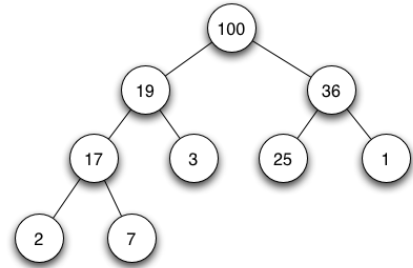


Binary Heap

A binary heap is a data structure which has all the constraints of a binary tree and some additional ones to make it well formed.



I. Language Agnostic Overview

The most common use of binary heaps is in the construction priority queues – in fact it's so frequently used that in the context of priority queues only the term “heap” is necessary to mean binary heap. Priority queues, and thus their underlying data structures, are extremely important in computer science. They ensure the equitable distribution of resources, whether these resources are cpu time or network bandwidth or any other finite resource.

Implementations of binary heaps are typically required to satisfy the following constraints (in addition to those placed on a regular binary tree):

- The tree must be perfectly balanced (all the leaves should be on the same level)
 - If the last row cannot be completely filled, then it should be filled from left to right.
- Every node must be larger than both of its children.

II. Java Specific Overview

The included version of the binary heap is implemented as a min heap for integers. The implementation is straightforward and has only one custom exception (overflow). A new heap can be constructed with either a user-defined size or a preset default. Insertion consists of appending the new element to the bottom of the heap, and then percolating it up until the heap data structure is sound.

Only one field is used in conjunction with the invariants, and that is `currentSize`. Current size is an integer which stores the count of elements in the array.

deleteMin

- This method removes the minimum value from the heap. Since the heap is technically a min-heap, this operation consists of simply removing and repercolating.
- The preconditions check that the heap isn't null, that the size of the heap hasn't somehow been set to a negative value or a value larger than the size of the underlying array, and that the heap is well formed (explained below). The postconditions verify that the array is still well formed.

wellFormed

- This method performs a number of checks to make sure the binary heap is still properly structured and ordered. The two if statements simply verify that if the current node has children, that they are both greater than the parent (the first if statement checks the left child, the second checks the

right).

b. This method has no pre or postconditions as it is used in the evaluation of other method's conditions.

Metrics:

Lines of code for implementation: 72

Methods for implementation: 11

Lines of code for invariants: 20

Methods for invariants: 2

References:

Wikipedia's entry on Binary Heaps: http://en.wikipedia.org/wiki/Binary_heap

Data Structures and Algorithm Analysis in Java, 2nd Edition by Mark Allen Weiss. Section 6.3 “Binary Heap.” pgs 202-214.