Heaps

CSE 2011
Fall 2009

Priority Queues

- Data structure supporting the following operations:
  - *insert* (equivalent to enqueue)
  - *deleteMin* (or *deleteMax*) (equivalent to dequeue)
  - Other operations (optional)
- Applications:
  - Emergency room waiting list
  - Routing priority at routers in a network
  - Printing job scheduling

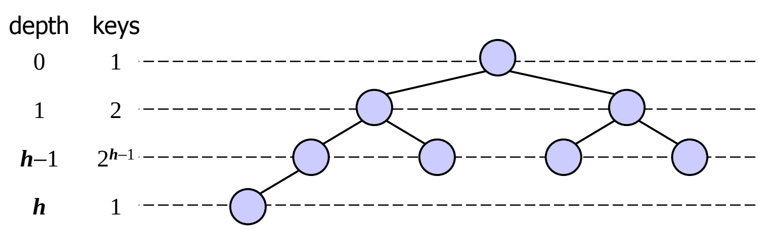
Simple Implementations of PQs

- Unsorted linked list
  - insertion O( )
  - deleteMin O( )
- Sorted linked list
  - insertion O( )
  - deleteMin O( )
- AVL trees
  - insertion O( )
  - deleteMin O( )
- Unsorted array
  - insertion O( )
  - deleteMin O( )
- Sorted array
  - insertion O( )
  - deleteMin O( )
- A data structure more efficient for PQs is heaps.

3

Complete Binary Trees

- Let $\boldsymbol{h}$ be the height of a binary tree.
  - for $\boldsymbol{i} = 0, \ldots, \boldsymbol{h} - 1$, there are 2^i nodes at depth $\boldsymbol{i}$.
    - that is, all levels except the last are full.
  - at depth $\boldsymbol{h}$, the nodes are filled from left to right.

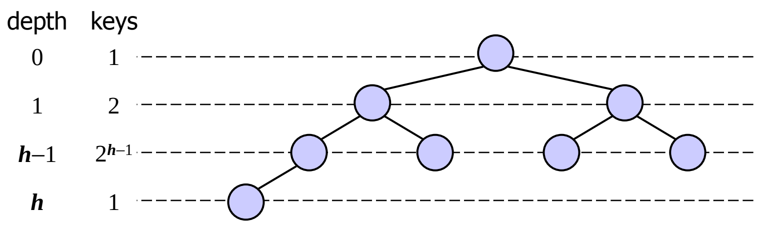


4

Complete Binary Trees (2)

- Given a complete binary tree of height *h* and size *n*,
  $2^h \leq n \leq 2^{h+1} - 1$

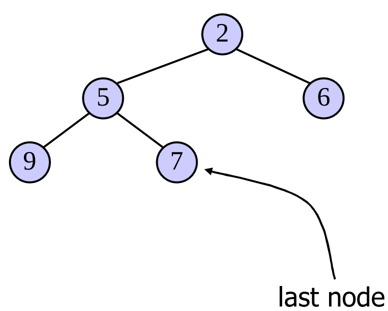
- Which data structure is better for implementing complete binary trees, arrays or linked structures?



5

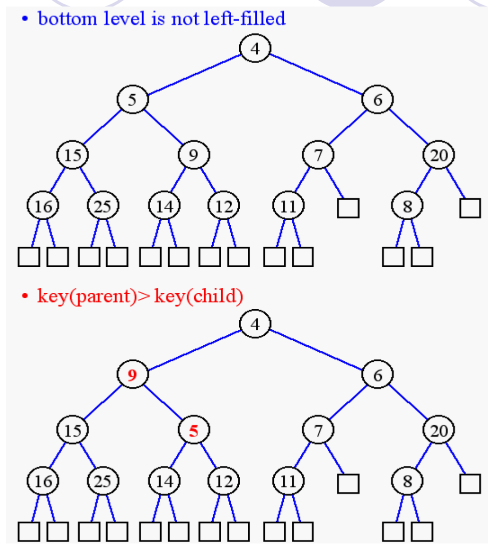
Heaps

- 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 at depth i.
    - at depth h, the nodes are filled from left to right.
- The last node of a heap is the rightmost node of depth h.
- Where can we find the smallest key in a min heap? The largest key?



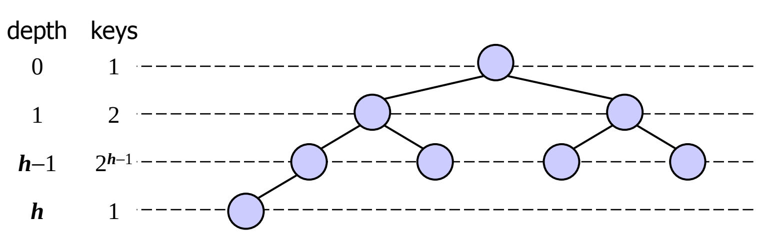
6

Examples that are not heaps



Height of a Heap

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



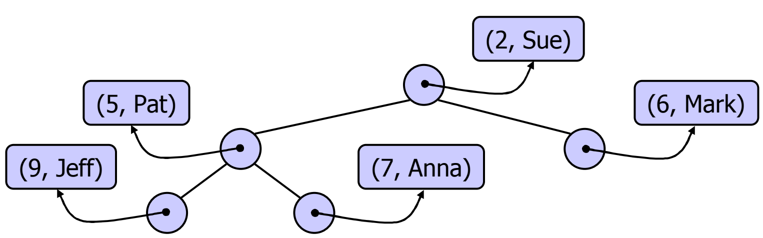
Max Heap

- The definition we just discussed is for a *min* heap.
- Analogously, we can declare a *max* heap if we need to implement *deleteMax* operation instead of *deleteMin*.

9

Heaps and Priority Queues

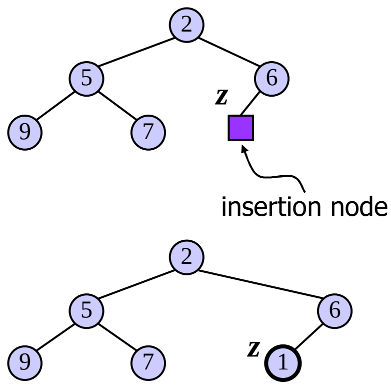
- We can use a heap to implement a priority queue
- We store a (key, element) item at each internal node
- We keep track of the position of the last node



10

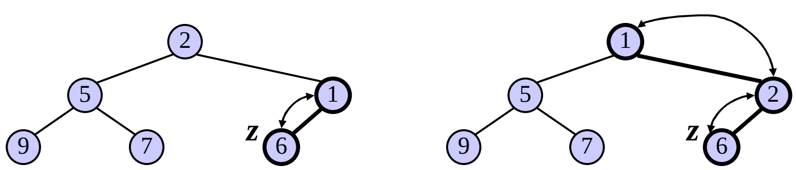
Insertion into a Heap

- Method insert of the priority queue ADT corresponds to the 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 Percolation (Bubbling)

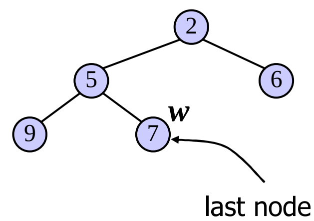
- After the insertion of a new key k, the heap-order property may be violated
- Algorithm upheap restores the heap-order property by swapping k along an upward path from the insertion node
- Upheap terminates when the key k reaches the root or a node whose parent has a key smaller than or equal to k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

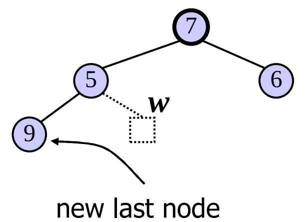


www.cs.hut.fi/Opinnot/T-106.1220/heaptutorial/lisaaminen.html

Removal from a Heap

- Method *deleteMin* of the priority queue ADT corresponds to the removal of the root key from the heap
- 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)

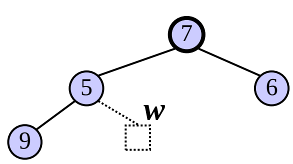


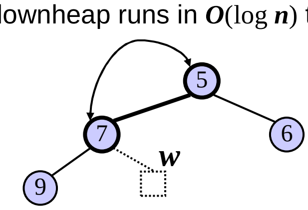


13

Downheap Percolation

- 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
- Upheap 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





14

Next Time …

- Heap Sort (8.3.5)

15