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 an 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: \([10, 5, 2]\)

  \[
  \text{peek( ): 10}
  \]

After adding 15 and 7: \([17, 10, 7, 5, 2]\)
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

[ 6, 5, 3, 2, 4, 1 ]

<table>
<thead>
<tr>
<th>Parent</th>
<th>P. Index</th>
<th>L.C. index</th>
<th>R.C index</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

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
Heap Sort

- Sort an unordered array by creating a binary heap (buildHeap) \( O(N) \)
- Poll the heap repeatedly \( O(N \log N) \)