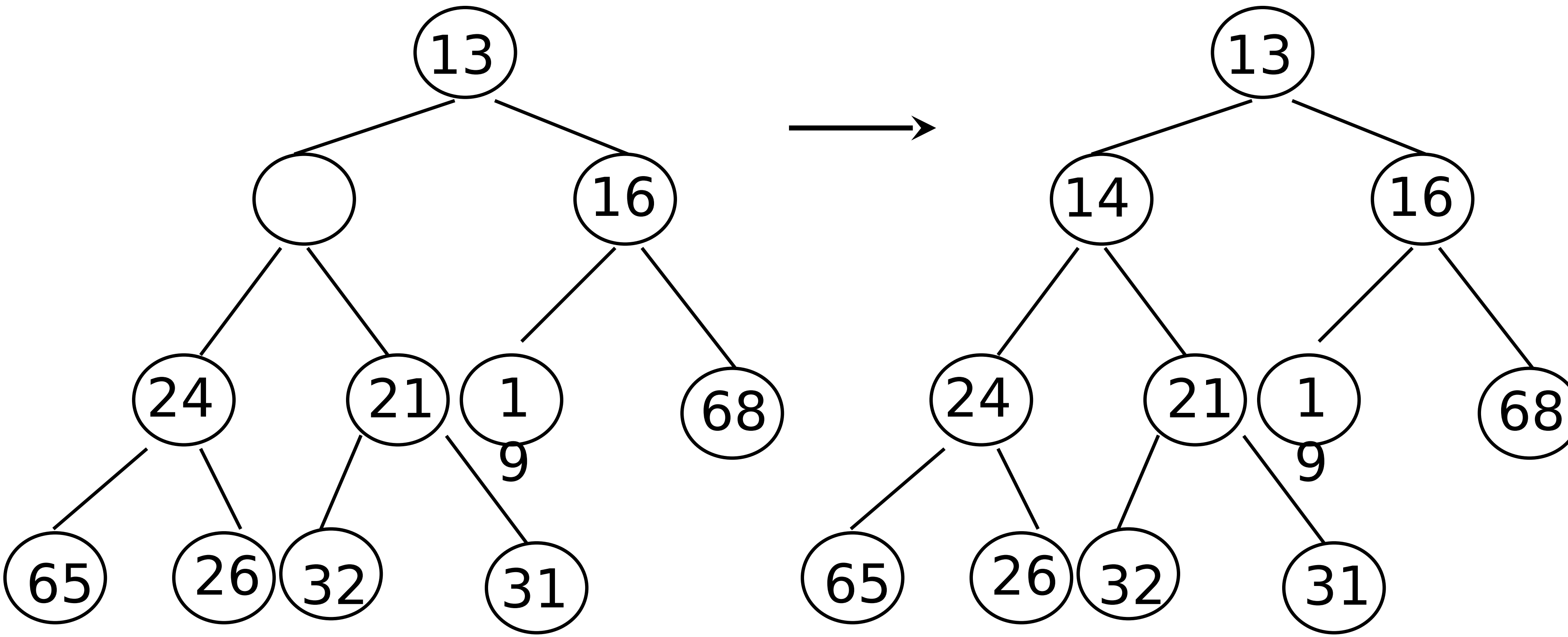
Basic Heap Operations



Basic Heap Operations

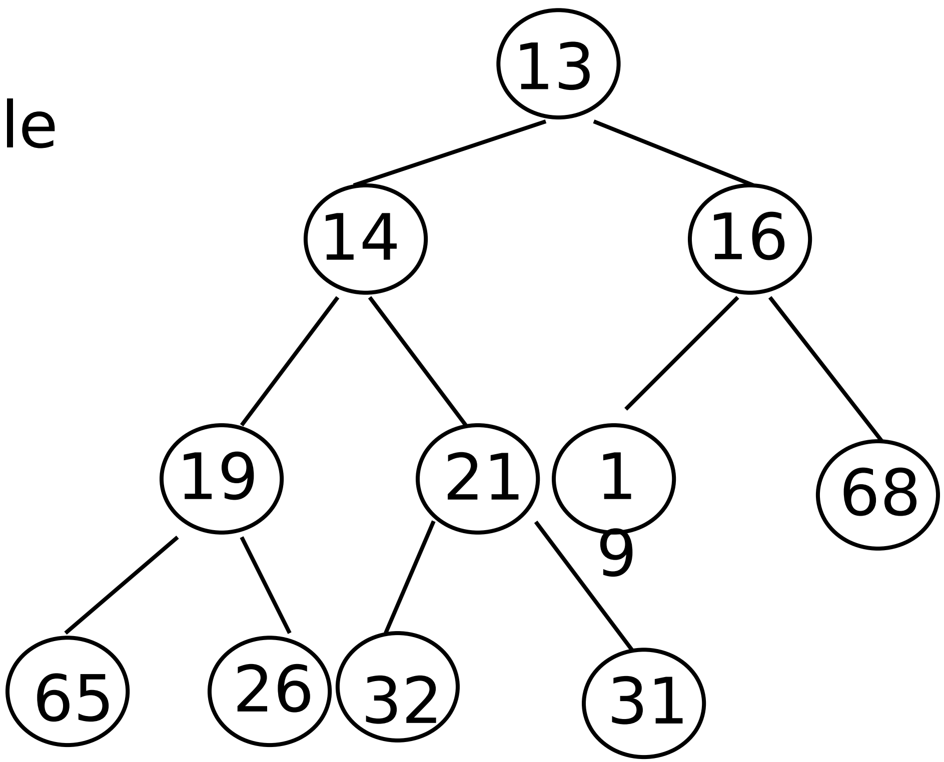


Basic Heap Operations



Basic Heap Operations

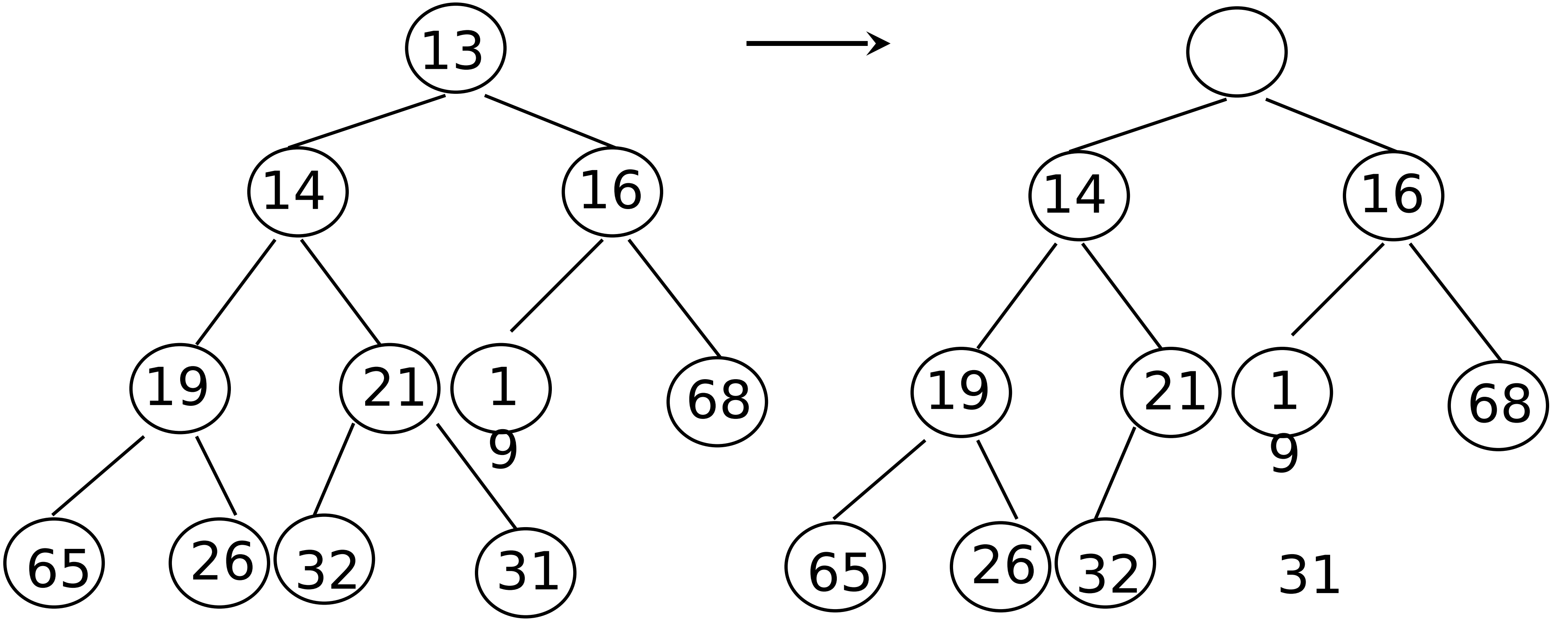
Example
:



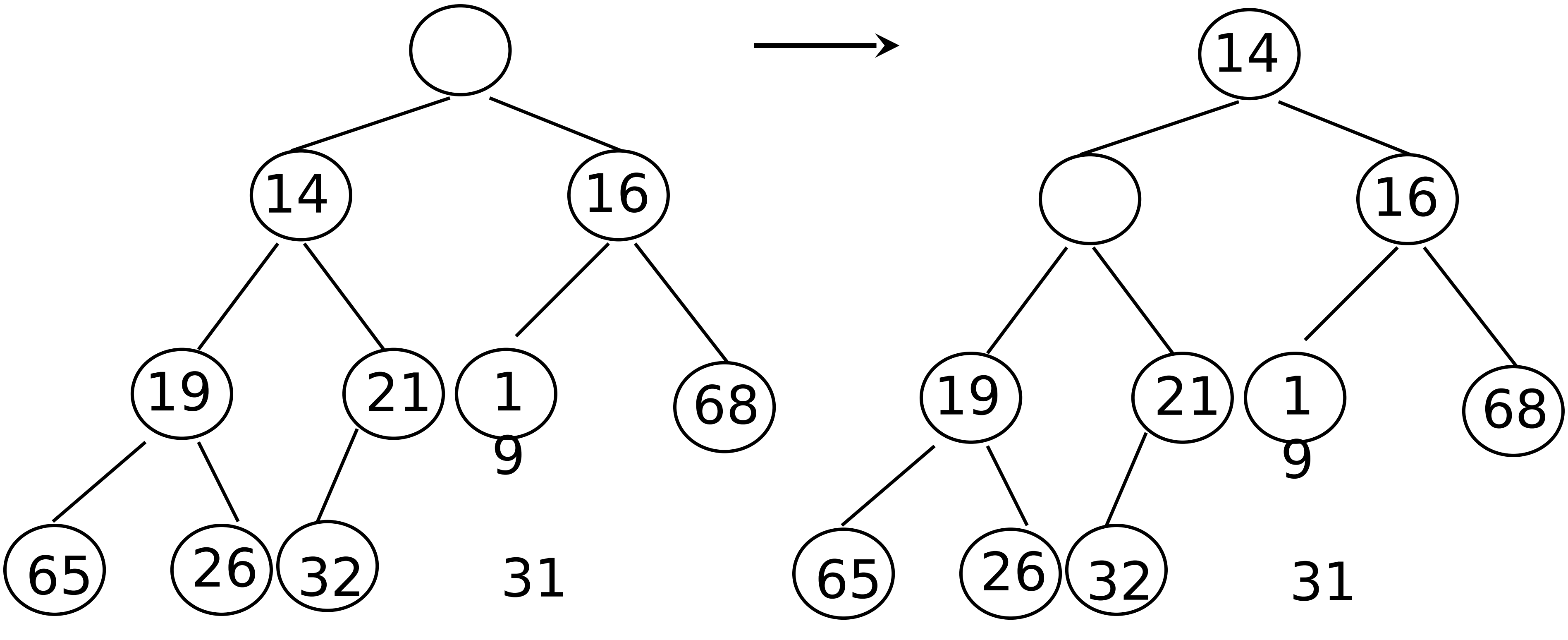
Delete the
minimum
key

Original Tree

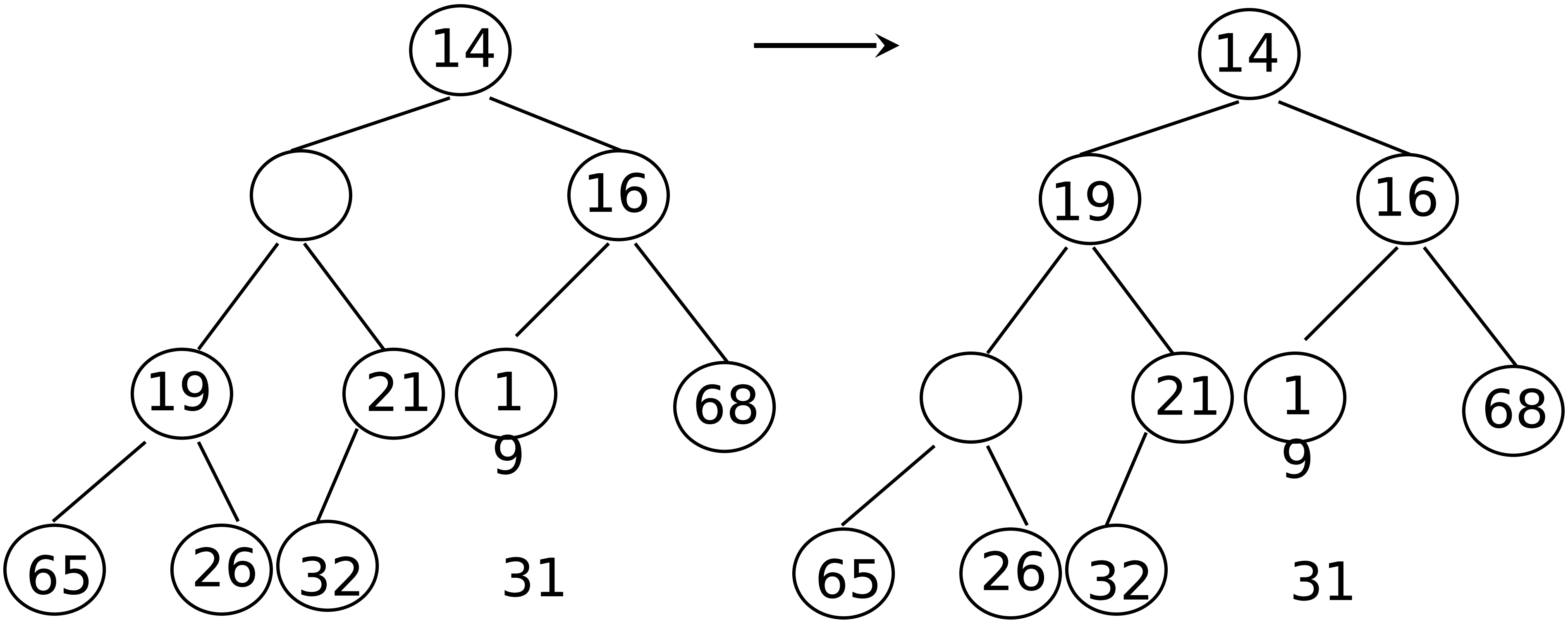
Basic Heap Operations



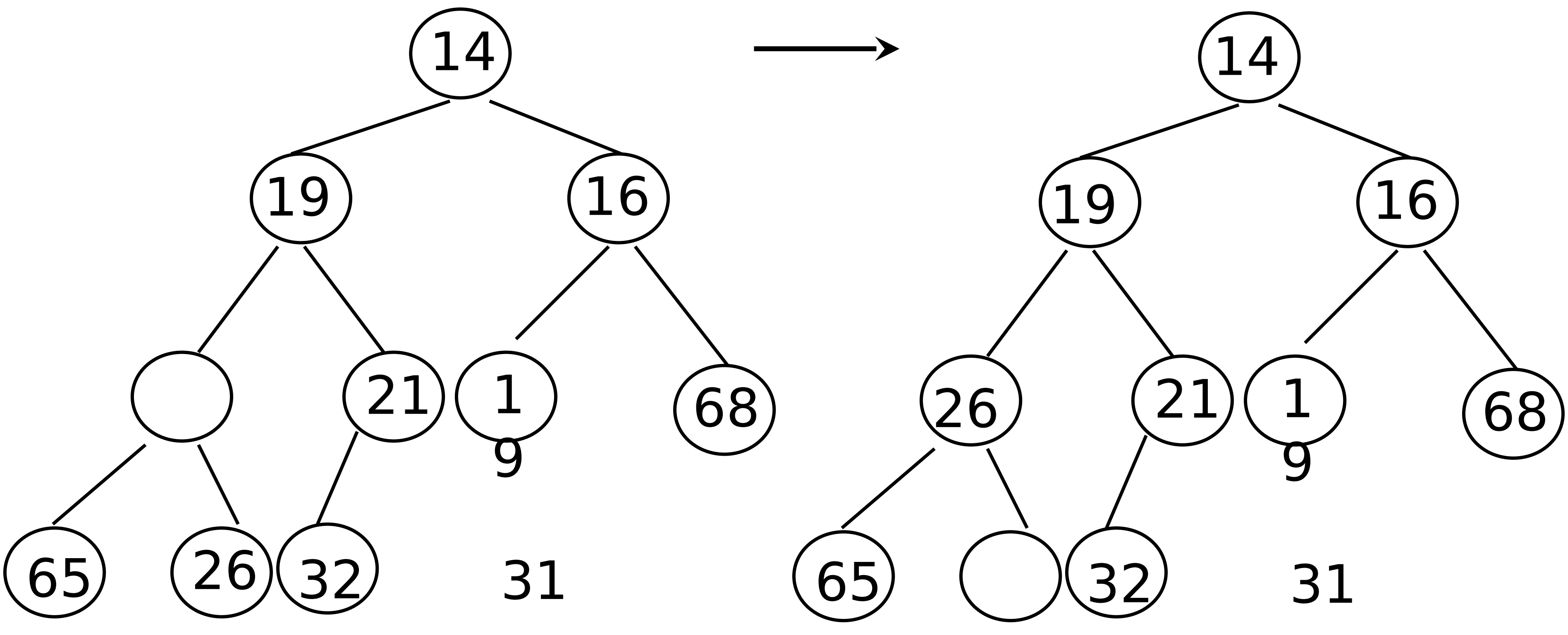
Basic Heap Operations



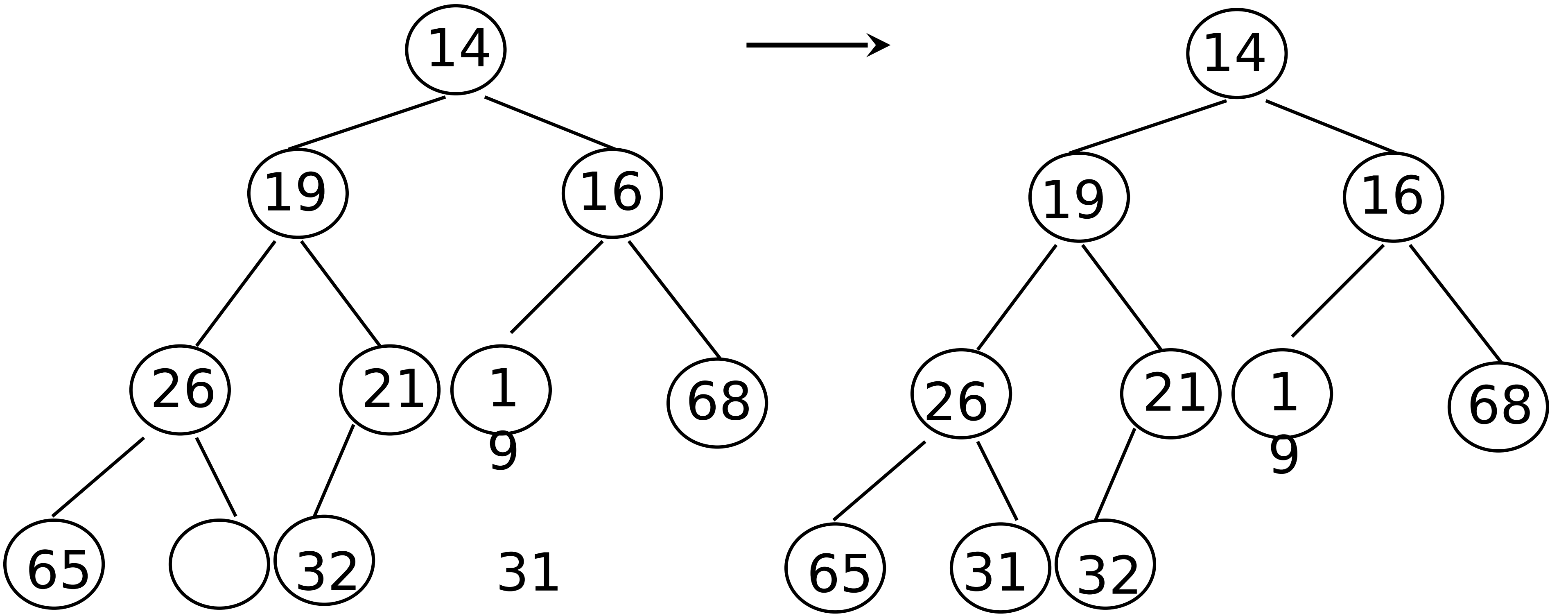
Basic Heap Operations



Basic Heap Operations



Basic Heap Operations



What are Heaps Useful For ?

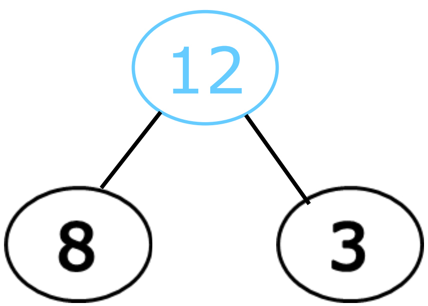
- To implement priority queues
- Priority queue = a queue where all elements have a “priority” associated with them
- Remove in a priority queue removes the element with the smallest priority
- insert
- removeMin

What are Heaps Useful For ?

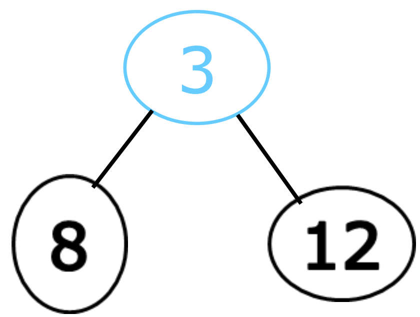
- A stack is first in, last out
- A queue is first in, first out
- A priority queue is least-first-out
- The “smallest” element is the first one removed
- The definition of “smallest” is up to the programmer (for example, you might define it by implementing Comparator or Comparable)
- If there are several “smallest” elements, the implementer must decide which to remove first
- Remove any “smallest” element (don’t care which)
- Remove the first one added

Heap Implementation o PQ

- A priority queue can be implemented as a heap
- In order to do this, we have to define the heap property
- In Heapsort, a node has the heap property if it is at least as large as its children
- For a priority queue, we will define a node to have the heap property if it is as least as small as its children (since we are using smaller numbers to represent higher priorities)



Heapsort: Blue node has the heap property



Priority queue: Blue node has the heap property

Heaps and Priority Queues

- We can use a heap to implement a priority queue
- We store a (key, element) item at each internal node
- We keep track of the position of the last node