A New Data Structure Arrives:

ADT:

insert

remove
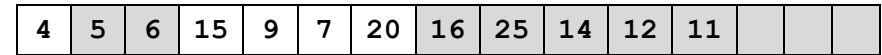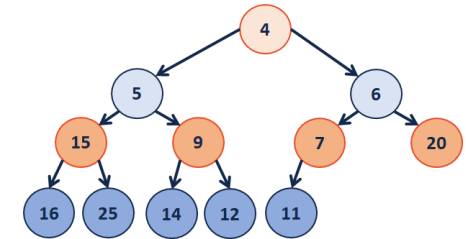
isEmpty

Implementation of _____

insert	removeMin	Implementation
O(n)	O(n)	Unsorted Array
O(1)	O(n)	Unsorted List
O(lg(n))	O(1)	Sorted Array
O(lg(n))	O(1)	Sorted List

Q1: What errors exist in this table? (Fix them!)

Q2: Which algorithm would we use?

Implementing a (min)Heap as an Array

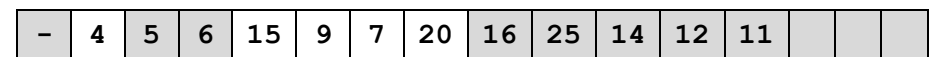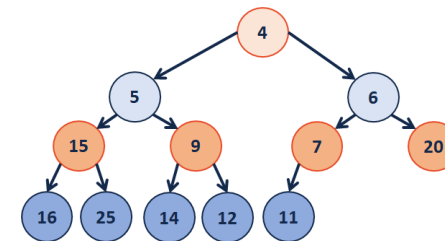| 4 | 5 | 6 | 15 | 9 | 7 | 20 | 16 | 25 | 14 | 12 | 11 | | | |

Operations:

leftChild(index) :=

rightChild(index) :=

parent(index) :=

Inserting into a Heap

| – | 4 | 5 | 6 | 15 | 9 | 7 | 20 | 16 | 25 | 14 | 12 | 11 | | | |

...running time?

<table>
<tr><td colspan="2" align="center">Heap.hpp (partial)</td></tr>
<tr><td>1</td><td><code>template <class T></code></td></tr>
<tr><td>2</td><td><code>void Heap<T>::_insert(const T & key) {</code></td></tr>
<tr><td>3</td><td><code>// Check to ensure there's space to insert an element</code></td></tr>
<tr><td>4</td><td><code>// ...if not, grow the array</code></td></tr>
<tr><td>5</td><td><code>if (size_ == capacity_) { _growArray(); }</code></td></tr>
<tr><td>6</td><td></td></tr>
<tr><td>7</td><td><code>// Insert the new element at the end of the array</code></td></tr>
<tr><td>8</td><td><code>item_[++size] = key;</code></td></tr>
<tr><td>9</td><td></td></tr>
<tr><td>10</td><td><code>// Restore the heap property</code></td></tr>
<tr><td>11</td><td><code>_heapifyUp(size);</code></td></tr>
<tr><td>12</td><td><code>}</code></td></tr>
<tr><td>31</td><td><code>template <class T></code></td></tr>
<tr><td>32</td><td><code>void Heap<T>::_heapifyUp(_____) {</code></td></tr>
<tr><td>33</td><td><code>if (index > _____) {</code></td></tr>
<tr><td>34</td><td><code>if (item_[index] < item_[parent(index)]) {</code></td></tr>
<tr><td>35</td><td><code>std::swap(item_[index], item_[parent(index)]</code></td></tr>
<tr><td>36</td><td><code>);</code></td></tr>
<tr><td>37</td><td><code>_heapifyUp(_____);</code></td></tr>
<tr><td>38</td><td><code>}</code></td></tr>
<tr><td>39</td><td><code>}</code></td></tr>
<tr><td>40</td><td><code>}</code></td></tr>
</table>

What's wrong with this code?

Heap Operation: removeMin / heapifyDown:

| – | 4 | 5 | 6 | 15 | 9 | 7 | 20 | 16 | 25 | 14 | 12 | 11 | | | |

<table>
<tr><td colspan="2" align="center">Heap.hpp (partial)</td></tr>
<tr><td>1</td><td><code>template <class T></code></td></tr>
<tr><td>2</td><td><code>void Heap<T>::_removeMin() {</code></td></tr>
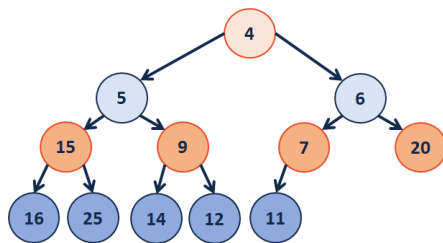<tr><td>3</td><td><code>// Swap with the last value</code></td></tr>
<tr><td>4</td><td><code>T minValue = item_[1];</code></td></tr>
<tr><td>5</td><td><code>item_[1] = item_[size_];</code></td></tr>
<tr><td>6</td><td><code>size--;</code></td></tr>
<tr><td>7</td><td></td></tr>
<tr><td>8</td><td><code>// Restore the heap property</code></td></tr>
<tr><td>9</td><td><code>heapifyDown();</code></td></tr>
<tr><td>10</td><td></td></tr>
<tr><td>11</td><td><code>// Return the minimum value</code></td></tr>
<tr><td>12</td><td><code>return minValue;</code></td></tr>
<tr><td>13</td><td><code>}</code></td></tr>
<tr><td>1</td><td><code>template <class T></code></td></tr>
<tr><td>2</td><td><code>void Heap<T>::_heapifyDown(int index) {</code></td></tr>
<tr><td>3</td><td><code>if (!_isLeaf(index)) {</code></td></tr>
<tr><td>4</td><td><code>T minChildIndex = _minChild(index);</code></td></tr>
<tr><td>5</td><td><code>if (item_[index] ____ item_[minChildIndex]) {</code></td></tr>
<tr><td>6</td><td><code>std::swap(item_[index], item_[minChildIndex]);</code></td></tr>
<tr><td>7</td><td><code>_heapifyDown(_____);</code></td></tr>
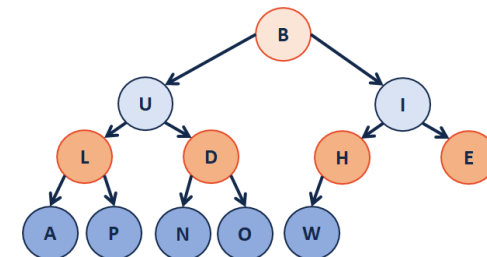<tr><td>8</td><td><code>}</code></td></tr>
<tr><td>9</td><td><code>}</code></td></tr>
<tr><td>10</td><td><code>}</code></td></tr>
</table>

Q: How do we construct a heap given data?

| – | B | U | I | L | D | H | E | A | P | N | O | W | | | |

<table>
<tr><td align="center">CS 225 – Things To Be Doing:</td></tr>
<tr><td>
1. Exam 2 onging
2. mp_mosaix due next Monday (Nov. 1)
3. lab_hash is out tomorrow, due on Sunday (Oct. 31)
4. Daily POTDs are ongoing :)
</td></tr>
</table>