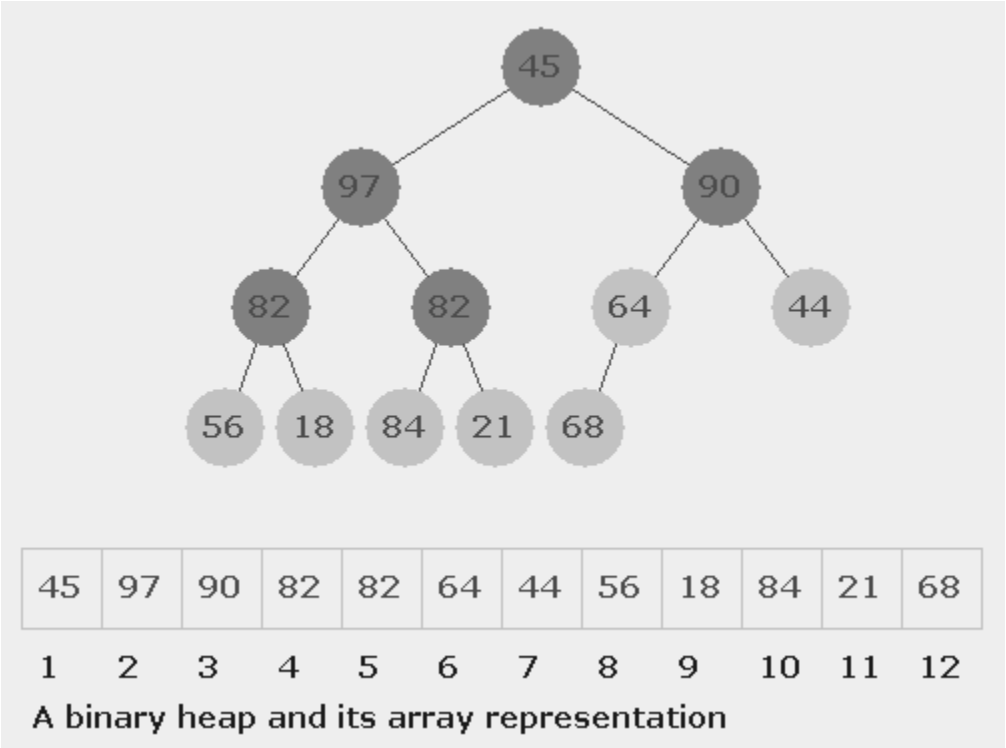
CS 112 Lab 9: Binary heaps

Agenda

Heap representation

A binary heap is a complete binary tree and can be stored in an array as illustrated below. Heap manipulations can be performed using array indices and hence alleviates the need for pointer manipulation.

The left child index of a node N is 2*N, the right child index is (2*N+1). The parent index of N is N/2. Note that it is easier to use array indices starting with 1 when working with heaps.



A binary heap and its array representation

Heap property

Any node in a binary heap is smaller than all other nodes in its subtree. In the above example, nodes which violate the heap property are shown in gray, those which satisfy are in orange.

Heap operations

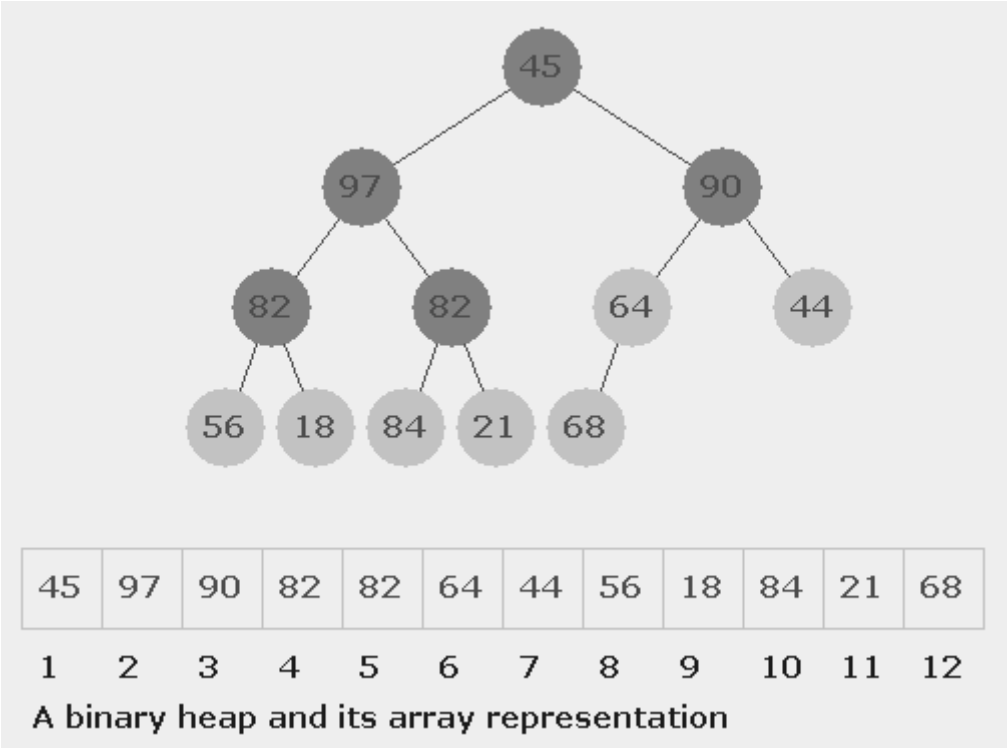
A heap datastructure is designed for very efficient **insert()** and **deleteMin()** operations. A tradeoff is

that operations like find() or merge() can be quite expensive depending on the implementation.

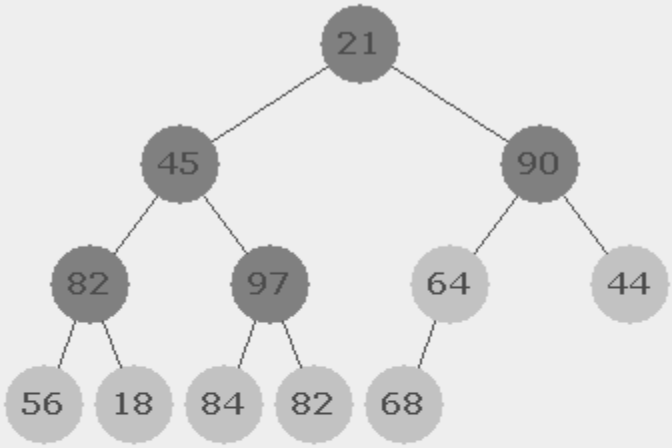
To perform operations on a heap, we need **percolateUp()** and **percolateDown()** to rebalance an unbalanced heap (when some nodes violate the heap property). These operations are best explained with examples illustrated below.

We first balance the above heap via **percolateUp()** and **percolateDown()** operations and then perform insert() and deleteMin() operations.

Each figure below illustrates the resulting heap after applying an operation (displayed in the lower left corner) on the heap in the previous image. The first figure is the initial unbalanced heap.



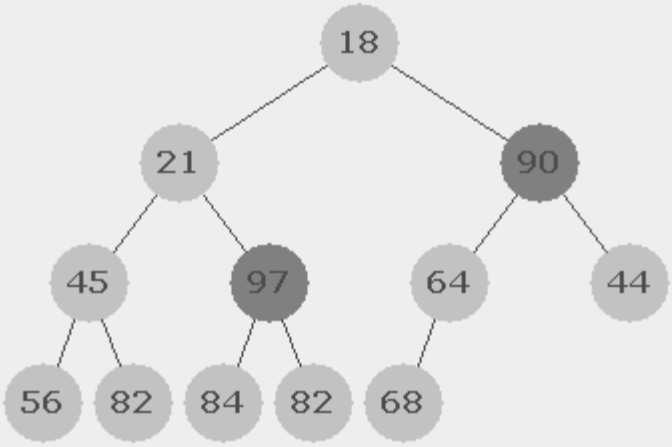
A binary heap and its array representation



21	45	90	82	97	64	44	56	18	84	82	68
1	2	3	4	5	6	7	8	9	10	11	12

A binary heap and its array representation

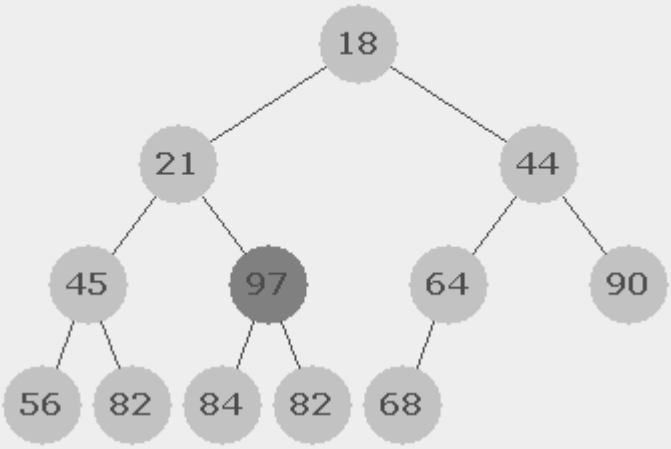
percolateUp(21)



18	21	90	45	97	64	44	56	82	84	82	68
1	2	3	4	5	6	7	8	9	10	11	12

A binary heap and its array representation

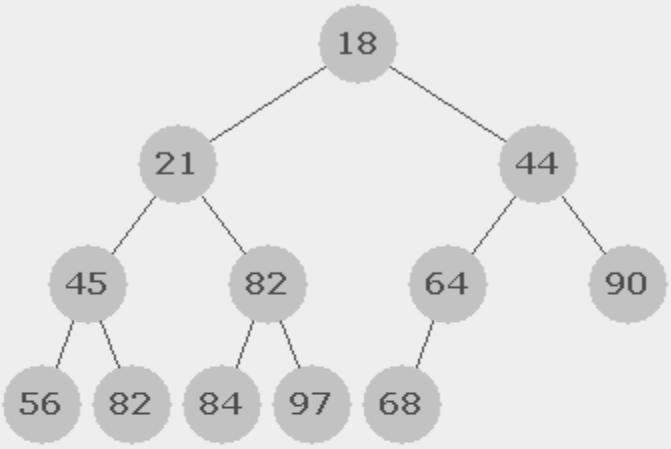
percolateUp(18)



18	21	44	45	97	64	90	56	82	84	82	68
1	2	3	4	5	6	7	8	9	10	11	12

A binary heap and its array representation

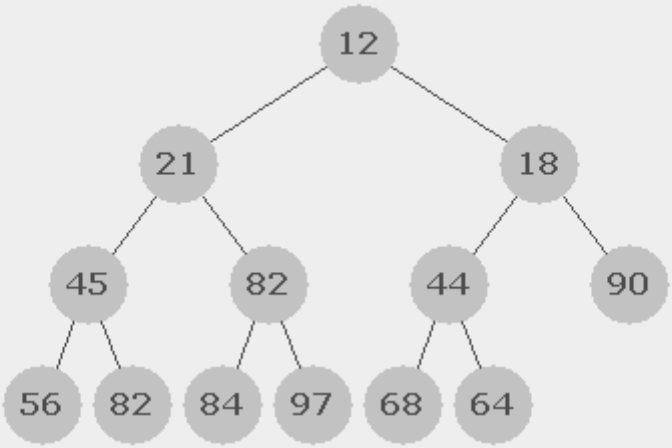
percolateDown(90)



18	21	44	45	82	64	90	56	82	84	97	68
1	2	3	4	5	6	7	8	9	10	11	12

A binary heap and its array representation

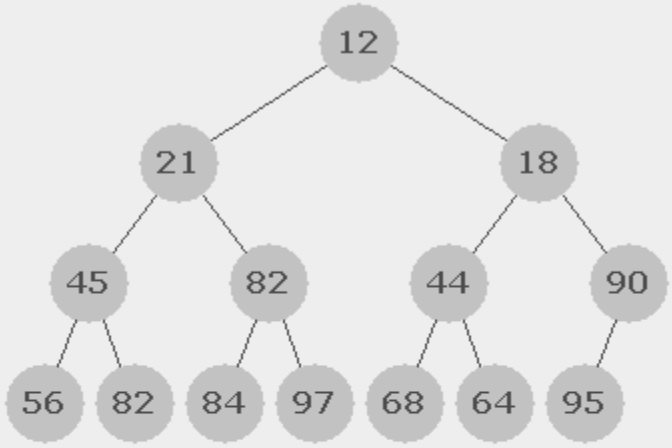
percolateDown(97)



12	21	18	45	82	44	90	56	82	84	97	68	64
1	2	3	4	5	6	7	8	9	10	11	12	13

A binary heap and its array representation

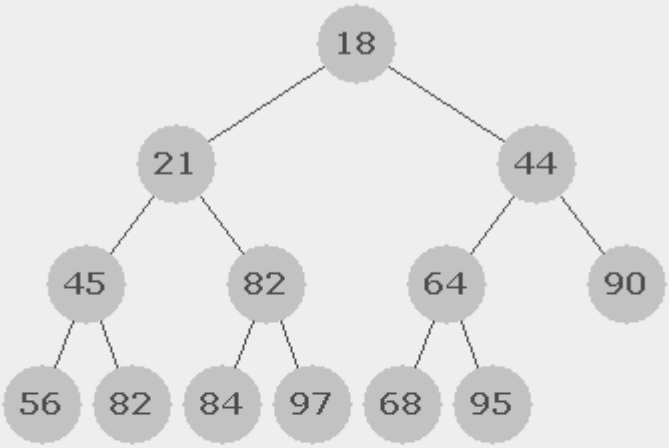
Insert value (12)



12	21	18	45	82	44	90	56	82	84	97	68	64	95
1	2	3	4	5	6	7	8	9	10	11	12	13	14

A binary heap and its array representation

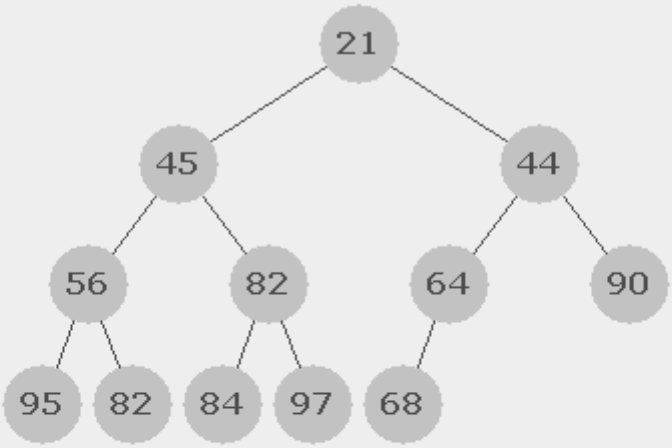
Insert value (95)



18	21	44	45	82	64	90	56	82	84	97	68	95
1	2	3	4	5	6	7	8	9	10	11	12	13

A binary heap and its array representation

deleteMin() = 12



21	45	44	56	82	64	90	95	82	84	97	68
1	2	3	4	5	6	7	8	9	10	11	12

A binary heap and its array representation

deleteMin() = 18

Today's example and associated tasks

Download class files for the lab below,

- BinaryHeap.java

  Contains an array based implementation of binary heap with the above basic heap operations.

  Task: complete percolateUp() and percolateDown() methods following the example above.

- Main.java

  This is our client class to test BinaryHeap. It open a graphics window to display the heap.

URL

http://cs-people.bu.edu/tvashwin/cs112_spring08/lab09.html

Solutions