Chap6. Heap Sort

- Heaps
- Maintaining the heap property
- Building a heap
- The heapsort algorithm
- Priority queues

Heapsort

- Combines the better attributes of merge sort and insertion sort.
  - Like merge sort, but unlike insertion sort, running time is $O(n \log n)$.
  - Like insertion sort, but unlike merge sort, sorts <u>in place</u>.
- Introduces an algorithm design technique
  - Create data structure (*heap*) to manage information during t he execution of an algorithm.
- The *heap* has other applications beside sorting.
  - Priority Queues

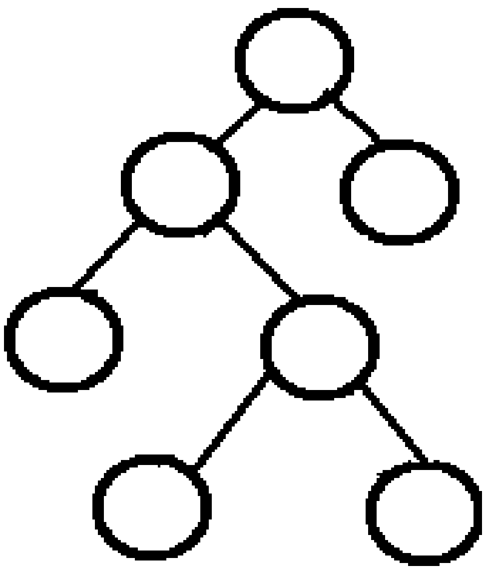
Some Definitions

- <u>In Place</u> Sorting
  - The amount of extra space required to sort the data is constant with the input size.
  - Some devices don't have enough space
    ex) Embedded system like PDA, cellphone
  - Reducing space usage is important
- <u>Not in place (out of place)</u> sorting
  - The opposite of in place sorting

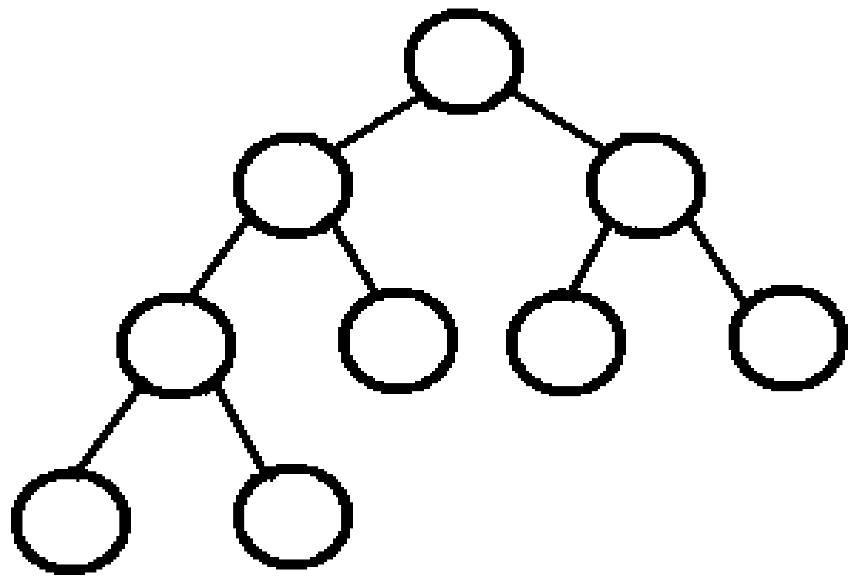
Full vs. Complete Binary Trees

- A binary tree T is *full* if each node is either a leaf or has exactly two child nodes.
- A binary tree T with n levels is *complete* if all levels except possibly the last are completely full, and the last level has all its nodes to the left side.

Full vs. Complete Binary Trees

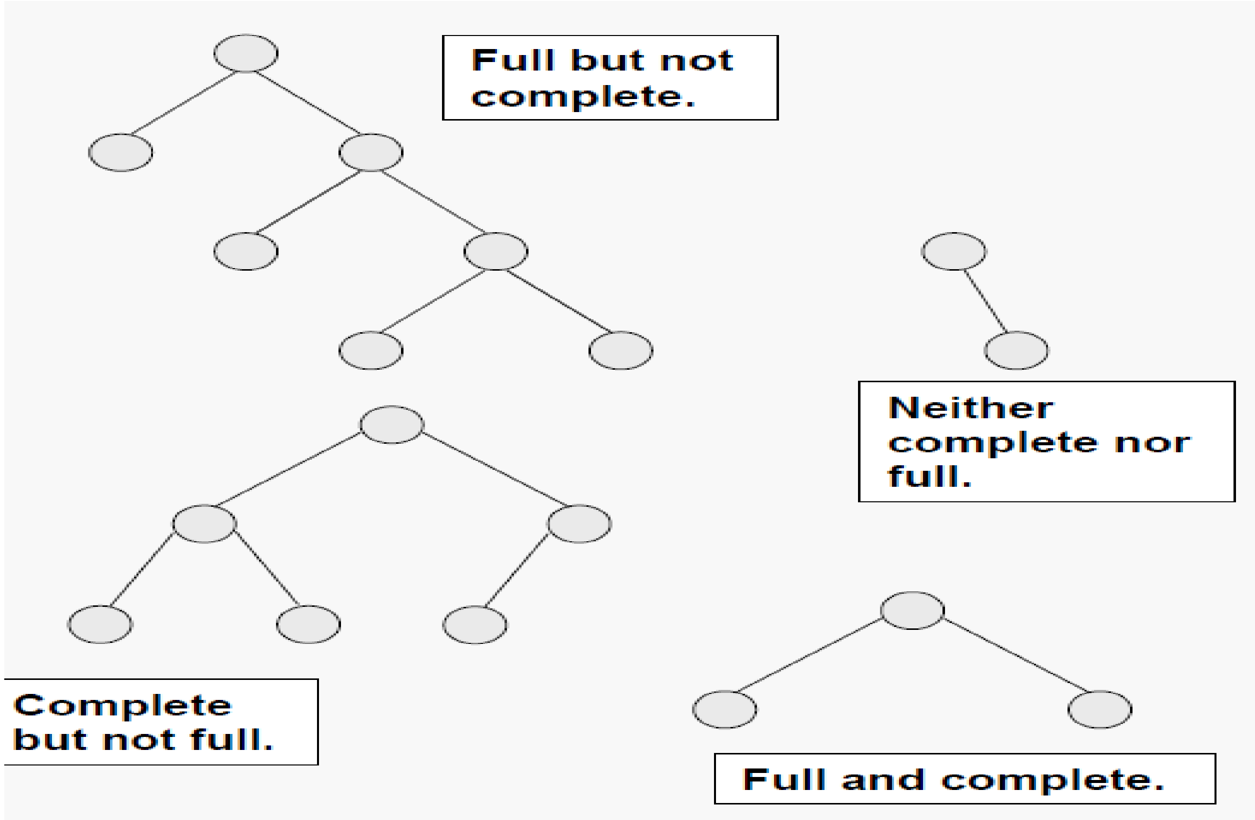


full tree



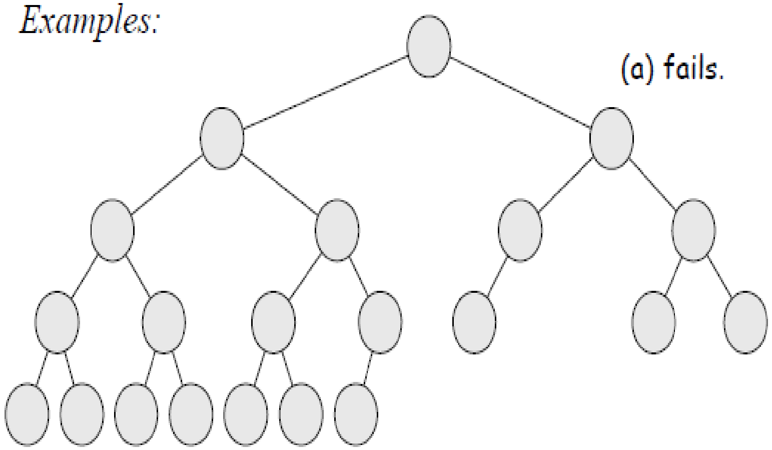
complete tree

Full vs. Complete Binary Trees

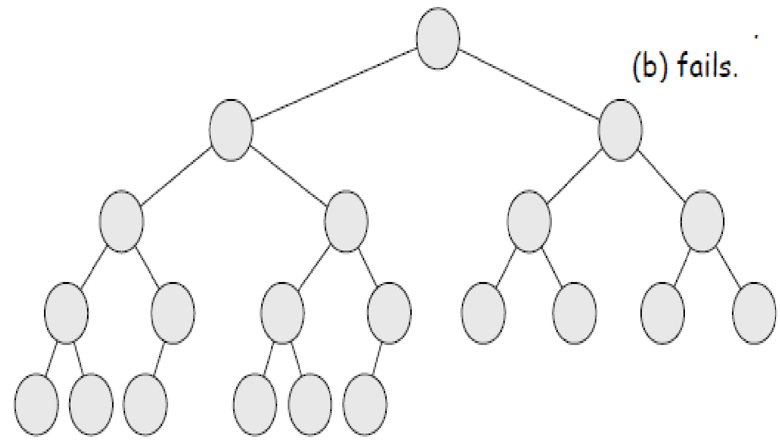


Complete Binary Tree

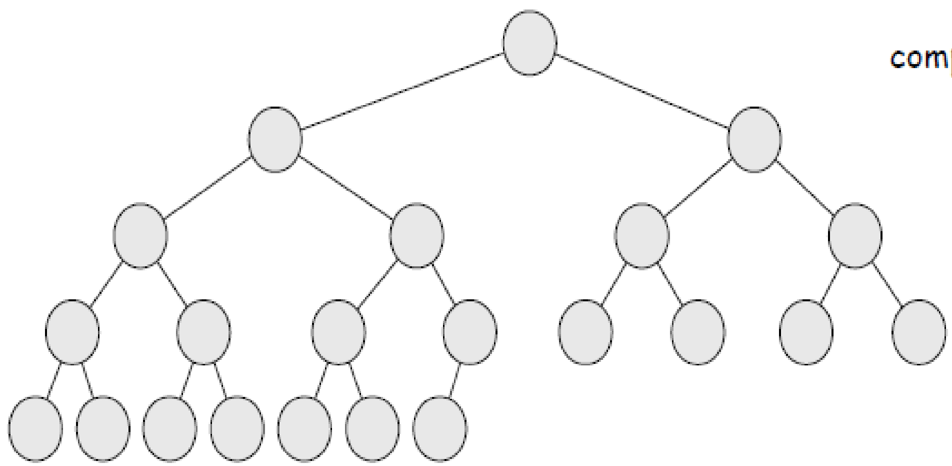
Examples:



(a) fails.



(b) fails.



complete.

Representation of Complete Binary Tree

- A complete binary tree may be represented as an array (i.e., no pointers):
- Number the nodes, beginning with the root node and moving from level to level, left to right within a level.
- The number assigned to a node is its index in the array.

Additional Properties of Complete Binary Trees

- The root of the tree is A[1].
- If a node has index i, we can easily compute the indices of its:
  - parent $\lfloor i/2 \rfloor$
  - left child 2i
  - right child 2i + 1
- Array viewed as a complete binary tree.
  - Physically linear array.
  - Logically binary tree, filled on all levels (except lowest)

Heap

- A heap is a *complete binary tree* that satisfies the heap property:

  max-heap: For every node i other than the root:

  $$A[\text{Parent}(i)] \geq A[i]$$

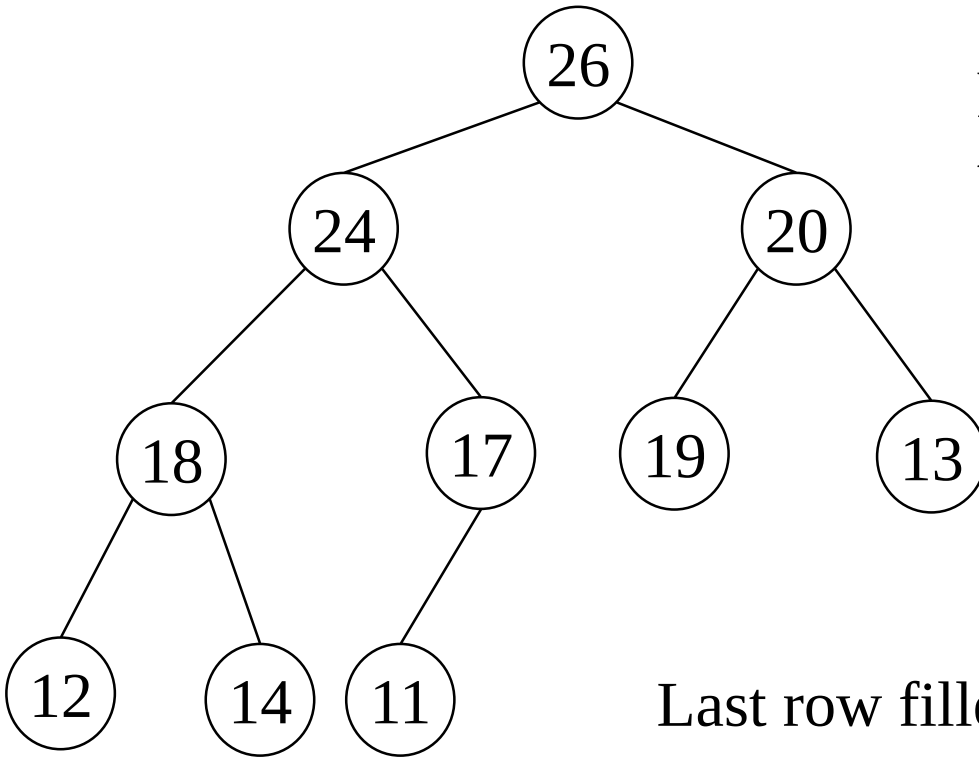
  min-heap: For every node i other than the root:

  $$A[\text{Parent}(i)] \leq A[i]$$

Heap vs. Array

1	2	3	4	5	6	7	8	9	10
26	24	20	18	17	19	13	12	14	11

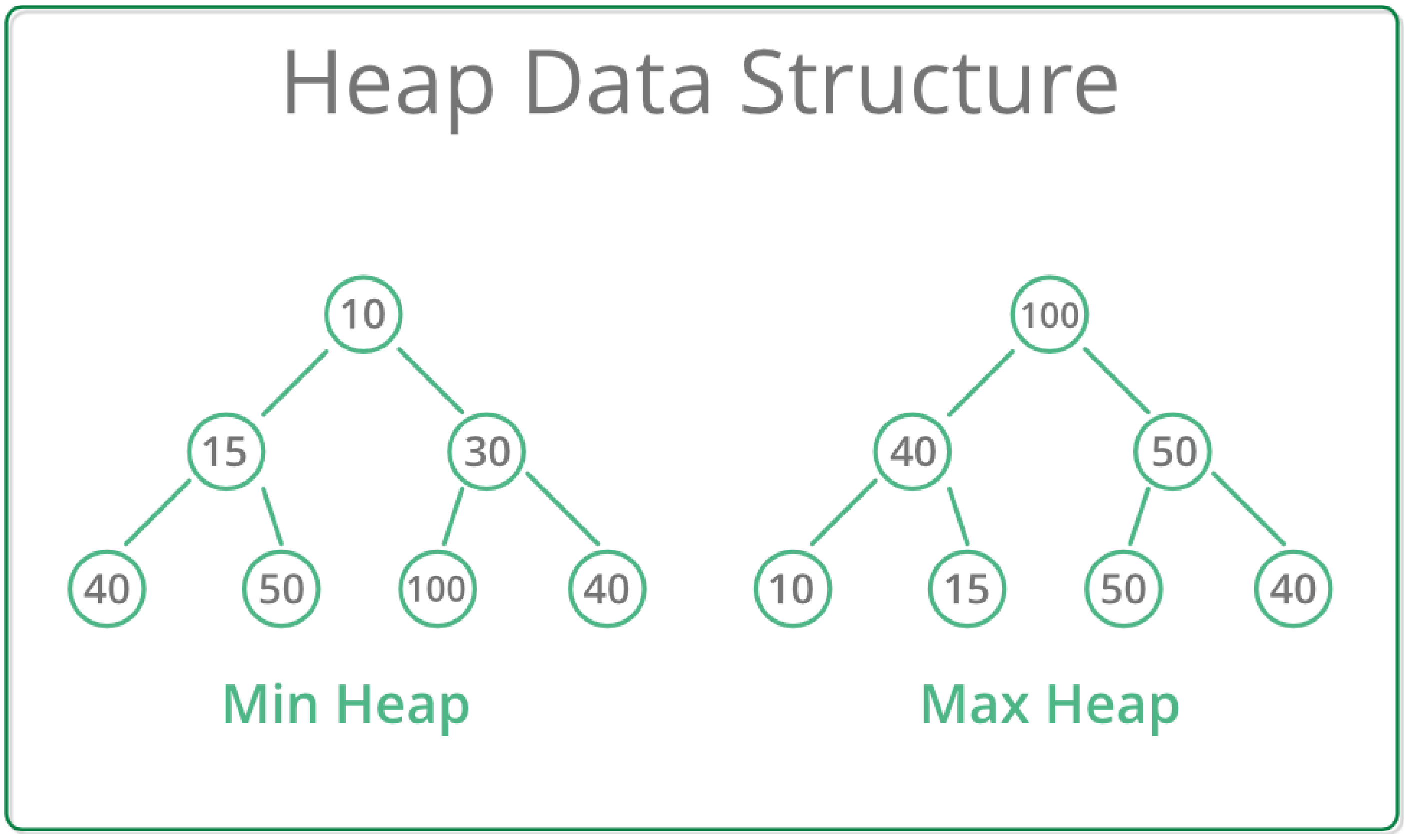
heap as an array.



heap as a binary tree.

Last row filled from left to right.

Min Heap vs. Max Heap



Max-Heap

- A max-heap is a *complete binary tree that* satisfies the heap property:
  For every node i other than the root,
  $A[PARENT(i)] \geq A[i]$
- What does this mean?
  - the value of a node is at most the value of its parent
  - <u>the largest element in the heap is stored in the root</u>

Height

- *Height of a node in a tree*: the number of edges on the longest simple downward path from the node to a leaf.
- *Height of a tree*: the height of the root.
- Height of a heap: $\lfloor \log n \rfloor$
  - <u>Basic operations on a heap run in $O(\log n)$ time</u>

Heap Characteristics

- *Height* $= \lfloor \log n \rfloor$
- # of *leaves* $= \lceil n/2 \rceil$
- # of nodes of height h $\leq \lceil n/2^{h+1} \rceil$