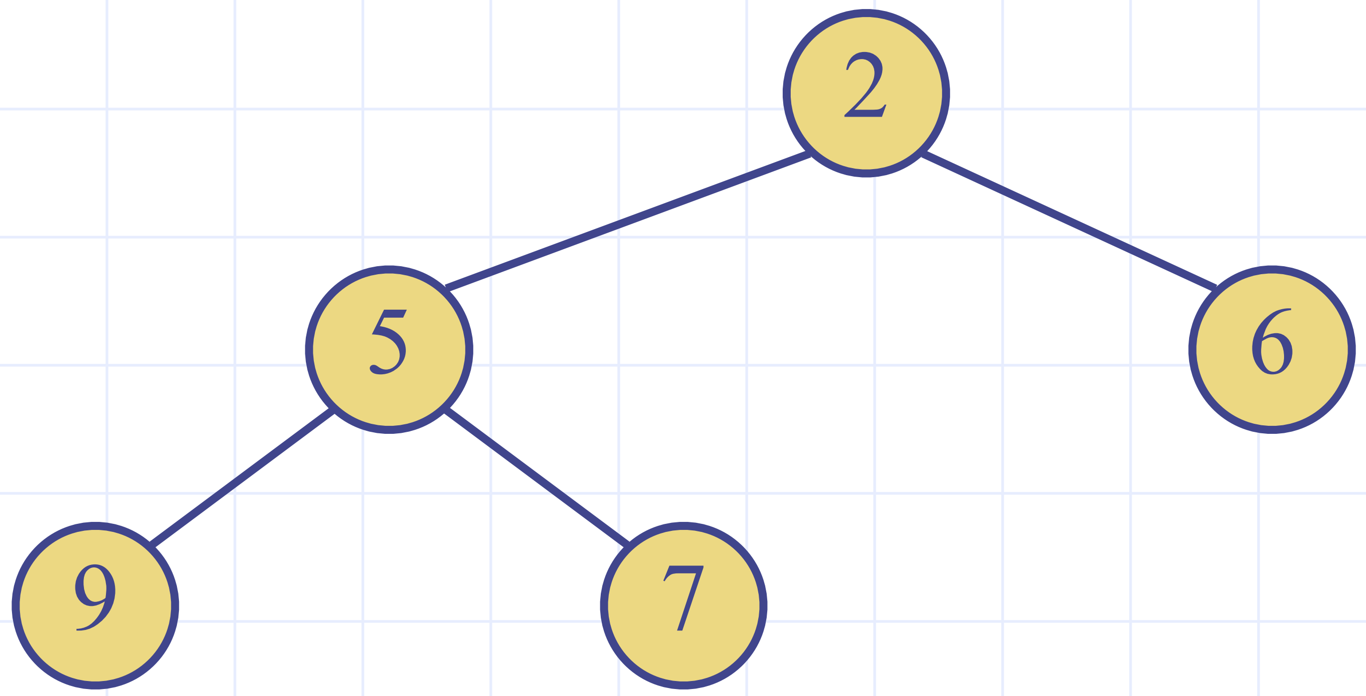
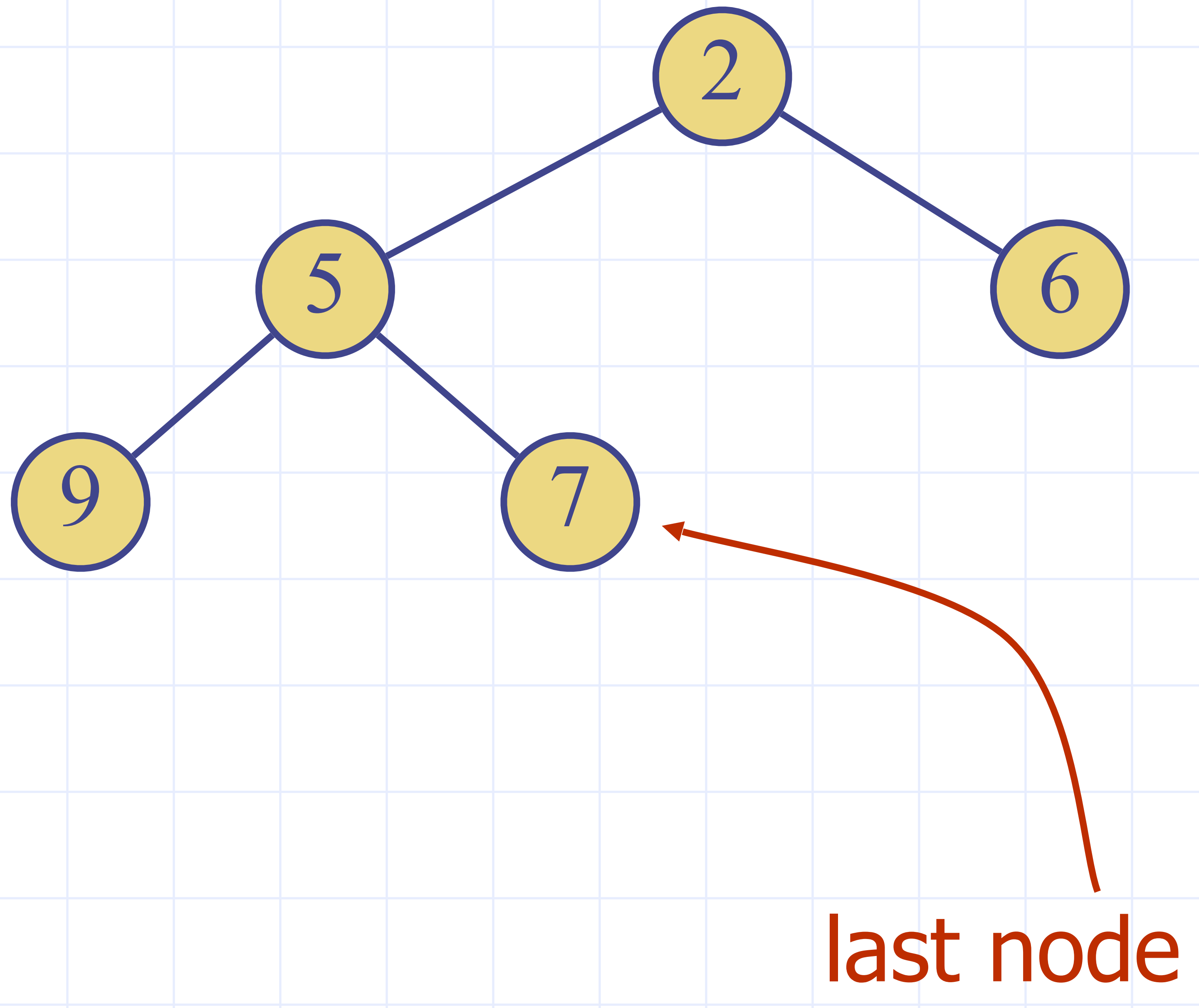
Heaps & Heap-Sort



Heaps

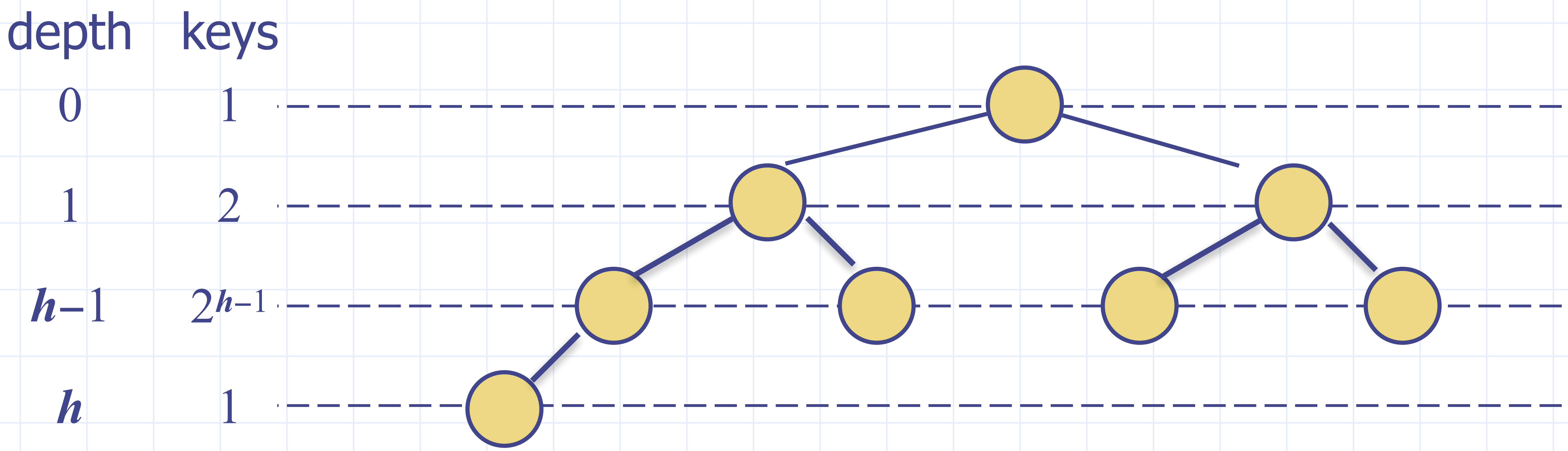
- A heap is a binary tree storing keys at its nodes and satisfying the following properties:
  - Heap-Order: for every internal node v other than the root,
    $key(v) \geq key(parent(v))$
  - Complete Binary Tree: let h be the height of the heap
    - for $i = 0, \ldots, h - 1$, there are 2^i nodes of depth i
    - at depth $h - 1$, all internal nodes are to the left of any external nodes
    - at depth h, all nodes are external and as far left as possible
- The last node of a heap is the rightmost node of maximum depth



Height of a Heap

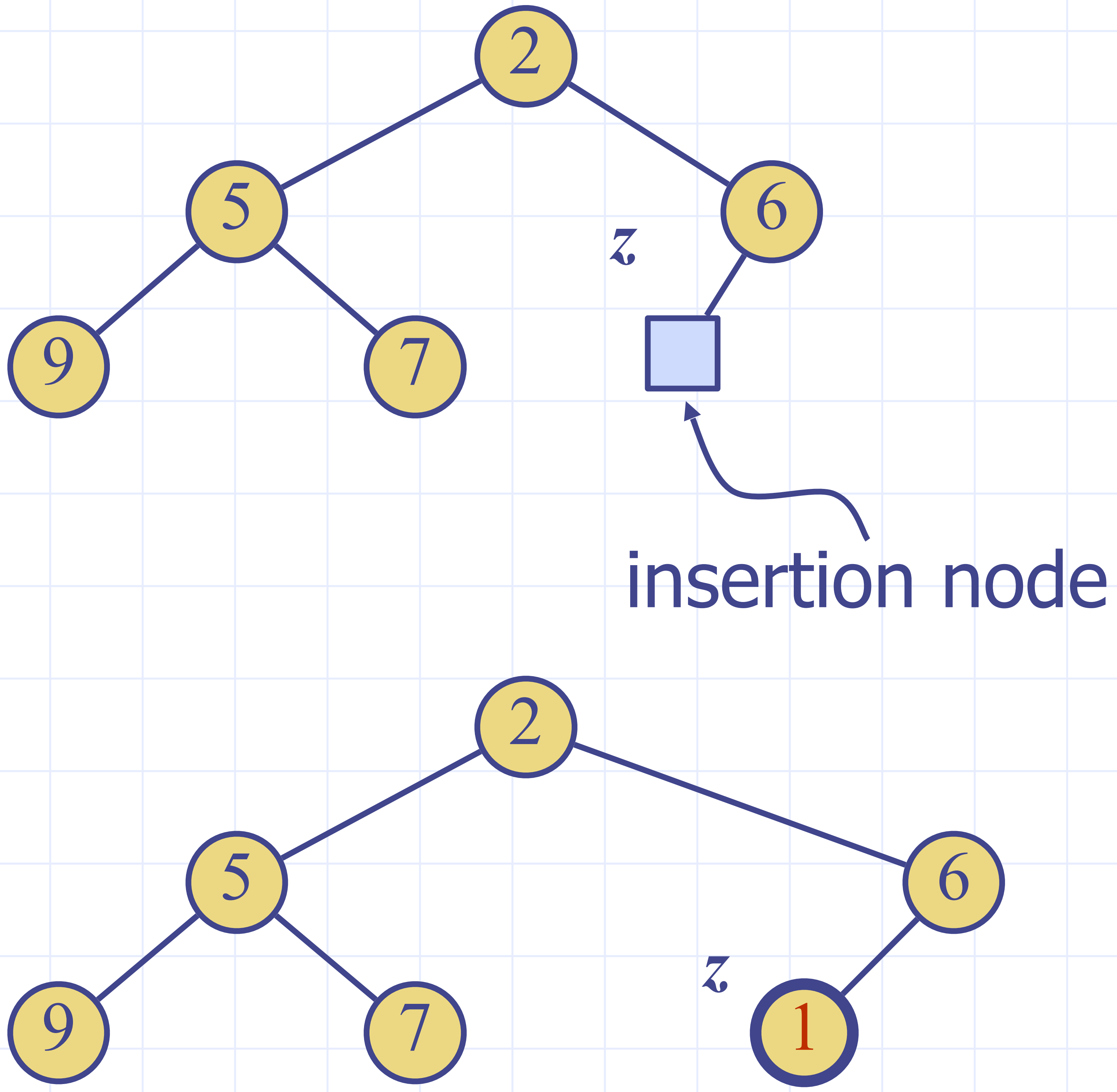
- **Theorem:** A heap storing n keys has height $O(\log n)$

  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, $n \geq 2^h$ … so … $h \leq \log n$



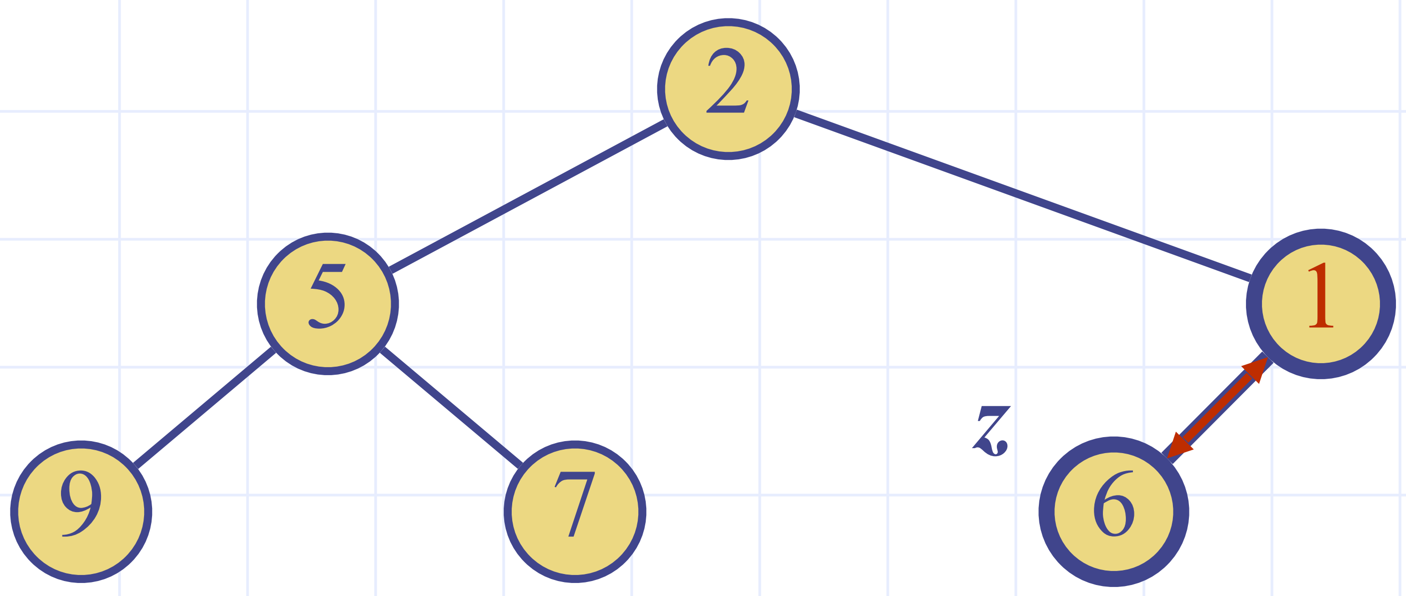
Insertion into a Heap

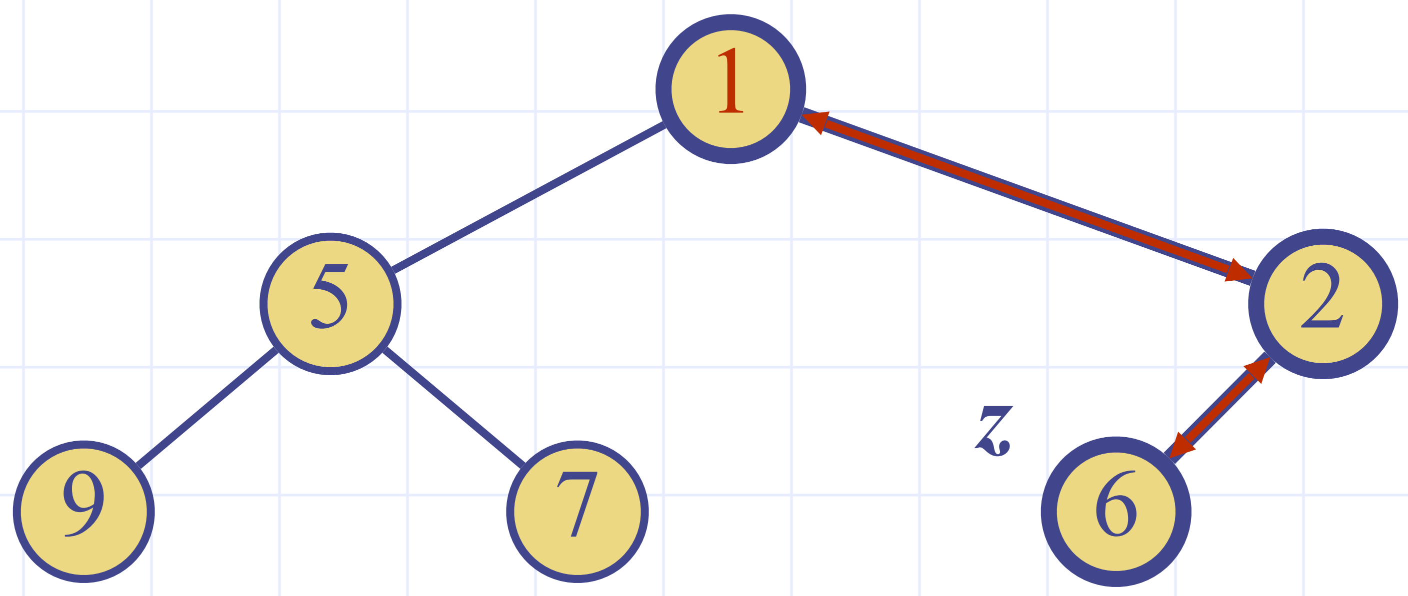
- insertion of a key k to the heap
- The insertion algorithm consists of three steps
  - Find the insertion node z (the new last node)
  - Store k at z
  - Restore the heap-order property (discussed next)



Upheap

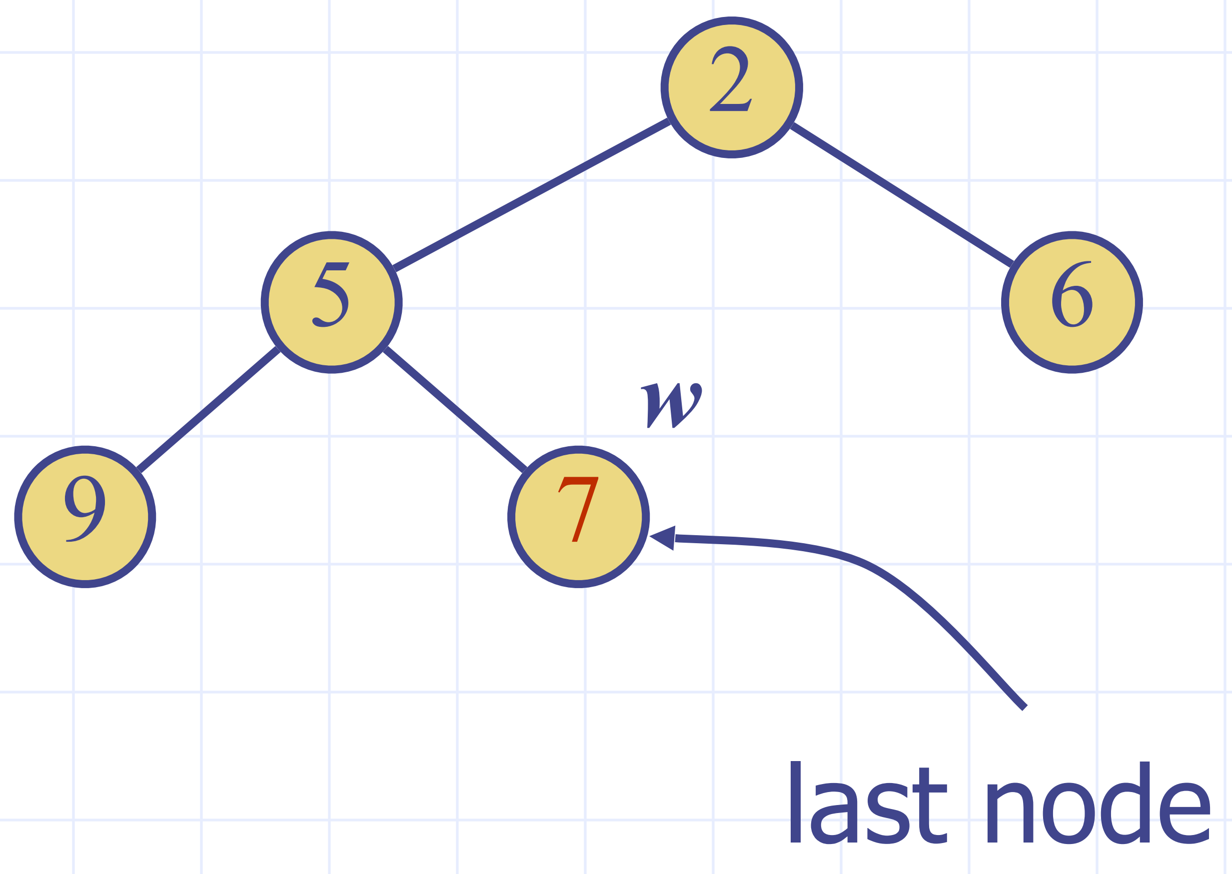
- After the insertion of a new key k, the heap-order property may be violated
- Algorithm upheap restores the heap-order property by swapping k along an upward path from the insertion node
- Upheap terminates when the key k reaches the root or a node whose parent has a key smaller than or equal to k
- Since a heap has height $O(\log n)$, upheap runs in $O(\log n)$ time

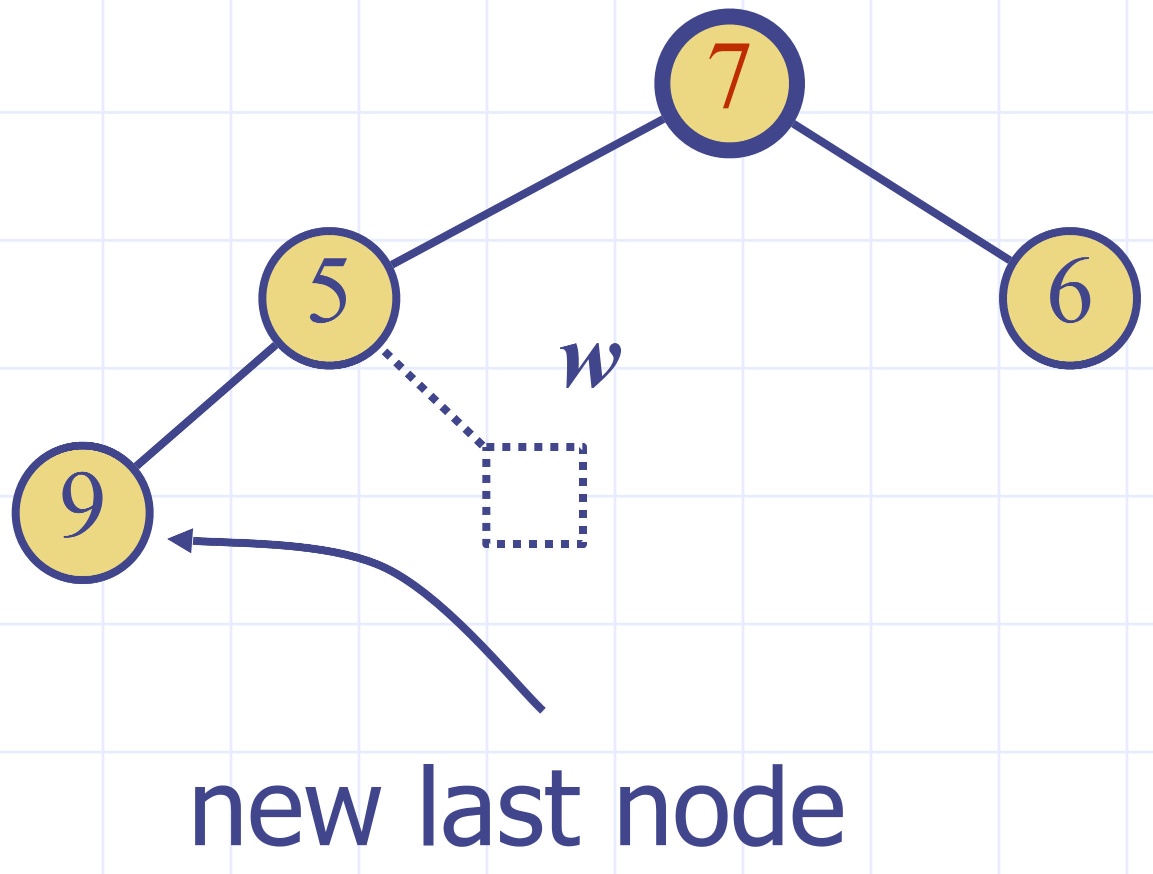




Removal from a Heap

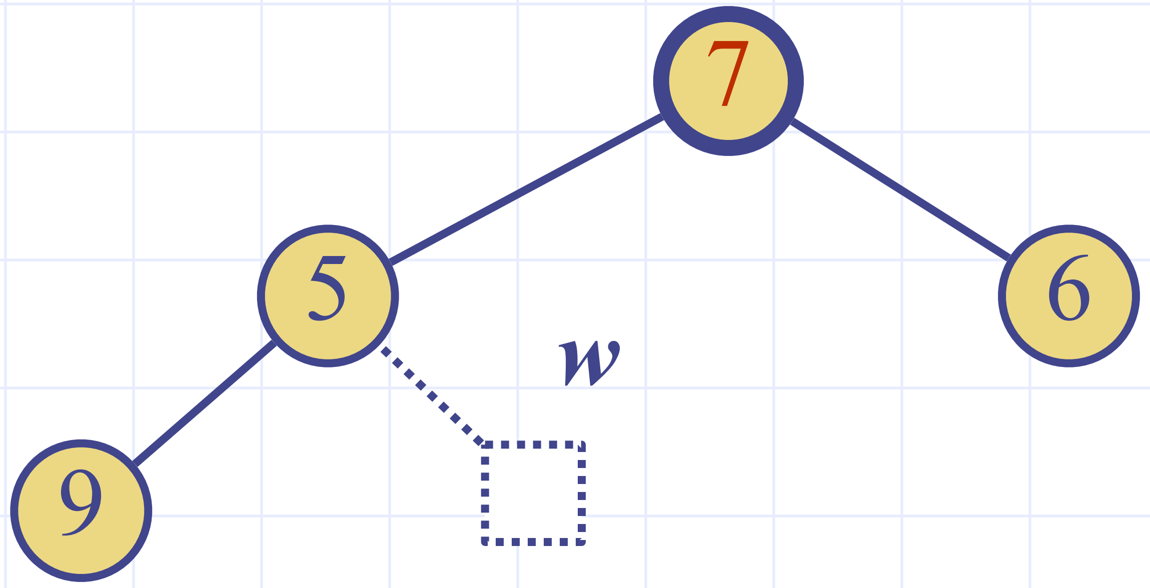
- Only allowable removal: the root.
- Note that the root is the minimum value!
- The removal algorithm consists of three steps
  - Replace the root key with the key of the last node w
  - Remove w
  - Restore the heap-order property (discussed next)

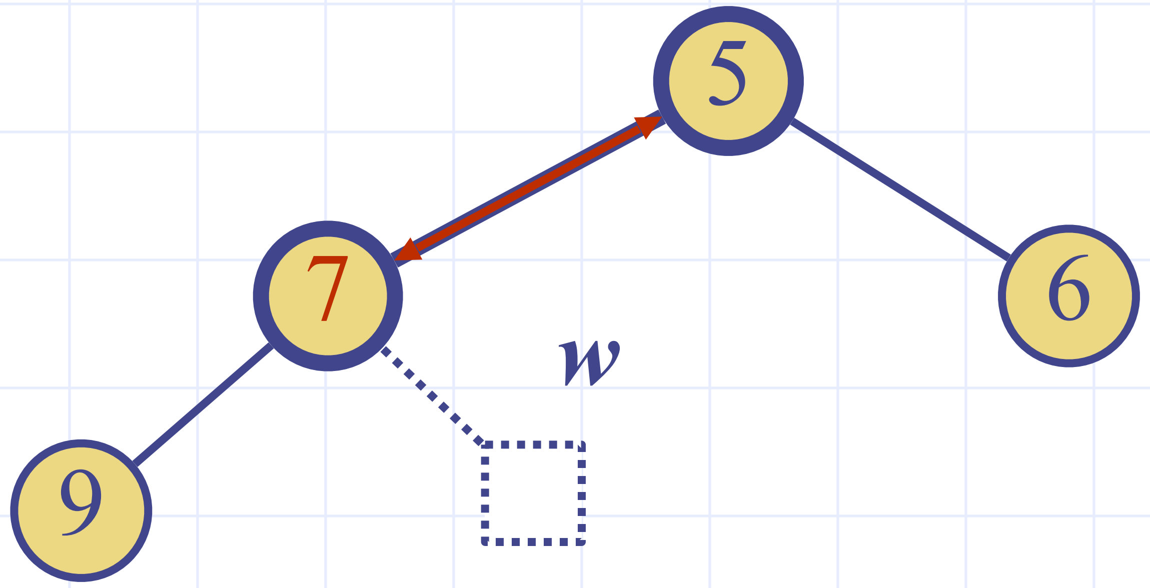




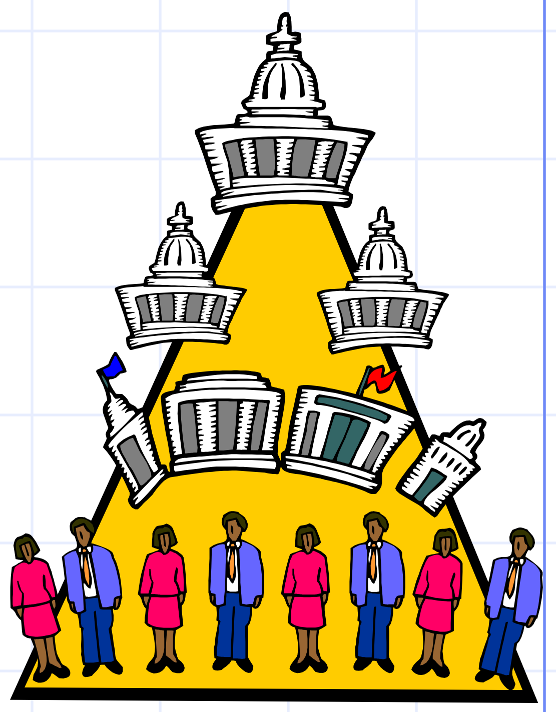
Downheap

- After replacing the root key with the key k of the last node, the heap-order property may be violated
- Algorithm downheap restores the heap-order property by swapping key k along a downward path from the root
- Downheap terminates when key k reaches a leaf or a node whose children have keys greater than or equal to k
- Since a heap has height $O(\log n)$, downheap runs in $O(\log n)$ time





Heap-Sort



- Consider a heap with n items
  - the space used is $O(n)$
  - methods insert and removeMin take $O(\log n)$ time
  - methods size, empty, and min take time $O(1)$ time
- Using a heap, we can sort a sequence of n elements in $O(n \log n)$ time
- The resulting algorithm is called heap-sort
- Heap-sort is much faster than selection-sort.
- Heap sort is guaranteed to be $O(n \log n)$
  - beats quick-sort
- Can be done “in place”
  - less memory than merge-sort

Vector-based Heap Implementation

- We can represent a heap with n keys by means of a vector of length $n + 1$
- For the node at index i
  - the left child is at index $2i$
  - the right child is at index $2i + 1$
- Links between nodes are not explicitly stored
- The cell of at index 0 is not used
- Operation insert corresponds to inserting at index $n + 1$
- Operation removeMin corresponds to returning from index 1 and moving index n to index 1
- Yields in-place heap-sort