ENSF 593/594
Data Structures – Heaps and Heapsort

Mohammad Moshirpour

Outline

- Heap
  - Max heap
  - Min heap
- Enqueue
- Dequeue
- Re-organizing an array into a heap
- Heapsort

Goal

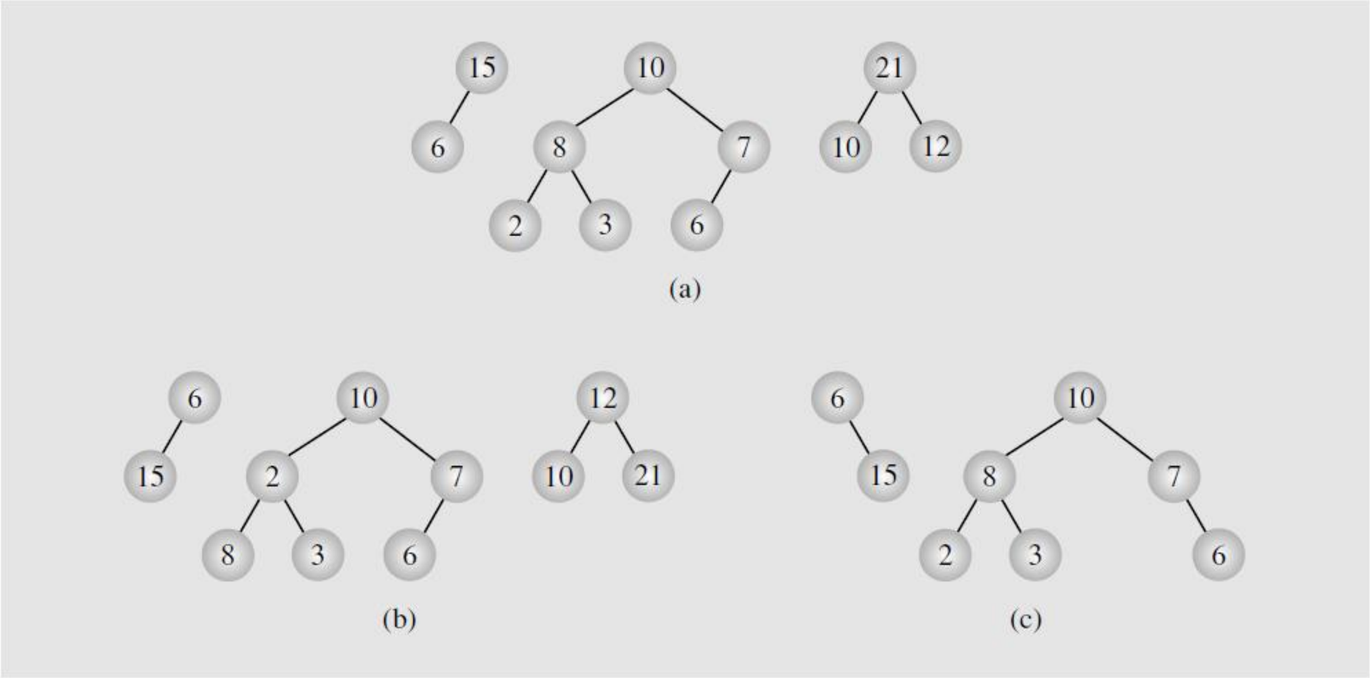
- In this lecture we will learn about heap trees which are another organization of node in a tree. We will learn about how to make them and how to use them to sort data in array.

Heaps

- A *max heap* is a binary tree where:
  - The value of each node is ≥ the values of its children
  - The tree is *complete*
    - The tree is perfectly balanced
    - The leaf nodes in the last level are all pushed to the left
    - Height is $O(\lg n)$

Heaps

- E.g.



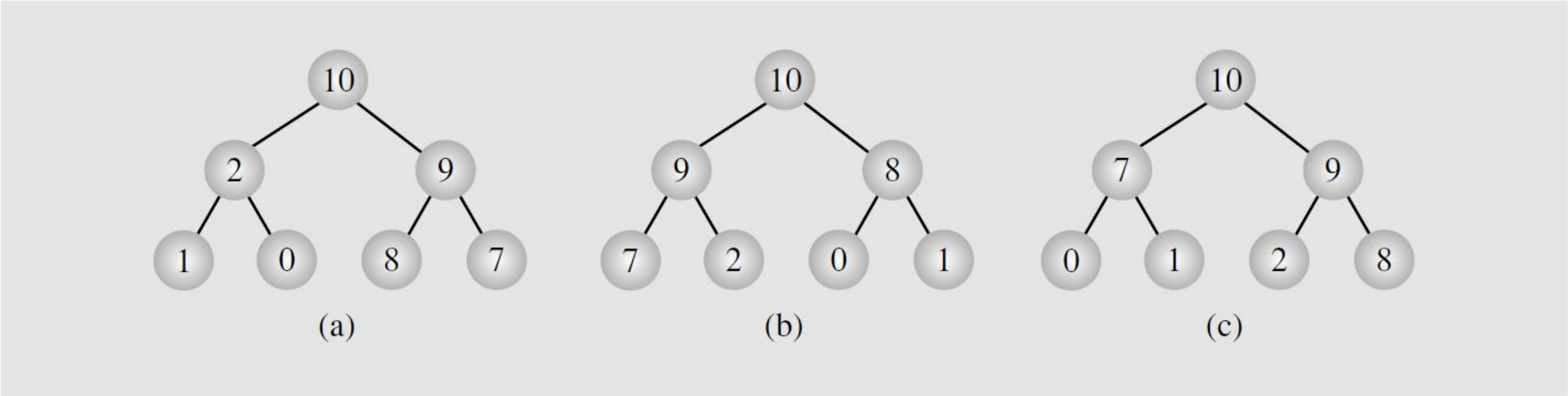
Examples of (a) heaps and (b–c) nonheaps

Heaps (cont'd)

- The largest element is always the root node
- A *min heap* is similar except the value of each node is ≤ the values of its children
    - The root contains the smallest element
- Heaps are not perfectly ordered
    - The above properties ensure only that order is maintained through linear lines of descent
    - Lateral lines may be out of order

Heaps (cont’d)

- E.g.



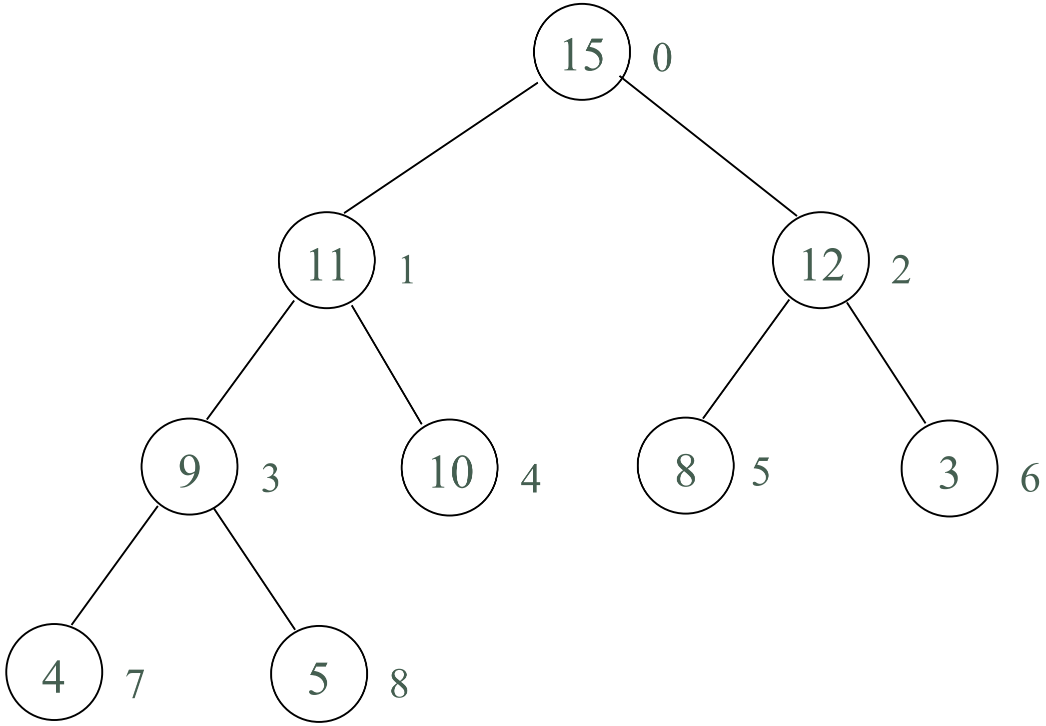
Different heaps constructed with the same elements

Heaps (cont’d)

- Heaps are normally implemented using arrays (or vectors)
  - Elements are stored sequentially in the array:
    - Level by level from top to bottom, and
    - From left to right at each level

Heaps (cont'd)

- E.g.



15	11	12	9	10	8	3	4	5
0	1	2	3	4	5	6	7	8

Heaps (cont'd)

- The root node is always at position 0
- The position of the left child of a node at i is:

  $2i + 1$, where the position is $< n$
- The position of the right child is:

  $2i + 2$, where position is $< n$
- The position of the parent of a node at i is:

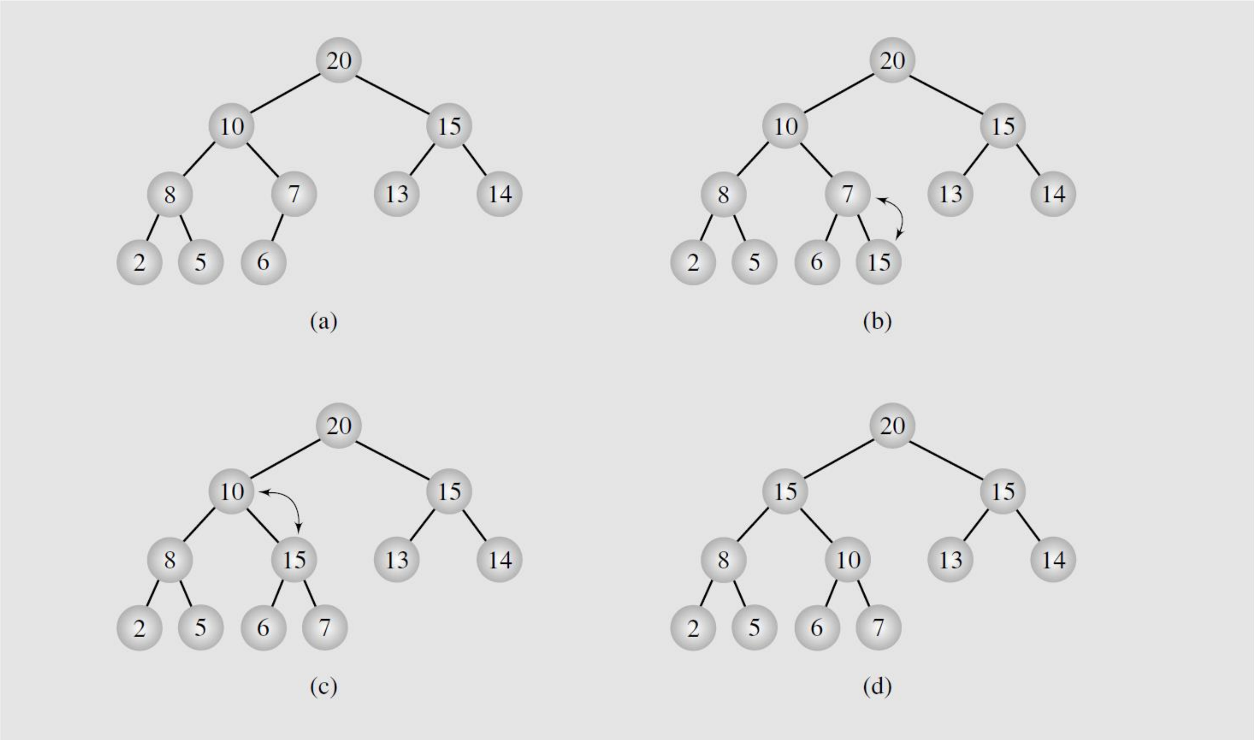
  $(i - 1) / 2$, where $1 < i < n$
  - Note: assumes integer division

Heaps (cont'd)

- Heaps are often used to implement priority queues
  - More efficient than linear structures
    - $O(\lg n)$ vs. $O(n)$
  - Enqueue procedure:
    - Add the new element to the end of the heap
      - i.e. At the end of the array
    - If necessary, restore the heap property by swapping the element with its parent
      - Repeat until proper position found, or is at the root

Heaps (cont'd)

- E.g.



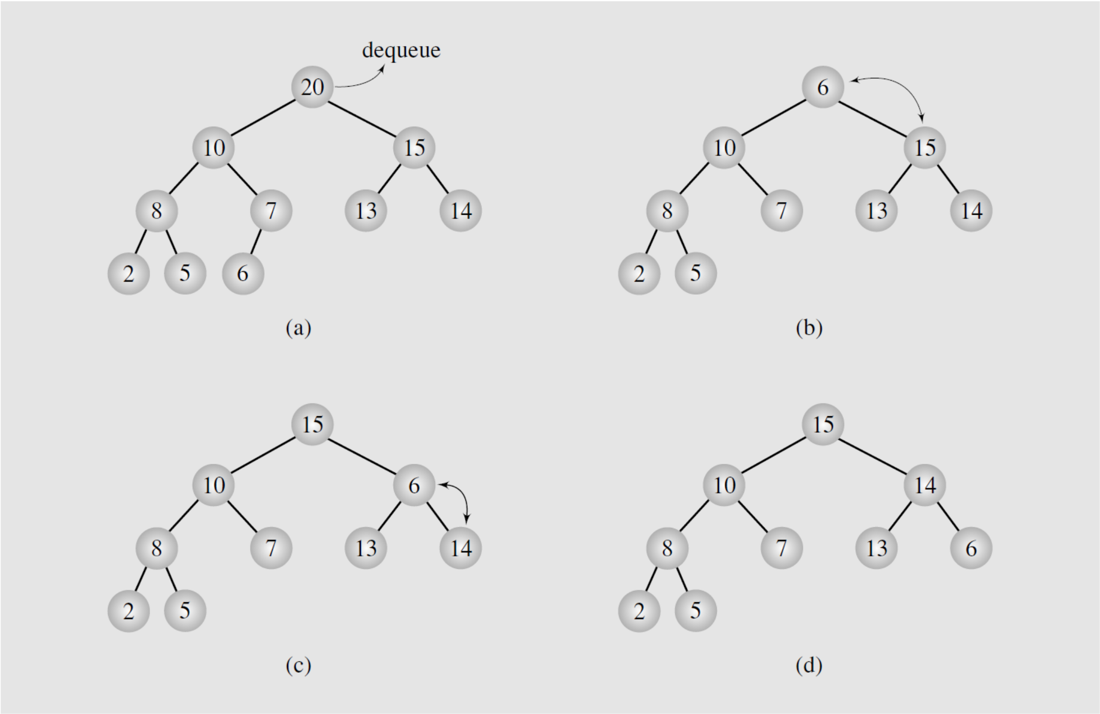
Enqueuing an element to a heap

Heaps (cont’d)

- Dequeue procedure:
  - Remove the root element
    - Always has the highest priority
  - Replace it with the last leaf node
  - If necessary, restore the heap property by swapping the root with its larger child
    - Repeat until proper position found, or it becomes a leaf node

Heaps (cont'd)

- E.g.



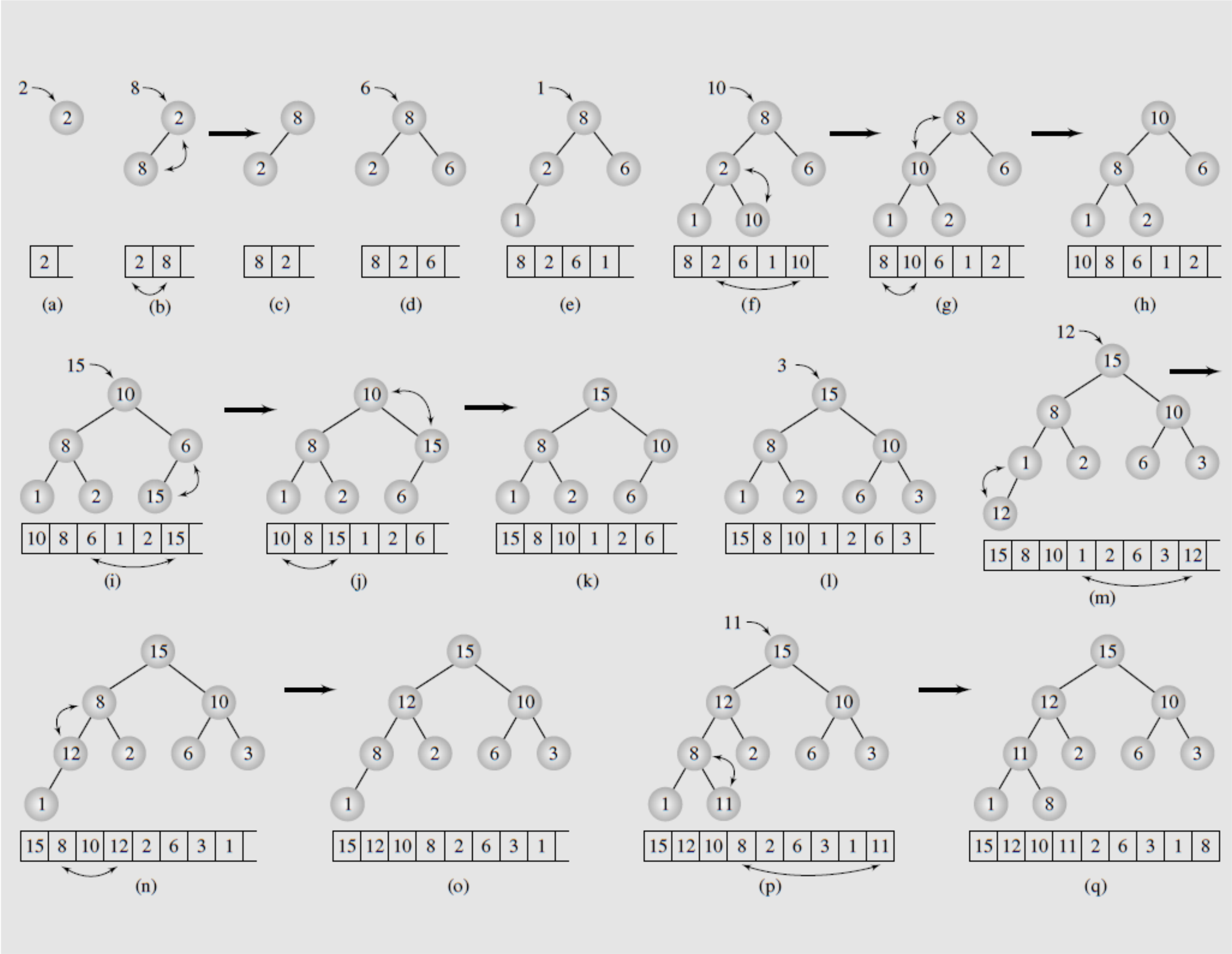
Dequeuing an element from a heap

Heaps (cont'd)

- Sometimes we need to reorganize the contents of an array into a heap
  - E.g. For the heapsort
  - Top-down method:
    - Start with empty heap
    - Sequentially enqueue new elements
    - Is $O(n \lg n)$ in the worst case

Heaps (cont’d)

- E.g.



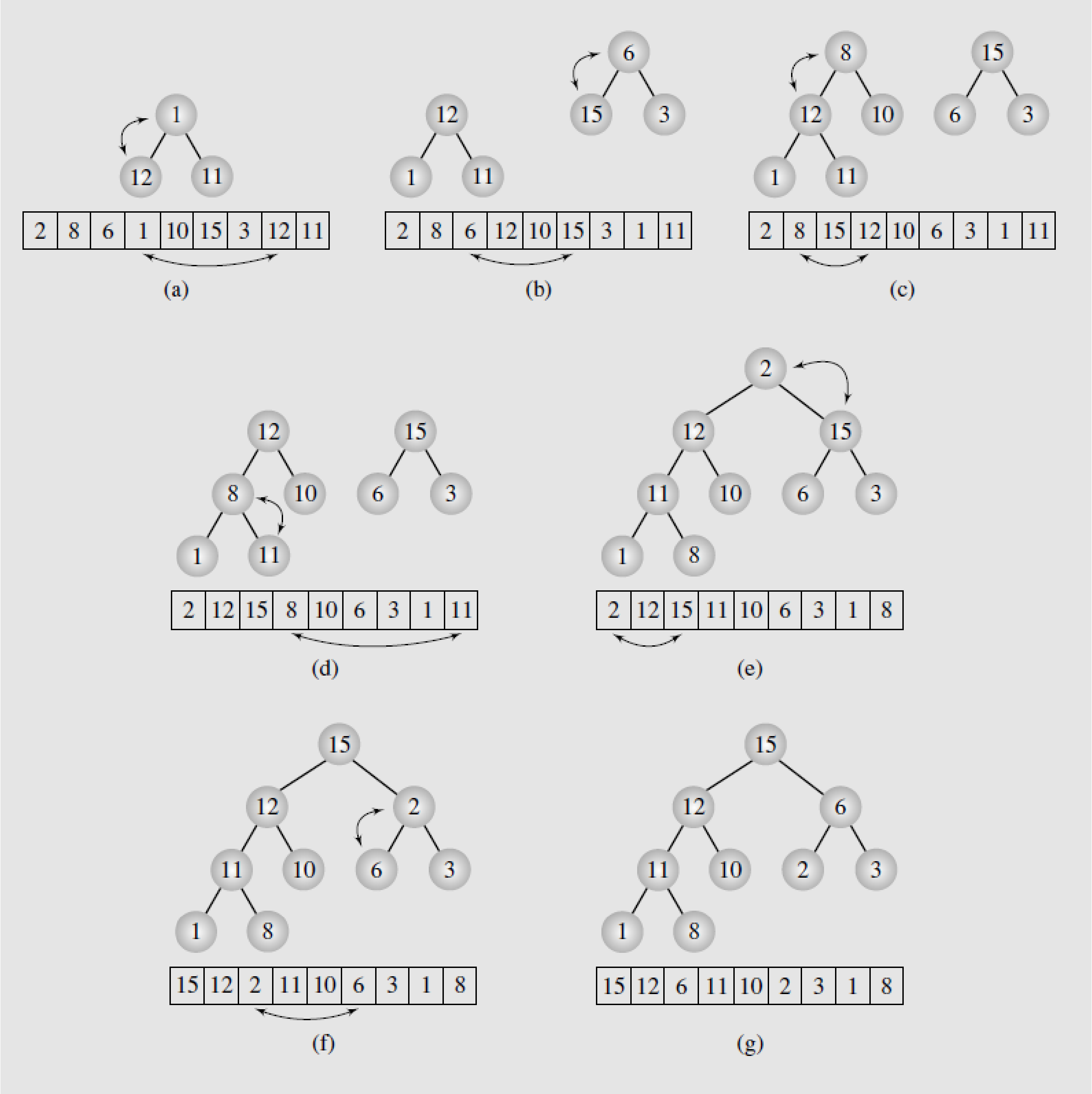
Organizing an array as a heap
With a top-down method

Heaps (cont'd)

- Bottom-up method:
  - Is $O(n)$ in the worst case
  1) Start with the last non-leaf node
     - Is at position $n/2 - 1$
     - Set index to this position
  2) If necessary, restore the heap property by swapping with largest child
     - Repeat until proper place found, or becomes a leaf node
  3) Repeat step 2 after decrementing index
     - Stop once the root has been processed

Heaps (cont'd)

- E.g.



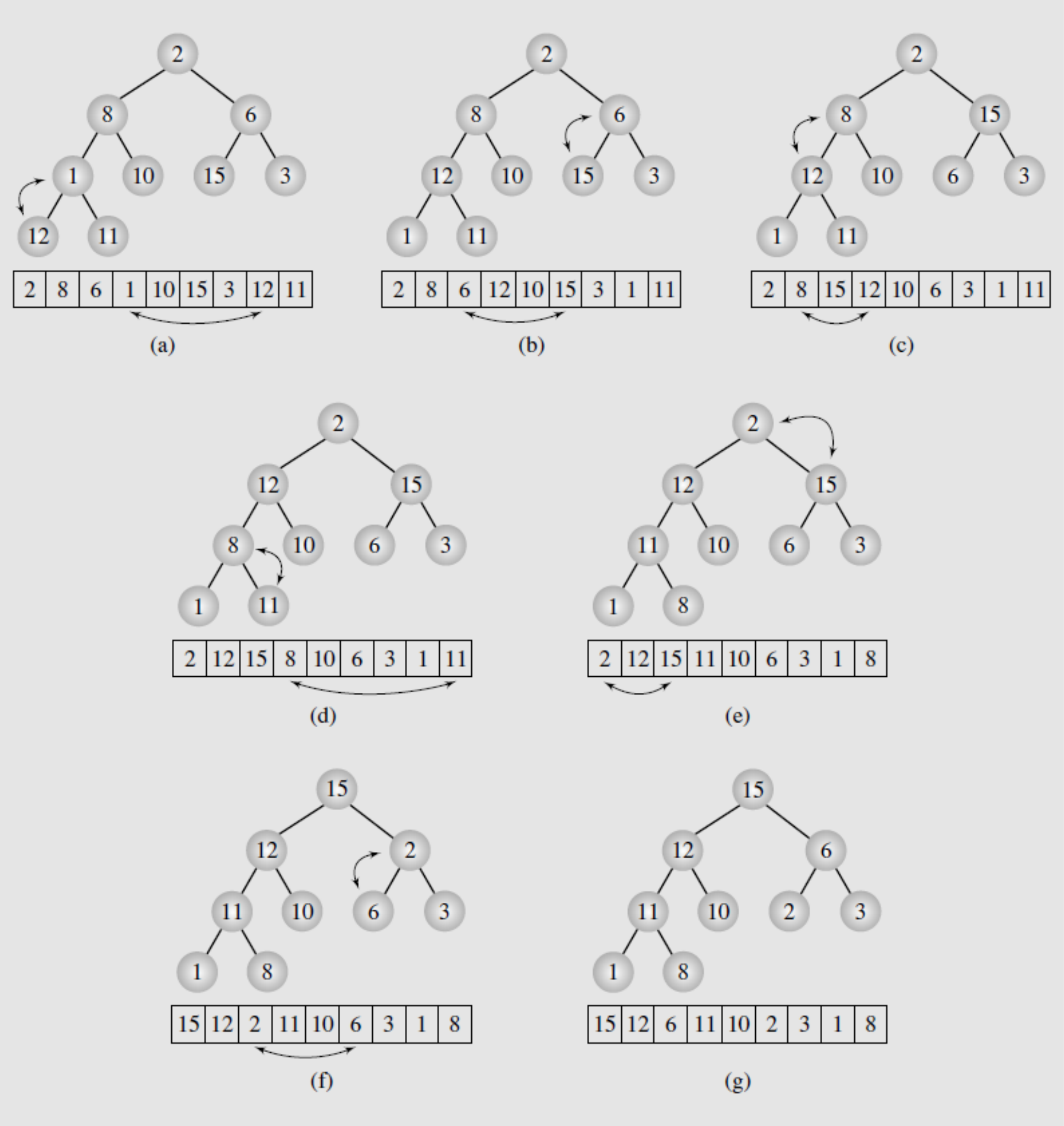
Transforming the array [2 8 6 1 10 15 3 12 11] into a heap with a bottom-up method

Heapsort

- Is an in-place sort of an array
- Procedure:
  - Reorganize the array into a heap
    - Bottom-up method is quickest
  - for (i = n–1; i > 0; i--)
    - Swap root with element i
      - Puts the largest element at the end of the array, so is no longer considered
    - Restore heap property for elements 0 to i – 1

Heapsort (cont'd)

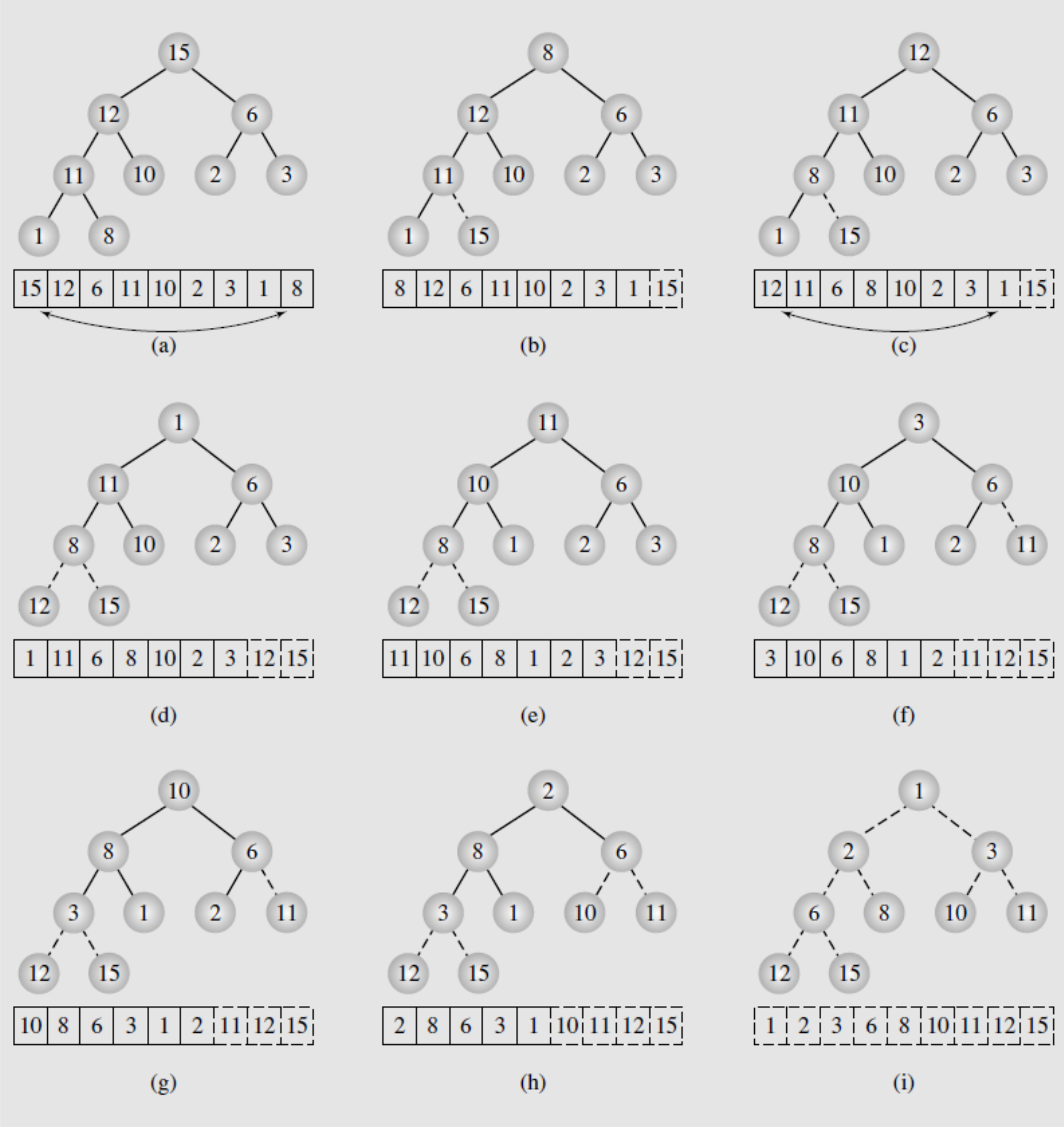
- E.g.



Transforming the array [2 8 6 1 10 15 3 12 11] into a heap

Heapsort (cont'd)

- E.g.



Execution of heap sort on the array [15 12 6 11 10 2 3 1 8], which is the heap constructed in the previous slide

Heapsort (cont'd)

- Is $O(n \lg n)$ in worst and average cases
- Is $O(n)$ in the best case for an array containing identical elements

Summary

- Max (min) heap:
  - The value of each node is ≥ (≤) the values of its children
  - The tree is complete
- Heap used to implement priority queue
  - Enqueue procedure starts by adding the new element to the end of the heap
  - Dequeue procedure starts by removing elements from root
- Heap sort on an arbitrary array:
  - Reorder arrays element to make a heap
  - Applying heap sort algorithm

Review Questions

- What is the difference between max heap and min heap?
- How can you store a heap tree in a array?
- Explain about enqueue and dequeue procedure in a heap.
- Explain how to perform a heap sort on a arbitrary array?

Any questions?