Heaps and Heapsort

Weiss, chapter 21 (pages 807–839)
Priority Queue

• Queues provide first-in-first-out behavior
  – enqueue: adds an item to the end of queue
  – dequeue: removes a item from the head of queue
  – Inefficient for items with priorities

• Dequeue returns an item with the highest priority
• Priority: ordering of items in the queue
Priority Queue (cont.)

• Operations
  – void add( AnyType v ): adds an item considering priority order
  – AnyType poll( ): removes and returns an item with the highest priority
  – AnyType peek( ): returns, but does not remove, an item with the highest priority
    
    After adding 10, 5, and 2: [ 10, 5, 2 ]     peek( ): 10
    
    After adding 15 and 7: [ 15, 10, 7, 5, 2 ]
Binary Heap

• Implement priority queue
  – Order property: for any node, the subtrees of that node have items that are less than or equal to the node item
  – Structure property: all levels of the tree, except the last level, are complete. Last level is filled from left to right.

• Max heap versus min heap

\[
\begin{array}{c}
\text{Parent} \\
6 \quad 5 \quad 3 \quad 2 \quad 4 \quad 1
\end{array}
\]

\[
\begin{array}{c|c|c|c}
\text{Parent} & \text{P. Index} & \text{L.C. index} & \text{R.C index} \\
6 & 0 & 1 & 2 \\
5 & 1 & 3 & 4 \\
3 & 2 & 5 & 6 \\
i & 2i+1 & 2i+2
\end{array}
\]

Parent of node i is floor \((i-1)/2\)
Heap Operations

• peek: returns the element in the first index of the array (O(1))

• add
  – Add to the end of heap O(1)
  – Percolate up O(log N)
Heap Operations (cont.)

• **poll**
  – Analogous to add
  – Move the last element to root $O(1)$
  – Percolate down $O(\log N)$
buildHeap Operation

• Restore the heap order property for a complete binary tree
  – Perform percolate down in reverse order
  – $O(N)$ running time
buildHeap Operation (cont.)

- Running time of buildHeap is defined by the number of swaps
- At worst case, for each node, the number of swaps is many as the height of that node.
- Compute the sum of the heights of nodes
- Sum of the heights is $O(N)$
  - Show by induction
  - Assume the perfect binary tree

<table>
<thead>
<tr>
<th>Height(h)</th>
<th>Sum of heights</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>h</td>
<td>$2^{h+1} - 1 - (h+1)$</td>
</tr>
</tbody>
</table>

Number of nodes ($N$) in the perfect tree with height $h$ is $2^{h+1} - 1$.
Sum of heights of nodes in the perfect tree is $O(N)$. 
Heap Sort

• Sort an unordered array by creating a binary heap (buildHeap) \( (O(N)) \)

• Poll the heap repeatedly \( (O(N \log N)) \)