Heaps

Priority Queue

Min priority queue

- like a "sack" but only lets you interact w/ the smallest item.

Solution: Tracking the M best things.

getNextMessage(), add it to the priority queue

if size of priority queue > M, remove the smallest.

- How to make these?

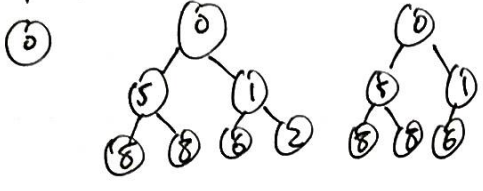
  can't use; Arrays, Bushy BSTs, or HashTables, so use Heaps.

Heaps

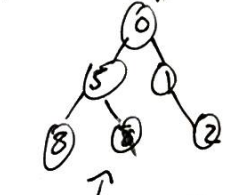
Binary min heap = Binary tree that is complete & obeys the min-heap property.

- Complete: Missing items only at bottom level (if any).
  All nodes are as far left as possible.
- Min-heap: Every node is less than or equal to the children.

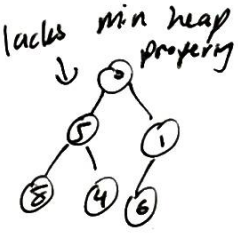
Examples:



Not Examples:



2 not all the way to left.
(Incomplete)

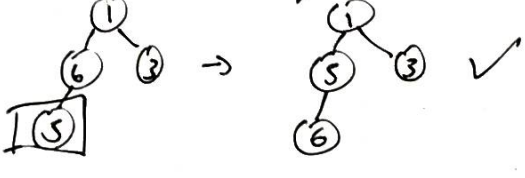


*Note: Every node should have 2 children unless at bottom.

Adding:

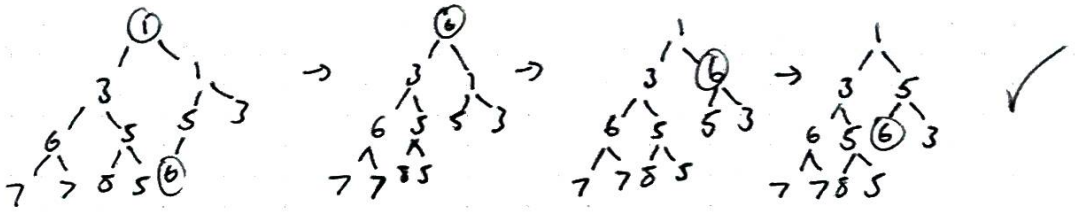
- Insert into an arbitrary leaf position. Have the node climb up and swap as you move along. ensures the right position.

Example:



Deletion of min value

1) Swap last item in heap into the root.
2) Sink the root into its correct spot.



Given a heap:

Priority queue →

getSmallest() → return root node

add(x) → place in the ~~for~~ last position, then promote accordingly.

removeSmallest() → remove root, promote rightmost leaf as root, change positions accordingly.

Heap implementation of a priority queue

	ordered array	Bushy BST	HashTable	Heap
add	$\Theta(n)$	$\Theta(\log n)$	$\Theta(1)$	$\Theta(\log n)$
getSmallest	$\Theta(1)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(1)$
removeSmallest	$\Theta(n)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(\log n)$

hard to handle items w/ same priority.

Notes:

- Priority is useful.
- Heap is log n time Amortized (resizes)
- BST can have a constant getSmallest if you keep a pointer to the smallest value.
- Heaps handle duplicate priorities better than BSTs
- Array based heaps save memory.

keys : [1 a, 2 b, 3 c, 4 d, 5 e, 6 f, 7 g, 8 h, 9 i] parent = (k-1)/2

parents: [0, 0, 0, 1, 1, 2, 2, 3, 3]

can show as:

[a, b, c, d, e, f, g, h, i]