Data Structures in Java

Session 12

Instructor: Bert Huang

http://www1.cs.columbia.edu/~bert/courses/3134

Announcements

- Homework 2 solutions posted
- Homework 3 due 10/20, if submitting late, contact me
- Midterm Exam, open book/notes 10/22
  - example problems posted on courseworks

Review

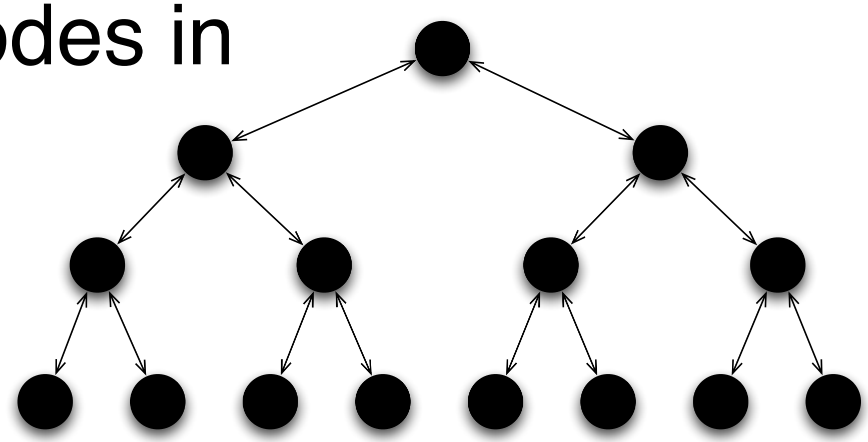
- Priority Queue data type
- Heap data structure
- Insert, percolate up
- deleteMin, percolate down

Building a Heap from an Array

- How do we construct a binary heap from an array?
- Simple solution: insert each entry one at a time
- Each insert is worst case **O(log N)**, so creating a heap in this way is **O(N log N)**
- Instead, we can jam the entries into a full binary tree and run **percolateDown** intelligently

buildHeap

- Start at deepest non-leaf node
  - in array, this is node N/2
- **percolateDown** on all nodes in reverse level-order
  - for i = N/2 to 1
    percolateDown(i)

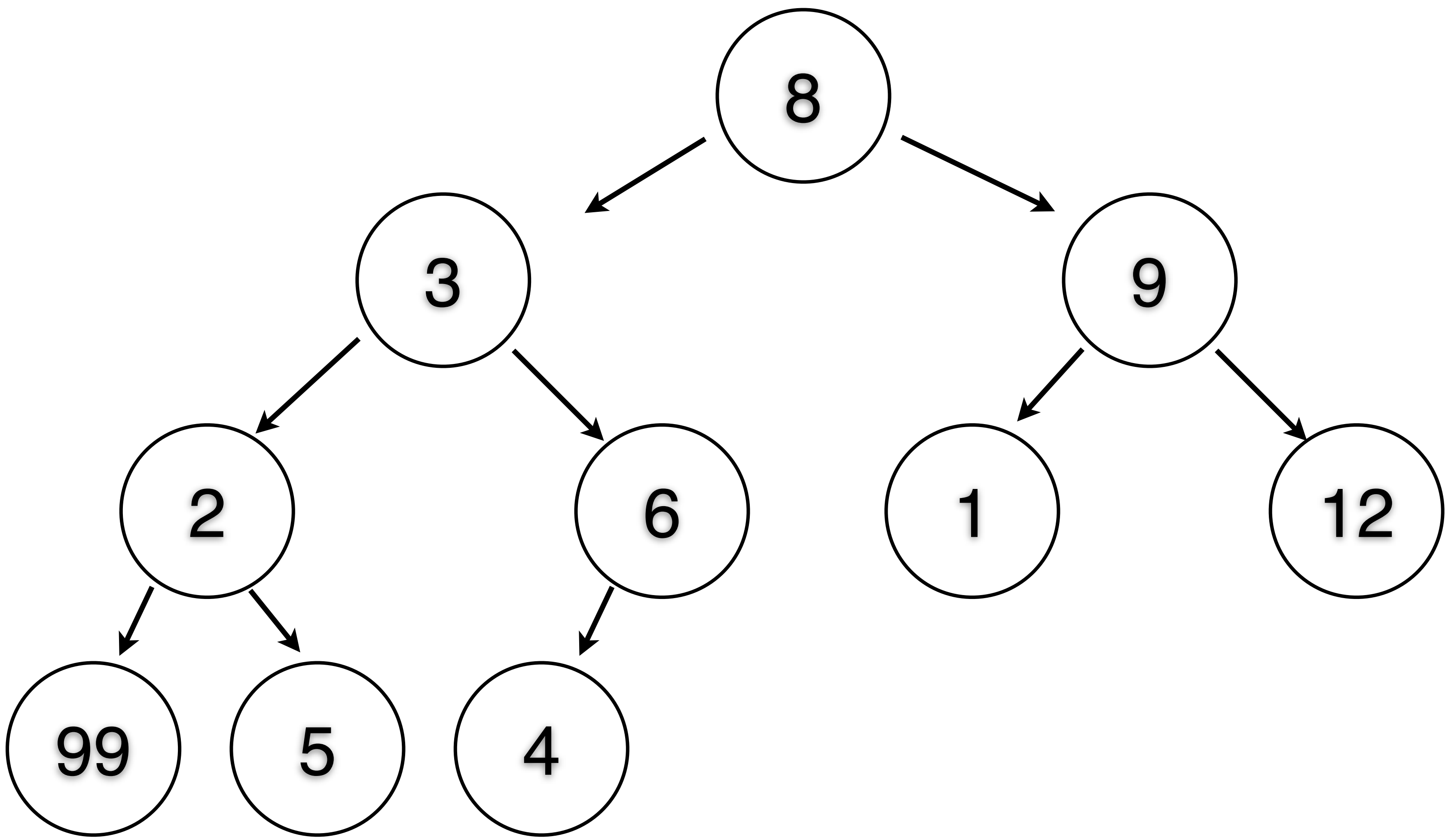


buildHeap Example

8	3	9	2	6	1	12	99	5	4

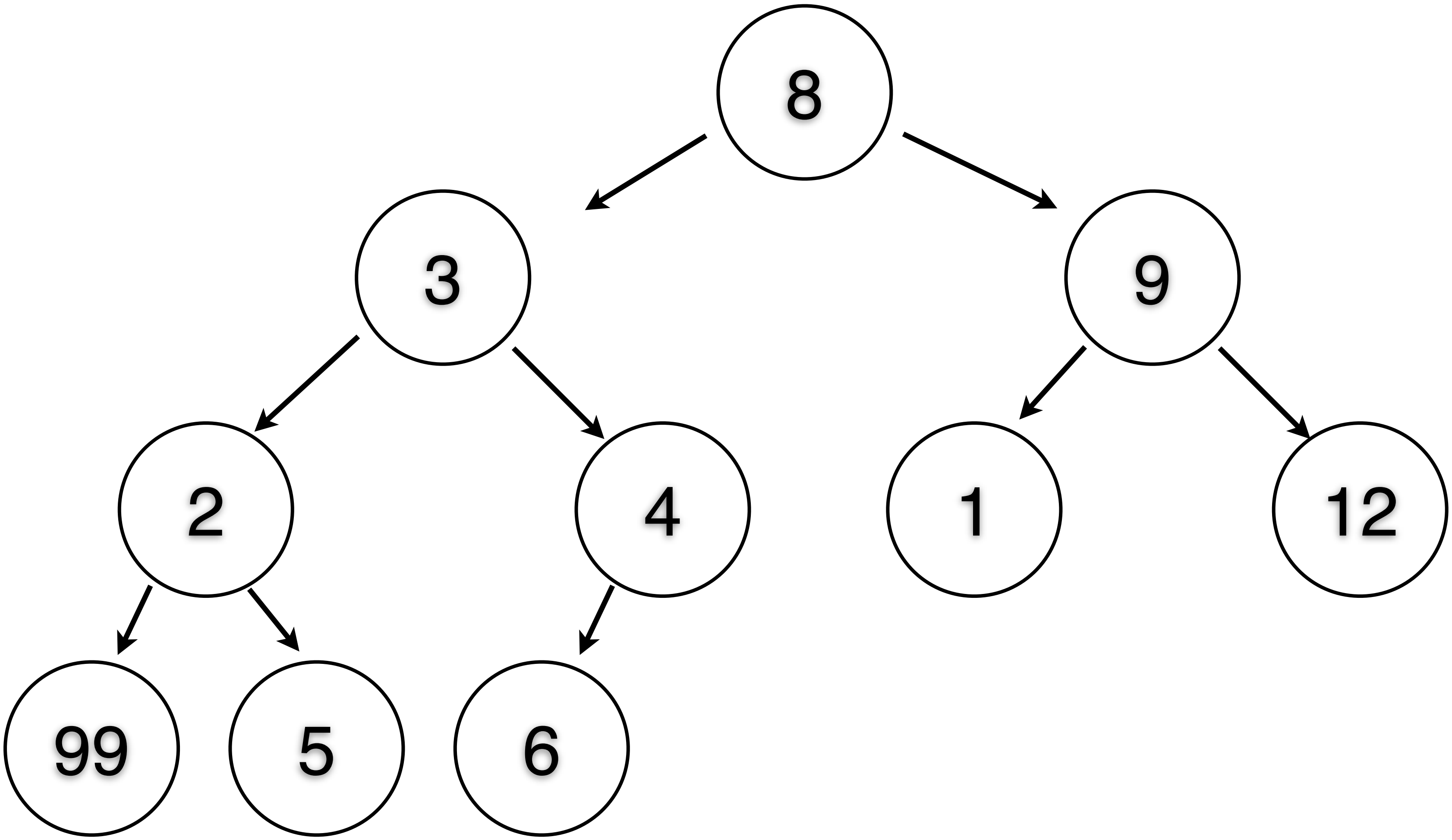
buildHeap Example

8	3	9	2	6	1	12	99	5	4



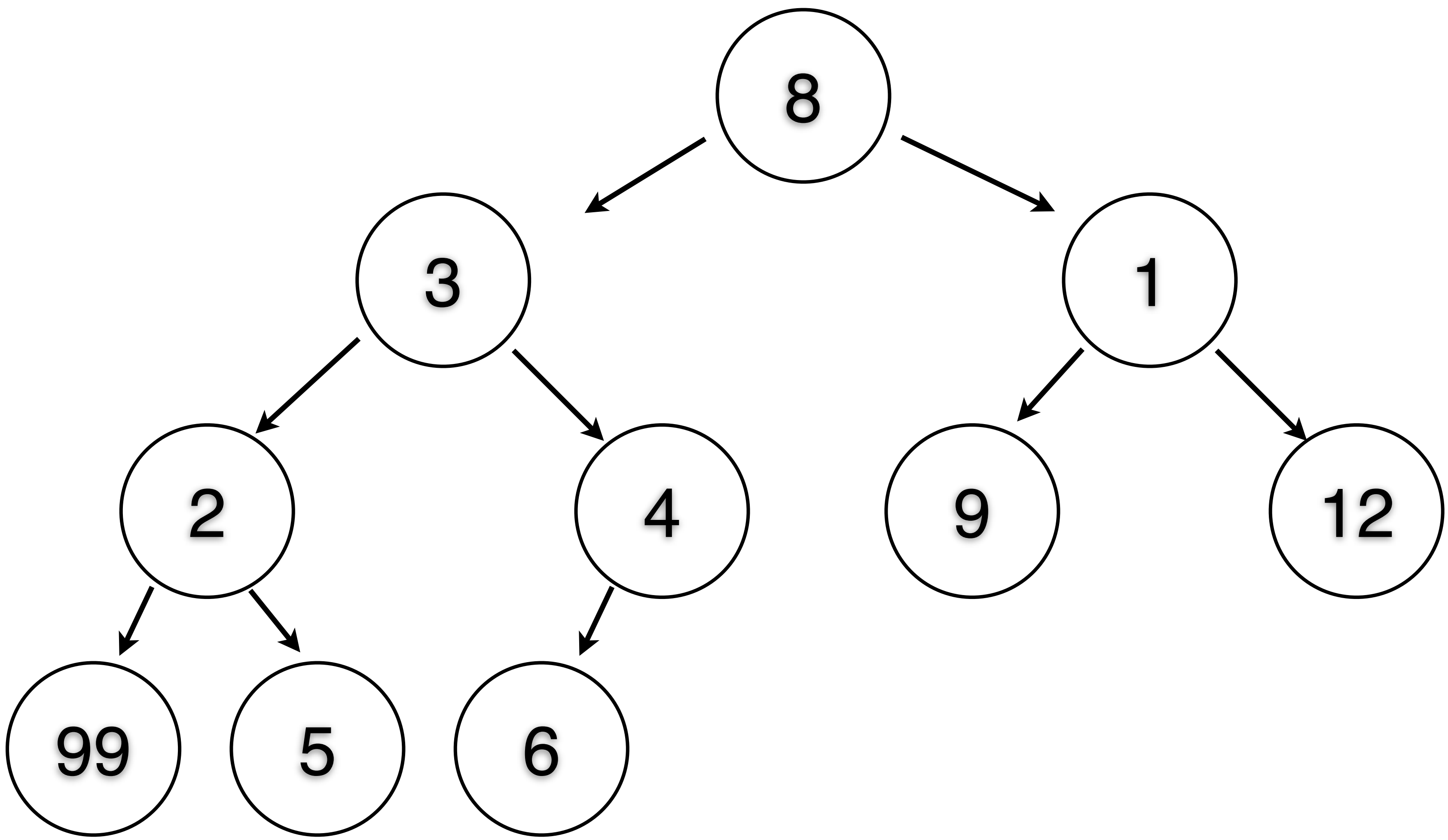
buildHeap Example

8	3	9	2	6	1	12	99	5	4



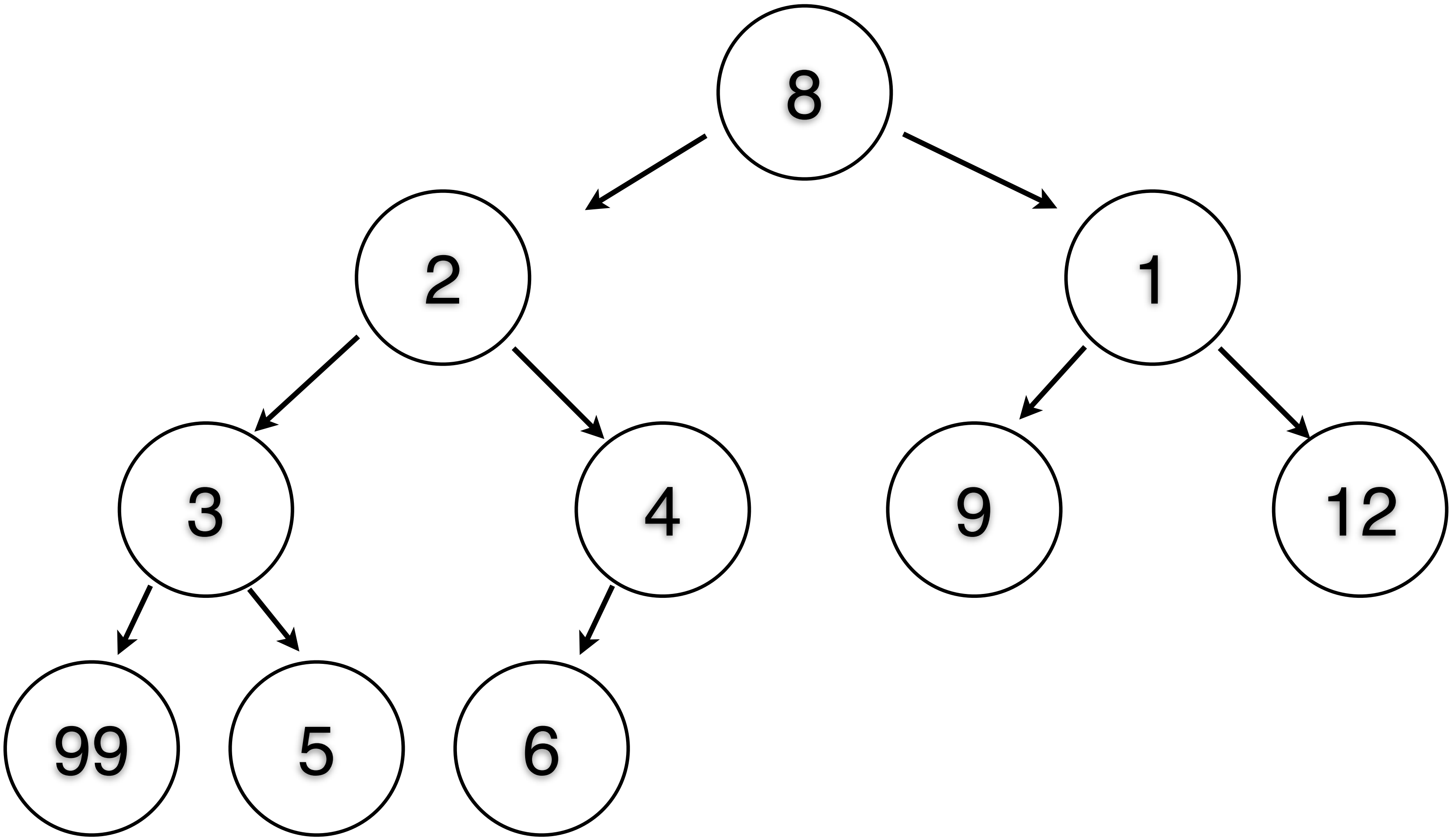
buildHeap Example

8	3	9	2	6	1	12	99	5	4



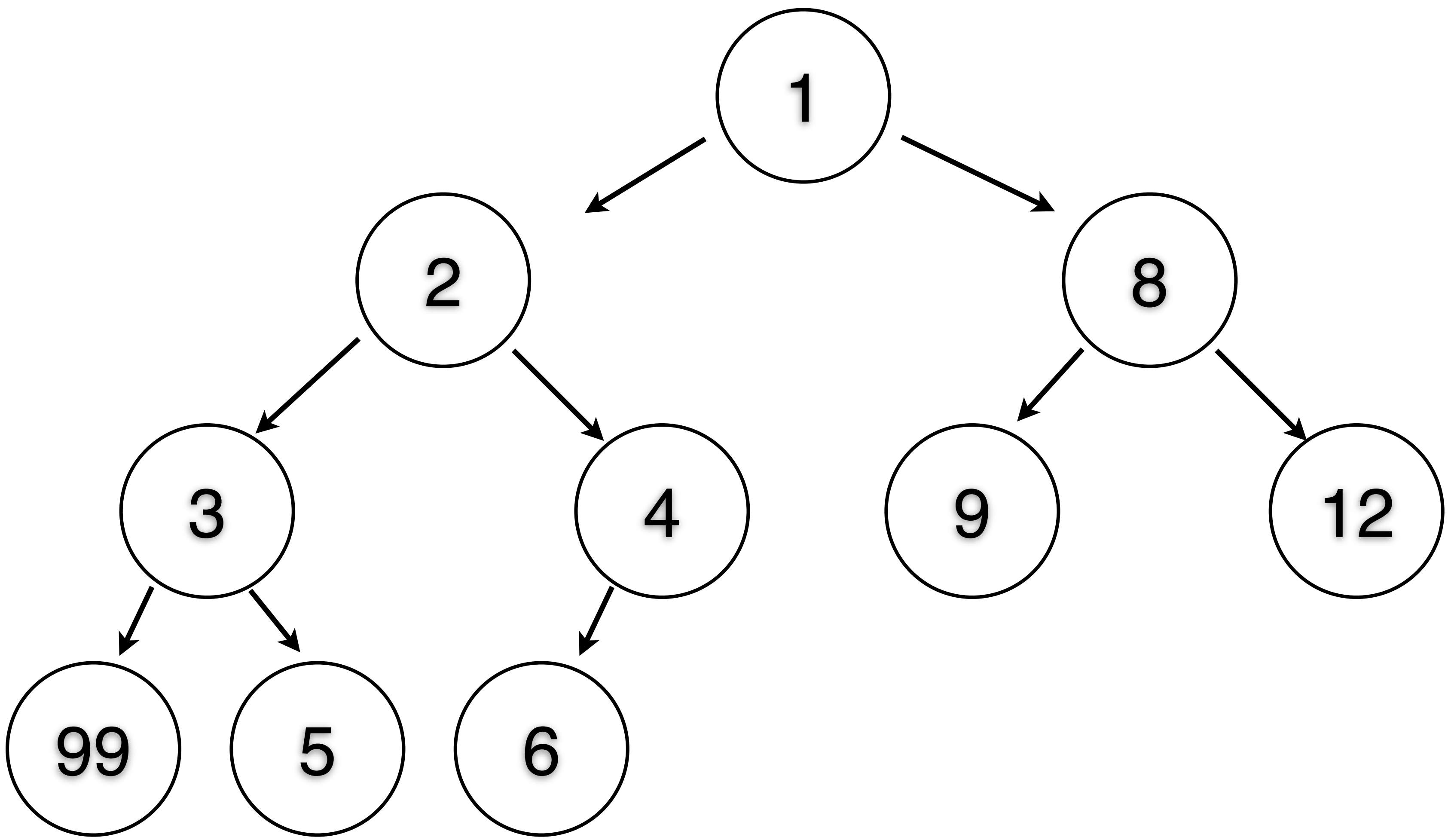
buildHeap Example

8	3	9	2	6	1	12	99	5	4



buildHeap Example

8	3	9	2	6	1	12	99	5	4

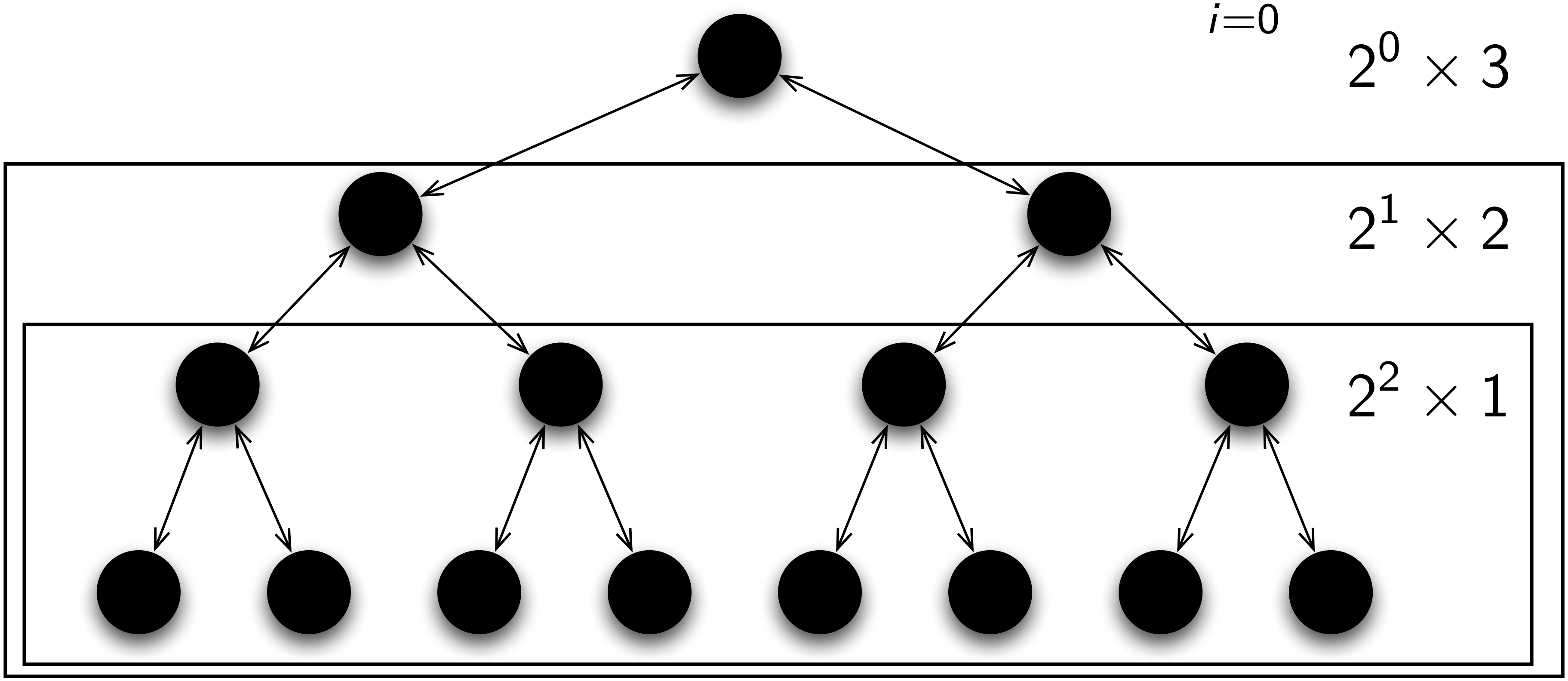


Analysis of buildHeap

- **N/2** percolateDown calls: **O(N log N)**?
  - But calls to deeper nodes are much cheaper
- Percolate Down costs the height of the node
- Let **h** be height of tree. 1 node at height **h**
  - 2 nodes at **(h-1)**, 4 nodes at **(h-2)**...
  - 2^h nodes at height 0

buildHeap Running Time

$$T(2^h) = \sum_{i=0}^{h} 2^i (h - i)$$



Reducing the Summation

$$T(2^h) \quad = \quad \sum_{i=0}^{h} 2^i \,(h-i) \qquad\qquad 2T(N) = \sum_{i=0}^{h} 2^{i+1}(h-i)$$

$$\begin{aligned}
T(N) &= 2T(N) - T(N) = \\
&\quad 2^1(h-0) + 2^2(h-1) + 2^3(h-2) + \ldots + 2^h(1) + 2^{h+1}(0) \\
&- \left[2^0(h-0) + 2^1(h-1) + 2^2(h-2) + \ldots + 2^{h-1}(1) + 2^h(0)\right] \\
\hline
\end{aligned}$$

Reducing the Summation

$$T(2^h) \quad = \quad \sum_{i=0}^{h} 2^i \, (h-i) \qquad\qquad 2T(N) = \sum_{i=0}^{h} 2^{i+1}(h-i)$$

$$T(N) = 2T(N) - T(N) =$$

$$\begin{array}{l} \qquad\qquad 2^1(h-0) + 2^2(h-1) + 2^3(h-2) + \ldots + 2^h(1) + \\ -\left[2^0(h-0) + 2^1(h-1) + 2^2(h-2) + \ldots + 2^{h-1}(1) + 2^h(0)\right] \\ \hline \quad -h \quad + \quad 2^1 \quad + \quad 2^2 \quad + \quad \ldots\ldots \quad + 2^h \end{array}$$

$$= -h + \sum_{i=1}^{h} 2^i = 2^{h+1} - 2 - h$$

Series identity

$$= 2^{\log N + 1} - 2 - \log N = 2N - 2 - \log N$$

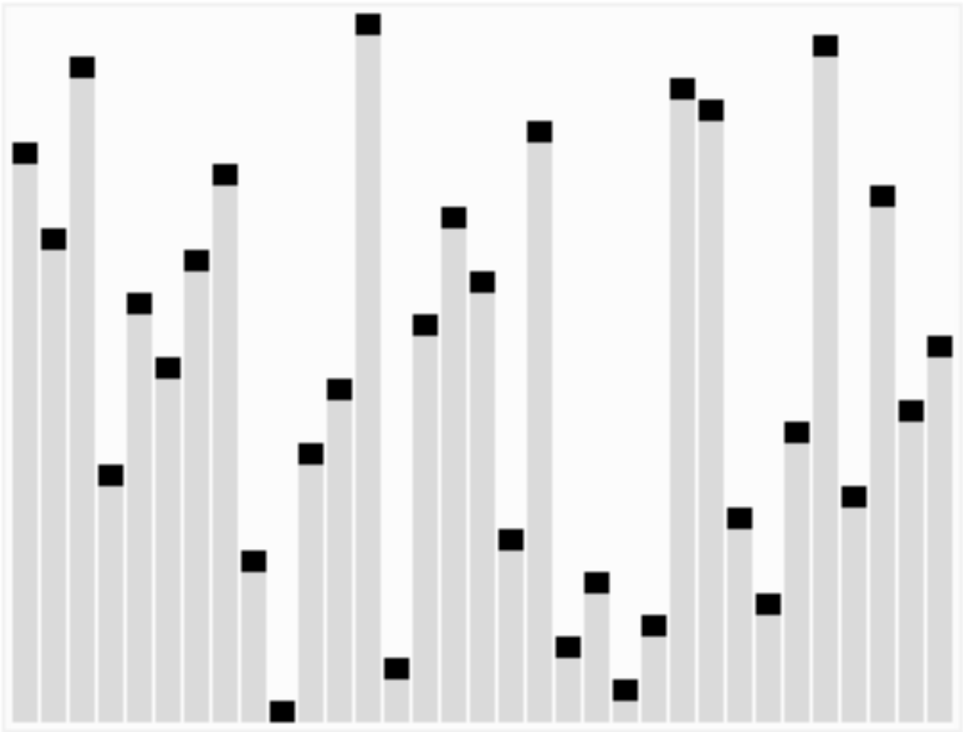
Heap Operations

- Insert – O(log N)
- deleteMin – O(log N)
- change key – O(log N)
- buildHeap – O(N)

HeapSort

- buildHeap, then deleteMin all elements
- but we'd need to store elements in separate array
- Instead, build a max-heap (parent always greater than child; root is max)
- After each delete**Max**, move deleted element to end of array

Heapsort Animation



- image from http://en.wikipedia.org/wiki/Heapsort

Reading

- This class and next: Weiss 6.1-6.3