Binary Trees (cont.)

Instructor: Dr. Sunho Lim (Ph.D., Assistant Professor)

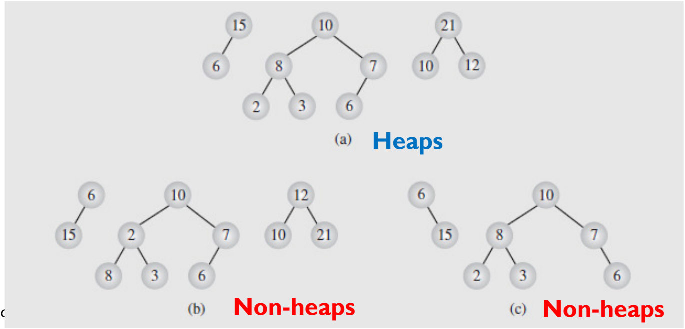
Lecture 11

sunho.lim@ttu.edu

Adapted partially from Data Structures and Algorithms in C++, Adam Drozdek, 4th Edition, Cengage Learning; and Algorithms and Data Structures, Douglas Wilhelm Harder, Mmath

Heaps

- A ***heap,*** a special type of binary tree
  - the value of each node is **greater than or equal** to the values stored in its **children**
  - the tree is perfectly balanced, and the leaves in the last level are **leftmost** in the tree



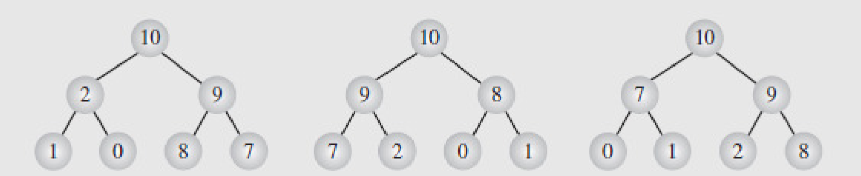
(a) Heaps

(b) Non-heaps

(c) Non-heaps

Heaps (cont.)

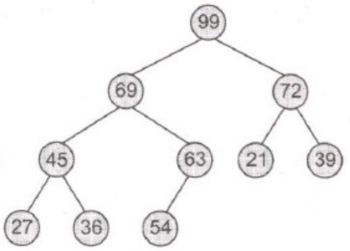
- Different heaps constructed with the same elements



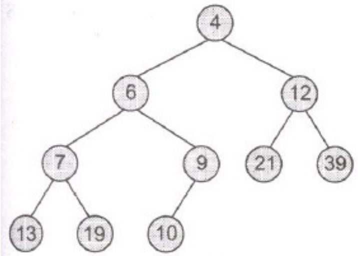
(a) (b) (c)

Heaps (cont.)

- A *max heap*;
  - the value of each node is **greater than or equal** to the values stored in its **children**
  - the root of a max heap, the **largest** element
- A *min heap*
  - the value of each node is **less than or equal** to the values stored in its **children**
  - the root of a min heap, the **smallest** element



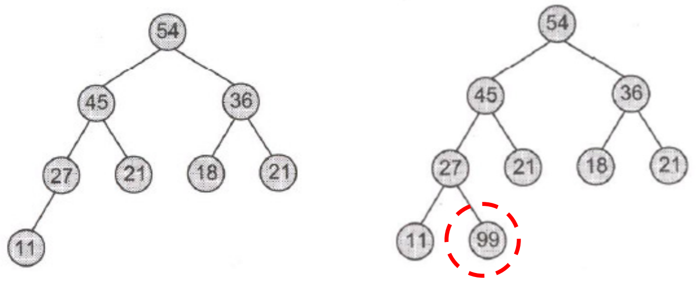
Max heap



Min heap

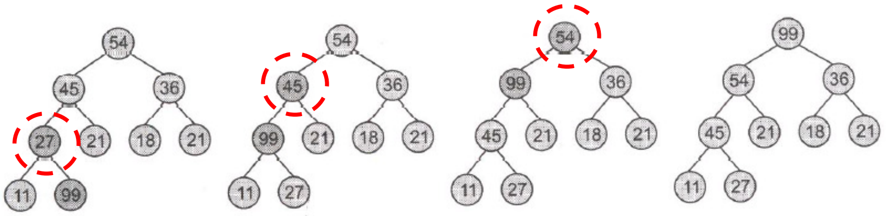
Heaps (cont.)

- Insert a new element, 99, in **max heap**
  - leftmost in the heap



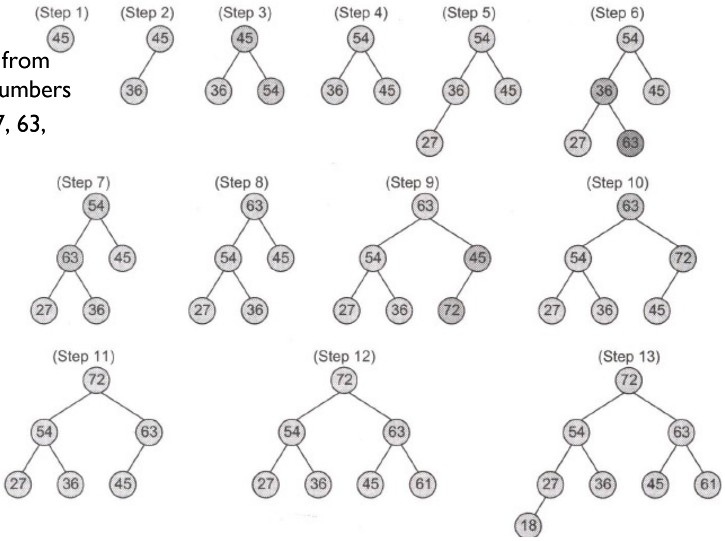
Heaps (cont.)

- Insert a new element, 99, in **max heap** (cont.)
  - heapify



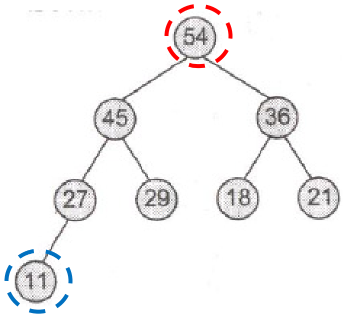
Heaps (cont.)

- Build a max heap from the given set of numbers
  - 45, 36, 54, 27, 63, 72, 61, 18



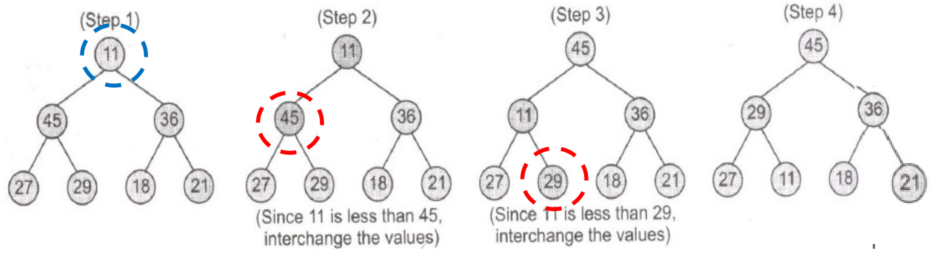
Heaps (cont.)

- Delete a node in **max heap**
  - always deleted from the root of the heap
  - **replace** the root node with the last node



Heaps (cont.)

- Delete a node in **max heap** (cont.)
  - always deleted from the root of the heap
  - **replace** the root node with the last node



(Since 11 is less than 45, interchange the values)

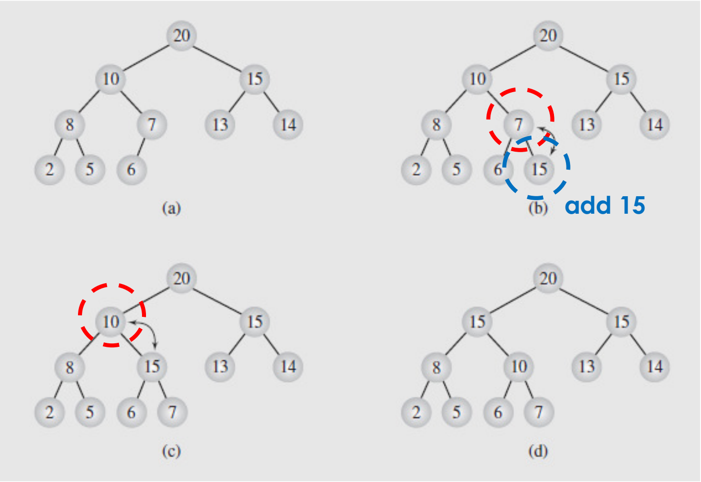
(Since 11 is less than 29, interchange the values)

Heaps (cont.)

- Heaps as **Priority Queues**
  - perfectly balanced trees, the inherent efficiency of searching such structures makes them more useful
  - **enqueue** and **dequeue** operations
  - enqueue,
    - add at the end of the heap as the last leaf
    - may need restructure the heap

```
heapEnqueue(el)
    put el at the end of the heap;
      while el is not in the root and el > parent(el)
         swap el with its parent;
```

Heaps (cont.)

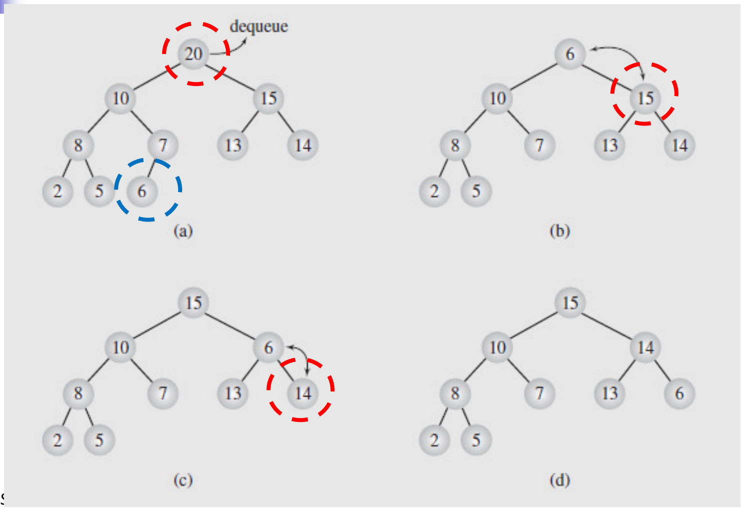


Heaps (cont.)

- Heaps as **Priority Queues** (cont.)
  - dequeue
    - remove the root (since it is the largest value) and replace it by the last leaf
    - need restructure the heap

```
heapDequeue()
    extract the element from the root;
    put the element from the last leaf in its place;
    remove the last leaf;
// both subtrees of the root are heaps
    p = the root;
    while p is not a leaf and p < any of its children
        swap p with the larger child;
```

Heaps (cont.)


dequeue
20
10
15
8
7
13
14
2
5
6
(a)
6
10
15
8
7
13
14
2
5
(b)
15
10
6
8
7
13
14
2
5
(c)
15
10
14
8
7
13
6
2
5
(d)