CSM 61B
Spring 2018

Heaps Crib Sheet
March 17, 2018

1 Overview

A heap is a tree based data structure that allows for the retrieval of a min/max quickly. A min heap has the following invariants (max heap invariants are analogous):

1. It is a complete binary tree. Each node has up to 2 children and every level except the last is filled. The last level, if not filled, is filled as far left as possible.
2. A node's value is less than both its children's values.

The following are important consequences of the invariants:

1. The root of the heap contains the min value.
2. There are $\log n$ levels, where n is the number of elements in the heap.

2 Implementation

Because the heap is a complete binary tree, it can be represented using an array. Assuming the root is level 0, elements at level i in the heap are placed at indices $[2^i, 2^{i+1} - 1)$ in the array. There is no element placed at index 0 because it makes the following index computations easier.

For some element at `index = k`:

- `leftChildIndex(k) = 2*k`
- `rightChildIndex(k) = 2*k + 1`

3 Operations & Runtimes

Min heaps support the following core operations:

1. `getMin()`: $\Theta(1)$
   - The min is the root of the tree as long as heap invariants are fulfilled.
2. `insert(E elem)`: $O(\log n)$
   - Add `elem` to the leftmost position in the bottom-most level, starting a new level if necessary.
   - If `elem` $<$ parent, swap elem and parent.
   - Repeat until `elem` $>$ parent.

   In the worst case, the new element swims up all levels.

3. `removeMin()`: $O(\log n)$:
   - Swap the root's value with that of the rightmost element on the bottom-most level.
   - Remove the rightmost element on the bottom-most level (which now contains the min value).
   - To find the new root `elem`:
     - Starting from the root, if `elem` $>$ any child, swap with smaller of the two children.
     - Repeat until `elem` $<$ all children.