Heaps

Slightly different from binary search tree:

(1) value in a node is never < values in node's children (**weaker ordering**)

(2) must be a **complete** binary tree (i.e., every level except the deepest must contain as many nodes as possible and at deepest level all nodes are as far left as possible)

Priority Queue

- Highest priority item is first out
- Heap can be used to implement a priority queue
- Runtime for operations is better than other queue implementations ***…why???***

Insertion

Place x in heap in first available location (to maintain a complete binary tree).

while (x's parent < x)
  swap x with its parent

Note: Stops when x becomes the root or when x's parent is no longer < x

Deletion

Remove root node

Remove rightmost entry from deepest level of tree; we'll call it x. Make x the new root.

while (x < one of its children)
  swap x with its highest child

Note: Stops when x becomes a leaf or when x is no longer < one of its children.

Runtime Analysis

Height of heap is $O(\log_2 n)$ where n = # data values

Implementation

Easiest to implement with (dynamic) array

- Nodes in tree stored in partially filled array called *data*
- Root node is in *data[0]*
- For node in *data[i]*, its left child is in *data[2i + 1]* and its right child is in *data[2i + 2]*
- For node in *data[i]*, its parent is in *data[(i – 1)/2]*
- *Used* and *capacity* of array are maintained