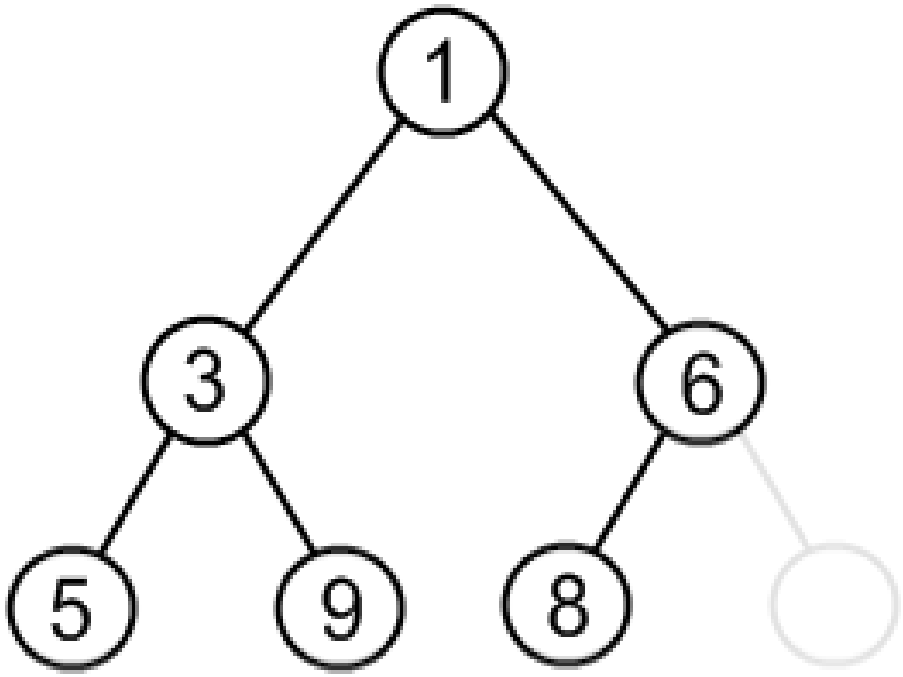
More Tree Definitions

- **Complete Binary Tree:** A binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible. Complete trees are **always** balanced.



Heaps - Motivation

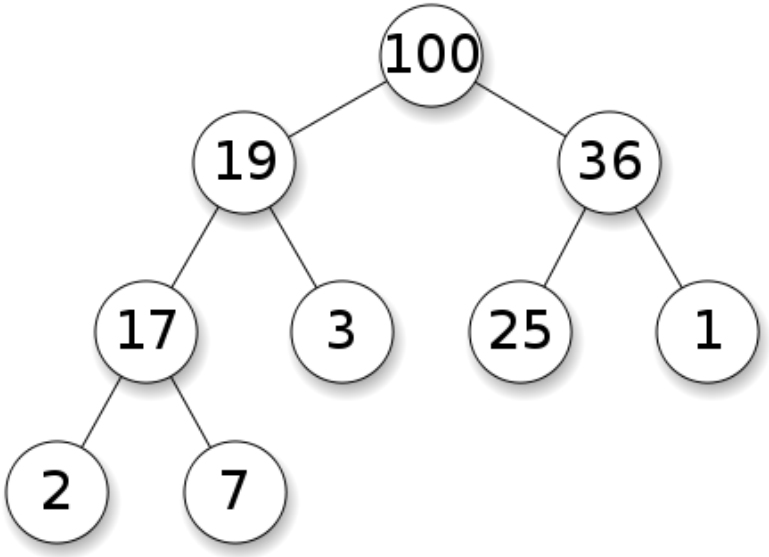
- BSTs are more efficient the closer they are to being ***complete***.

- What if we designed a type of binary tree that would order itself such it will always be ***complete*** even as new items are added?

- This is the concept behind **Heaps**!

Heaps

- Concept: Binary tree where node's value >= values of each of its descendents.



- Essential properties:
  - Tree is always ***complete***.
  - For all nodes, node's value <= parent.value.

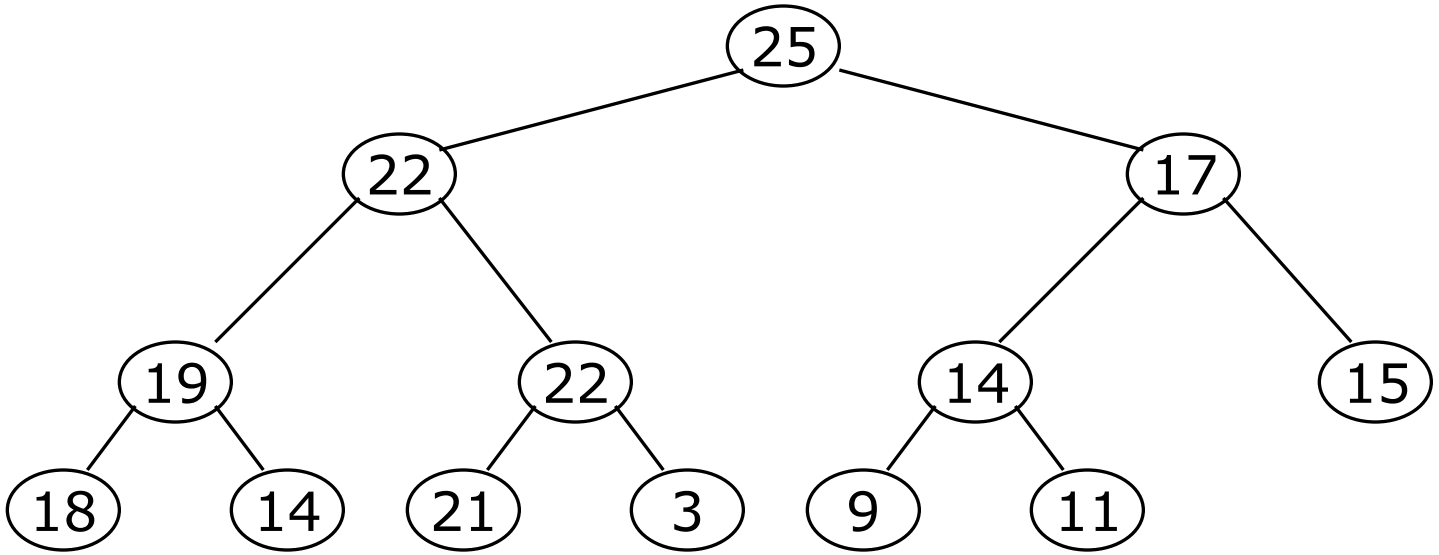
Heap - Implementation

- Since heaps involve nodes, it seems like a reference-based implementation (similar to basic binary tree) would make sense.

- In fact, heaps are usually implemented using an array!

- Because heaps are complete, there will be no wasted space in the array.

Heaps – Array Implementation



0	1	2	3	4	5	6	7	8	9	10	11	12
25	22	17	19	22	14	15	18	14	21	3	9	11

- Notice:
  - The left child of index i is at index 2*i+1
  - The right child of index i is at index 2*i+2
  - Example: the children of node at index 3 (19) are at index 7 (18) and at index 8 (14).

Heap ADT

- What are the essential operations of a Heap?

- `insert(T item)`: Add an item to the heap.
  - Put the item in the next empty spot of the array.
  - Swap it up until it reaches its proper place:

```
while (the new item is > than its parent)
    Swap the new item with its parent
```

Heap ADT

- What are the essential operations of a Heap?

- `delete()`: Returns & removes top value.
  - Store the top item.
  - Overwrite the top item with the last item in the array.
  - Swap the new top down until it reaches its proper place – always compare it with its dominant child:

```
while (item is < than its dominant child)
    Swap with the dominant child.
```

  - Return the top item.

Heaps - Efficiency

Operation	Big-O
heapInsert	O(height)
heapDelete	O(height)

- Heaps are always complete, so their height is ~ $\log_2 n$ where n is the number of items in the heap.

Heaps - Efficiency

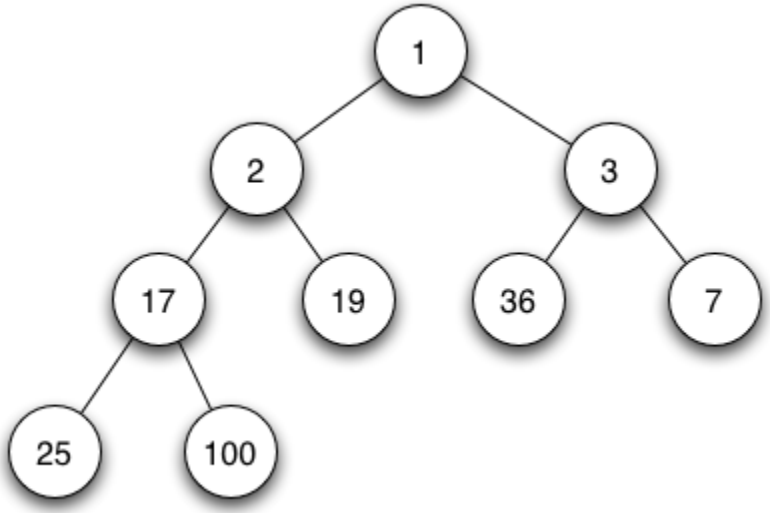
Operation	Big-O
heapInsert	O(log n)
heapDelete	O(log n)

- Heaps are always complete, so their height is ~ $\log_2 n$ where n is the number of items in the heap.

Heap Variants

- The heap we have defined is actually known as a maxheap (or maximum-on-top heap).

- Some applications are better suited to a minheap (minimum-on-top heap) – same idea, just reverse the order.



BSTs vs. Heaps

- **Binary Search Trees** store items sorted left to right. When searching for a particular value, you can determine which subtree to check by checking the value of the current node.

- **Heaps** store items sorted from top to bottom. You cannot search heaps effectively, but you can easily get the ‘top’ value.

Heaps - Application

- Heaps provide '*weak*' order – not as good for searching, but well suited for storing priority.

- Common Heap Applications:
  - Sorting (HeapSort)
  - Priority (Priority queues)
  - Graph Traversals

HeapSort

- **Intuition**: Pulling the data off a heap will return it sorted in descending order.

- Basic HeapSort algorithm:

HeapSort

- **Intuition**: Pulling the data off a heap will return it sorted in descending order.

- Basic HeapSort algorithm:
  - Throw all your data on a heap.

HeapSort

- **Intuition**: Pulling the data off a heap will return it sorted in descending order.

- Basic HeapSort algorithm:
  - Throw all your data on a heap.
  - Grab all your data off the heap.

HeapSort

- **Intuition**: Pulling the data off a heap will return it sorted in descending order.

- Basic HeapSort algorithm:
  - Throw all your data on a heap.
  - Grab all your data off the heap.
  - You're done! Rejoice! (optional)

HeapSort

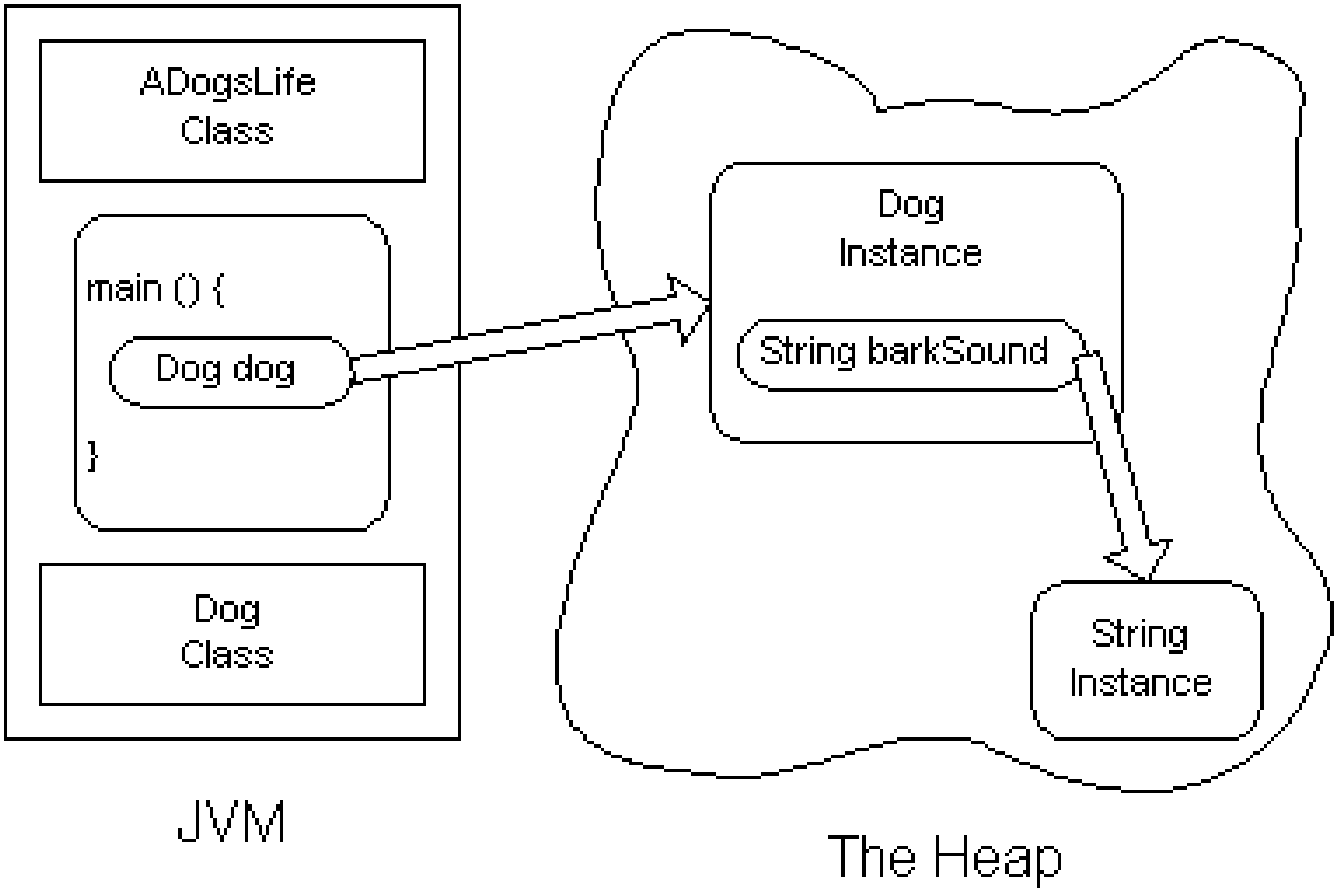
- **Intuition**: Pulling the data off a heap will return it sorted in descending/ascending order (min/max heap).

- Basic HeapSort algorithm:
  - Throw all your data on a heap.
  - Grab all your data off the heap.
  - You're done! Rejoice! (optional)

- HeapSort is O(n log n) – competitive with the fastest known sorting algorithms! See HeapSortDemo.java for an example.

Heaps - Application

- A heap is used to manage memory in the Java Virtual Machine (JVM).



Participation

```
heap.insert(5);
heap.insert(20);
heap.insert(25);
heap.insert(4);
heap.insert(28);
heap.insert(17);

heap.delete();
heap.delete();
```