A Heap Data Structure
(specifically a minHeap in this example, as the minimum element is at the root)

Heap Operation:
removeMin / heapifyDown
insert / heapifyUp

Q: How do we construct a heap given data?

Running Time?
Theorem: The running time of buildHeap on array of size n is:
__________.

Strategy:

Define S(h):
Let S(h) denote the sum of the heights of all nodes in a complete tree of height h.

S(0) =
S(1) =
S(h) =

Proof of S(h) by Induction:

Ideas
1) 
2) 
3) 

Finally, finding the running time:
**Disjoint Sets**

Let $R$ be an equivalence relation. We represent $R$ as disjoint sets:
- Each element exists in exactly one set.
- Every set is an equitant representation.
  - Mathematically: $4 \in [0]_R \Rightarrow 8 \in [0]_R$
  - Programmatically: $\text{find}(4) == \text{find}(8)$

**Building Disjoint Sets:**
- Maintain a collection $S = \{s_0, s_1, \ldots, s_k\}$
- Each set has a representative member

```cpp
void makeSet(const T & t);
void union(const T & k1, const T & k2);
T & find(const T & k);
```

**Operation:** find($k$)

**Operation:** union($k_1, k_2$)

**Implementation #2:**
- Continue to use an array where the index is the key
- The value of the array is:
  - -1, if we have found the representative element
  - The index of the parent, if we haven’t found the rep. element

---

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
</table>

---

What is the running time of find?

What is the ideal UpTree?

How do we want to union the two UpTrees?

---

**CS 225 – Things To Be Doing:**

1. mp_mosaics due date: Monday, November 1
2. lab_hash is due Sunday, October 31
3. Daily POTDs are ongoing!